



Provided by the author(s) and University of Galway in accordance with publisher policies. Please cite the published version when available.

Title	On Applying Controlled Natural Languages for Ontology Authoring and Semantic Annotation
Author(s)	Davis, Brian Patrick
Publication Date	2013-02-07
Item record	http://hdl.handle.net/10379/4176

Downloaded 2024-05-16T06:02:14Z

Some rights reserved. For more information, please see the item record link above.





OÉ Gaillimh
NUI Galway

On Applying Controlled Natural Languages for Ontology Authoring and Semantic Annotation

Brian Patrick Davis

Submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy

PRIMARY SUPERVISOR:
Prof. Dr. Siegfried Handschuh
National University of Ireland Galway

SECONDARY SUPERVISOR:
Professor Hamish Cunningham
University of Sheffield

INTERNAL EXAMINER:
Prof. Dr. Manfred Hauswirth
National University of Ireland Galway

EXTERNAL EXAMINER:
Professor Chris Mellish
University of Aberdeen

EXTERNAL EXAMINER:
Professor Laurette Pretorius
University of South Africa

Digital Enterprise Research Institute,
National University of Ireland Galway.
February 2013

Declaration

I declare that the work covered by this thesis is composed by myself, and that it has not been submitted for any other degree or professional qualification except as specified.

Brian Davis

The research contributions reported by this thesis was supported(in part) by the Lion project supported by Science Foundation Ireland under Grant No. SFI/02/CE1/I131 and (in part) by the European project NEPOMUK No FP6-027705.

Abstract

Creating formal data is a high initial barrier for small organisations and individuals wishing to create ontologies and thus benefit from semantic technologies. Part of the solution comes from ontology authoring, but this often requires specialist skills in ontology engineering. Defining a Controlled Natural Language (CNL) for formal data description can enable naive users to develop ontologies using a subset of natural language. However despite the benefits of CNLs, users are still required to learn the correct syntactic structures in order to use the Controlled Language properly. This can be time consuming, annoying and in certain cases may prevent uptake of the tool. The reversal of the CNL authoring process involves generation of the controlled language from an existing ontology using Natural Language Generation (NLG) techniques, which results in a round trip ontology authoring environment: one can start with an existing imported ontology (re)produce the CNL using NLG, modify or edit the text as required and subsequently parse the text back into the ontology using the CNL authoring environment. By introducing language generation into the authoring process, the learning curve associated with the CNL can be reduced. While the creation of ontologies is critical for the Semantic Web, without a critical mass of richly interlinked metadata, this vision cannot become a reality. Manual semantic annotation is a labor-intensive task requiring training in formal ontological descriptions for the otherwise non-expert user. Although automatic annotation tools attempt to ease this knowledge acquisition barrier, their development often requires access to specialists in Natural Language Processing (NLP). This challenges researchers to develop user-friendly annotation environments. While CNLs have been applied to ontology authoring, little research has focused on their application to semantic annotation. In summary, this research applies CNL techniques to both ontology authoring and semantic annotation, and provides solid empirical evidence that for certain scenarios applying CNLs to both tasks can be more user friendly than standard ontology authoring and manual semantic annotation tools respectively.

Acknowledgements

I would first like to thank Professor Hamish Cunningham, University of Sheffield, for giving me the opportunity to pursue postgraduate study and start an academic career at DERI, NUI Galway. In particular, I want to thank him and the GATE team for their support and invaluable training,

Most importantly, I would like to thank my supervisor, Prof. Dr. Siegfried Handschuh for his patience, guidance and mentorship with respect to scientific writing, research project management, teaching, proposal writing, research leadership etc and for giving me countless opportunities to development my research skills and academic career.

Other academics, I would like to thank include Prof. Dr. Stefan Decker, for his inspiration, Dr Conor Hayes for his practical advice on research project management and Dr Paul Buitelaar for being an veritable treasure chest of natural language processing experience.

I would like to thank Prof. Chris Mellish and Prof Laurette Pretorius, (who made the long journey from South Africa) and Prof. Dr. Manfred Hauswirth for ensuring that the final submission is in pristine condition, such that in later years I will be able to glance through this thesis with pride!

DERI (now INSIGHT) has and continues to be a stimulating environment to study and work in. I have met so many wonderful, intelligent and helpful people and made so many friends and acquaintances over the years that to list them all would be impossible! I want in particular thank the old Nepomuk/SMILE team past and present: Tudor, Knud, Laura, Cipri, Georgeta, Ioana, Ismael, Simon, VinhTuan, Alex, Pradeep, Vit, Manuela,

Keith, Judy and Jeremy, who have all gone on to bigger and better things! A particular thanks to Behrang for always raising the NLP bar with his vast knowledge and wonderful discussions! Thanks to Luk, Laleh and Bahareh for tea, friendship and the occasional needed kick in the backside!

Thanks to the DERI Operations, Administrative and Technical team over the years, which without, DERI would simply have ground to a halt: Hilda, Claire, Brian Wall, Ger and Andrew (and Caragh!) Gallagher, Carmel, Michelle and Sylvia and all the Irish not forgetting Ed and Sean! You have put up with a lot over the years!

To my other baby OCD Ireland, special thanks to the team past and present and my good friend Leslie for her wisdom over the years and for taking on so much extra work in the last year so I could finish thesis! A special mention to Prof. Fionnuala as well!

To all my childhood friends in Dublin, Neil, Ian and Ross, Eoin and Mark, who had no idea what I was studying and why it took so long, but have made me feel extremely proud! To staff of the former M. Davis and Co. for always making me feel welcome and not forgetting Rose and the infamous Mr Kitt!

I want to thank my mother and father, Louis and Mary for their unconditional love, support, understanding and patience over the years, as well as my brother Colm and sister Louise. I want to acknowledge Mark and Yvonne, and my army of nieces and nephews, Caragh, Eoghan, Amelie and my Goddaughters Niamh and Ella and all the extended family!!

I also want to thank my fiancé Alessandra, who has always been a steady force of love, patience and much needed motivation to finish, this thesis even when I resisted!

Finally, I want to mention my father, Louis Davis who passed away unexpectedly a year before this thesis was defended. I miss you very much Dad, and I wish you were here to celebrate with me.

*Dedicated to my father, Louis Michael Davis
(1940 - 2012).*

Table of Contents

Abstract	iv
Acknowledgements	v
Table of Contents	xi
List of Figures	xiv
List of Tables	xviii
I Prelude	1
1 Introduction	2
1.1 Motivation and Problem Description	2
1.2 Research Questions	6
1.3 Contributions	7
1.4 Thesis Structure	8
II Foundations	10
2 Human Language Technology and the Semantic Web	11
2.1 The Semantic Web and Linked Data	11
2.2 Human Language Technology	25

2.3	Information Extraction	28
2.4	Ontology based Information Extraction (OBIE)	40
2.5	Natural Language Generation	46
2.6	Natural Language Generation from Ontologies	53
2.7	Conclusions	59
3	Controlled Natural Languages and Semantic Annotation	63
3.1	Control Natural Languages for the Semantic Web	63
3.2	Semantic Annotation	74
3.3	Conclusions	87
III	Core Research	92
4	CLOnE - Controlled Language for Ontology Editing	93
4.1	Introduction	93
4.2	Requirements and Implementation	95
4.3	Evaluation	101
4.4	Related Work	110
4.5	Conclusion	113
5	Round Trip Ontology Authoring	115
5.1	Introduction	115
5.2	Design and Implementation	117
5.3	Evaluation	127
5.4	Related work	137
5.5	Conclusion & Discussion	143
6	Towards Controlled Natural Language for Semantic Annotation	146
6.1	Introduction	146
6.2	CLANN: Design and Implementation	154

6.3	Evaluation	171
6.4	Related work	184
6.5	Conclusion and Future Work	188
IV	Conclusion and Future Work	191
7	Conclusions and Future Work	192
7.1	Research Contributions	192
7.2	Open Questions	195
7.3	Ongoing Work with CLANN	196
7.4	Future Work	197
7.5	Summary	199
	Bibliography	201
A	Evaluation documents for CLIE/CLOnE	231
A.1	Training manual	231
A.2	Test procedure, tasks and questionnaires	247
B	Evaluation documents for Round Trip Ontology Authoring ROA	257
B.1	Training manual	257
B.2	Test procedure, tasks and questionnaires	273
C	Evaluation documents for Controlled Annotation	283
C.1	Training manual	283
C.2	Controlled Language ANNotator Type II (CLANN II)	298
C.3	OntoMat Annotatizer	312
C.4	Test procedure, tasks and questionnaires	334

List of Figures

2.1	The Semantic Web layer cake.	15
2.2	Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/ , Retrieved Tue 23 Jul 2013 14:36:02 IST	25
2.3	Performance trade-off relative to specificity and complexity for IE (ex- tracted from [Cunningham, 2005], Fri 18 Jan 2013 11:24:39 GMT	30
4.1	The CLIE pipeline	97
5.1	The ROA RoundTrip Ontology Authoring pipeline	119
5.2	Generated CLOnE output	121
5.3	ROA Ontology viewer	123
5.4	Example of a generation template	126
5.5	Text Generated by ROA	129
6.1	Controlled Annotation and the Semantic Web <i>Information Food Chain</i>	152
6.2	Meeting Minutes Template for CLANN I and II	158
6.3	Overview of CLANN I and II in GATE	159
6.4	CLANN I pipeline	161
6.5	CLANN I visualised in GATE	162
6.6	An example of a semantic annotation of type <code>Task</code> in CLANN I	163
6.7	CLANN II pipeline	168
6.8	CLANN II visualised in GATE	169

6.9	An example of a semantic annotation of type Conference in CLANN II	169
A.1	Graphical depiction of classes and instances	233
A.2	CLIE Text input	235
A.3	CLIE Ontology viewer	236
A.4	Symbols used in CLIE	236
A.5	Reserved words and phrases in CLOnE	241
A.6	Protégé’s Subclass Explorer and Class Editor	243
A.7	Protégé’s Property Browser and Property Editor	244
A.8	Protégé’s Instance Browser and Individual Editor	246
A.9	Symbols used in Protégé	246
A.10	Initial ontology (viewed in GATE)	248
A.11	Task list A	249
A.12	Intermediate ontology (viewed in GATE)	250
A.13	Task list B	251
A.14	Final ontology (viewed in GATE)	252
A.15	Pre-test questionnaire	253
A.16	Post-test questionnaire for each system	254
A.17	Post-test questionnaire comparing the tools	255
A.18	Post-test questionnaire comparing the tools	256
B.1	Graphical depiction of classes and instances	258
B.2	CLOnE Text input	261
B.3	ROA Ontology viewer	262
B.4	Protégé’s Subclass Explorer and Class Editor	269
B.5	Protégé’s Property Browser and Property Editor	270
B.6	Protégé’s Instance Browser and Individual Editor	272
B.7	Symbols used in Protégé	272
B.8	Initial ontology (viewed in GATE)	274

B.9	Task list A	275
B.10	Intermediate ontology (viewed in GATE)	276
B.11	Task list B	277
B.12	Final ontology (viewed in GATE)	278
B.13	Pre-test questionnaire	279
B.14	Post-test questionnaire for each system	280
B.15	Post-test questionnaire comparing the tools	281
B.16	Post-test questionnaire comparing the tools	282
C.1	Graphical depiction of classes and instances	284
C.2	Our use of the terms ontology, annotation and relational metadata	286
C.3	Overview of CLANN I and II in GATE	288
C.4	Initial Sample meeting minutes in free or uncontrolled text	290
C.5	Meeting Minutes Template for CLANN I and II in BNF	294
C.6	CLANN I - Final Correct Output	295
C.7	Sample of reserved words and phrases in CLANN I	297
C.8	Reserved template phrases and punctuation in CLANN 2	297
C.9	CLANN II - Initial Sample Text	302
C.10	CLANN II - Meeting Minutes Template	303
C.11	CLANN II - Correct Output	304
C.12	CLANN II - Correct Output Continued	305
C.13	CLANN II - Final Correct Output	306
C.14	Reserved template phrases and punctuation in CLANN 2	311
C.15	Opening Ontology Browser in Ontomat	313
C.16	Opening Web Browser in Ontomat	313
C.17	Opening a HTML file in Ontomat	314
C.18	Opening an OWL Ontology in Ontomat	315
C.19	Selecting text in Ontomat	315
C.20	Creating an instance in Ontomat	316

C.21 Creating Attributes in Ontomat	317
C.22 Creating Relations in Ontomat	318
C.23 Example Meeting Minutes for OntoMat	320
C.24 Creating a new instance of a Meeting in Ontomat.	321
C.25 Annotating the Date of the Meeting in the text using “hasDateLiteral”.	322
C.26 Annotate the Project name of the Meeting in Ontomat	323
C.27 Create Person instances for each person a note	324
C.28 Create and instance of Meeting Chair	325
C.29 Assign the Role of Chair to a Person	326
C.30 Assign Claudia (ClaudiaChair) as Chair	327
C.31 Create an Agenda Item in Ontomat	328
C.32 Link the Agenda Item to the current Meeting instance	329
C.33 Create relation Metadata	330
C.34 Linking to the Agenda Item	331
C.35 Create an instances of type Comment	332
C.36 Saving the Ontology in Ontomat	333
C.37 Exporting Metadata in Ontomat	333
C.38 Task A	335
C.39 Task B	336
C.40 Task C	337
C.41 Pre-test questionnaire	338
C.42 Post-test questionnaire for each Tool	339
C.43 Post-test questionnaire comparing the tools	340
C.44 Post-test questionnaire comparing the tools	341
C.45 Post-test questionnaire comparing CLANN I and CLANN II	342
C.46 Post-test questionnaire comparing the tools	343

List of Tables

4.1	Groups of equivalent sentences in CLOnE	95
4.2	Summary of the questionnaire scores	105
4.3	Confidence intervals (95%) for the SUS scores	105
4.4	Correlation coefficients	106
4.5	Groups of subjects by source and tool order	107
4.6	Comparison of the two sources of subjects	108
5.1	Excerpt of CLOnE grammar with examples	120
5.2	Summary of the questionnaire scores	132
5.3	Summary of the Task per Tool times	132
5.4	Confidence intervals (95%) for the SUS scores	133
5.5	Correlation coefficients	134
5.6	Groups of subjects by source and tool order	135
5.7	Comparison of the two sources of subjects	136
6.1	Excerpt of CLANN I grammar with examples	166
6.2	Excerpt of CLANN II grammar with examples	168
6.3	Groups of subjects by source and tool order	174
6.4	Summary of the questionnaire scores	176
6.5	Summary statistics for tool times	176
6.6	Correlation coefficients	178
6.7	Comparison of SUS scores against backgrounds	180

6.8	Comparison of task times against backgrounds	181
6.9	Summary of the Precision and Recall scores for each tool	184

Part I

Prelude

1 Introduction

1.1 Motivation and Problem Description

The Semantic Web endeavours to extend the current Web, by enriching information with well defined meaning, which is machine processable [Berners-Lee et al., 2001a]. It envisions the idea of having data on the Web defined and linked in a uniform manner that can be utilised and exploited by machines not just for display and visualisation purposes but for automation, integration and reuse of data across various applications [Kashyap et al., 2008a]. This would result in an environment, whereby intelligent agents can interact freely across web resources and engage in sophisticated tasks for users.

In order for the Semantic Web to become a reality, we need, as a *primer inter pares*, semantic data. The process of providing semantic data is called Semantic Annotation, because it frequently involves the embellishment of existing data, i.e. the text, with semantic metadata, which can subsequently describe the associated text. This process is heavily dependent on the existence of ontologies, which describe the domain of interest. Ontologies are considered one of the pillars of the Semantic Web, although their definition in the literatures tends to vary [Gómez-Pérez et al., 2007]. Studer et al define an ontology “as a formal explicit specification of a shared conceptualisation” [Gruber, 1993, Studer et al., 1998]. It is formal in that it must be machine readable, explicit in that the types of concepts used and their respective constraints must be explicitly defined and shared. in that it is not private but must be accepted by some group [Studer et al., 1998]. Both Semantic Annotation and ontologies are interconnected, whereby Semantic Annotation requires ontologies to drive the the annotation process and the subsequent association

of instances and relations within text to the ontology, while on the other hand, ontologies may be extended by instance population from facts discovered in text. This can occur as a side-effect of the annotation process. Consequently (i) Ontology Authoring and (ii) Semantic Annotation are two of the core challenges for building the Semantic Web.

With respect to (i) – Ontology Authoring – formal data representation can be a significant deterrent for non-expert users or small organisations seeking to create ontologies and subsequently benefit from adopting semantic technologies. Existing ontology authoring tools, such as Protégé¹[Knublauch et al., 2004] and Swoop [Kalyanpur et al., 2006] attempt to resolve this, but they often require ontology engineering skills on the part of the user. On the other hand, this is especially problematic for domain specialists, such as clinicians, business analysts, legal experts, etc. Such professionals cannot be expected to train themselves to comprehend Semantic Web formalisms and on the other hand, the process of knowledge gathering, involving both a domain expert and an ontology engineer, can be time-consuming and costly. This challenges researchers to develop user-friendly means for ontology authoring.

With regard to (ii) – manual semantic annotation is a complex and laboured task. It is both time-consuming and expensive often requiring specialist annotators or the subsequent training of such annotators. This invariably results in unnecessary exposure to formal ontological description. Such formal data representation can act as a significant deterrent for non-expert users or organisations seeking to annotate resources as part of their daily activity, thus allowing them to fully benefit from the adoption of Semantic Web technologies. While (semi)-automatic annotation tools attempt to remove this constriction, which is commonly known as the *knowledge acquisition bottleneck* [Hayes-Roth et al., 1983], their application often requires access to specialists who can combine Natural Language Processing(NLP), Machine Learning(ML) and Semantic Web ontology languages. Such specialists are costly and rare and furthermore the creation or acquisition of quality language resources to bootstrap (or train) such approaches may

¹<http://protege.stanford.edu/>, Retrieved 2011-05-22

require significant investment. Hence, this challenges researchers to develop more user-friendly manual annotation environments to support the knowledge acquisition process.

With respect to both of the aforementioned research problems, Controlled Natural Languages (CNLs) for knowledge creation and management offer an attractive alternative for non expert users wishing to (i) develop small to medium sized ontologies or a first draft ontology which can subsequently be post-edited by the ontology engineer and (ii) to annotate, while simultaneously authoring her documents in a user-friendly manner, yet at the same time shielding her from the underlying complex knowledge representation formalisms of ontology languages. Controlled Natural Languages are defined as “subsets of natural language whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity” [Schwitter and Tilbrook, 2004]. The use of CNLs for ontology authoring and population is by no means a new concept and it has already evolved into quite an active research area [Smart, 2008]. Furthermore, a natural overlap exists between tools used for both ontology creation and semantic annotation. Despite such efforts, very little research has focused on applying CNLs to Semantic Annotation.

It is important to note that there is difference between the process of ontology creation and population and that of semantic annotation. We describe Semantic Annotation as “a process as well as the outcome of the process”. Hence it describes (i) “the process of adding semantic data or metadata to content given an agreed ontology and (ii) it describes the semantic data or metadata itself as a result of this process” [Handschuh, 2005]. Of particular importance here is the notion of the *addition or association* of semantic metadata to *content*. From a strict viewpoint, ontology population from content involves modifying the ontology while annotation involves modifying the content. However, ontology population may occur as a side effect of annotation. Conversely, annotation may serve to track the origin or provenance of a newly created concept in an ontology, especially if the concept has been extracted from web content.

1.1.1 Annotation as Authoring

As with any annotation environment, a major drawback is that in order to create meta-data about a document, the author must *first* create the content and *second* annotate the content, in an additional *a posteriori*, annotation step. In the context of our application of CNL to Semantic Annotation, we seek to merge the authoring and annotation steps into one.

1.1.2 Defining Controlled Language for Semantic Annotation

We refer to the application of CNL to Semantic Annotation as Controlled (Natural Language) Annotation which reflects how traditional CNL intersects but also differs from our approach, whereby:

Controlled Annotation is the application of CNL technologies to the process of semantic annotation. Controlled Annotation aims to reduce or eliminate ambiguity with respect to the semi-automatic/manual semantic annotation of textual resources. It may include the creation of semantic data or metadata from machine processable content as in traditional CNL or apply CNL techniques that act as an interface to associate semantic data or metadata with free or uncontrolled text. Unlike traditional CNL, content in Controlled Annotation can be independent of the process.

1.1.3 CLOnE - Controlled Language for Ontology Editing

Much of the work described in this thesis is based on CLOnE - *Controlled Language for Ontology Editing* [Funk et al., 2007], which allows naive users to design, create, and manage information spaces without knowledge of complicated standards (such as XML², RDF³ and OWL⁴) or ontology engineering tools. CLOnE's components are based on

²"XML 1.0 Specification", W3.org. See <http://www.w3.org/TR/REC-xml>. Retrieved 2010-08-22

³See <http://www.w3.org/TR/PR-rdf-syntax/> Retrieved 2011-05-22

⁴See <http://www.w3.org/TR/owl2-overview/>, OWL 2 Web Ontology Language Document Overview W3C Recommendation 27 October 2009, retrieved 2011-05-08

GATE's⁵ existing tools for Information Extraction(IE) and Natural Language Processing (NLP) [Cunningham et al., 2002].

The CLOnE system was evaluated using a *task-based* methodology in comparison with a standard ontology editor – Protégé. CLOnE performed favourably with test users in comparison to Protégé. Despite the benefits of applying controlled language technology to ontology engineering, a frequent criticism against its adoption, is the learning curve associated with following the correct syntactic structures and/or terminology in order to use the controlled language properly. Adhering to a controlled language can be, for some non expert users, time consuming and frustrating. These difficulties are related to the *habitability* problem, whereby users do not really know what commands they can or cannot specify to the Natural Language Interface (NLI) [Thompson et al., 2005]. Where the CLOnE system uses natural language analysis to unambiguously parse CLOnE in order to create and populate an ontology, the reverse of this process, Natural Language Generation (NLG), involves the generation of the CLOnE language from an existing ontology. The text generator and CLOnE authoring processes combine to form a RoundTrip Ontology Authoring(ROA) environment: a user can start with an existing imported ontology or one originally produced using CLOnE, (re)produce the controlled language using the text generator, modify or edit the text as required and subsequently parse the text back into the ontology using the CLOnE environment. The process can be repeated as necessary until the required result is obtained.

1.2 Research Questions

The key research questions, arising from the challenges discussed above are as follows:

Regarding Ontology Authoring:

- **OA1:** Is a ROA environment more user friendly than a standard ontology editor for basic ontology editing tasks?

⁵General Architecture for Text Engineering - see <http://gate.ac.uk/>, Retrieved 2011-05-22

- **OA2:** Do users spend less time completing ontology editing tasks in ROA compared to a standard ontology editor?
- **OA3:** Is ROA more user friendly than a standalone CNL - CLOnE?

Regarding Semantic Annotation:

- **SA1:** Can Controlled Annotation effectively substitute for a standard manual semantic annotation tool in certain scenarios?
- **SA2:** Is Controlled Annotation more user friendly than a standard manual semantic annotation tool in certain scenarios?

The aforementioned research questions result in the following scientific and technological contributions which are described briefly in the next section.

1.3 Contributions

With respect to CNL for ontology authoring, this thesis makes the following contributions:

- A Round-Trip Ontology Authoring (ROA) environment, which combines both CLOnE and NLG, to improve the user experience with respect to basic ontology authoring and editing tasks (**OA1**).
- Empirical evidence that ROA is more user friendly than a standard ontology editor for basic ontology editing tasks (**OA2**).
- Empirical evidence that users complete basic ontology editing tasks faster using ROA compared to a standard ontology editor (**OA2**).
- Evidence that ROA is more user friendly than CLOnE for basic ontology editing tasks (**OA3**).

With respect to Semantic Annotation, this thesis makes the following contributions:

- A novel approach to manual annotation, using CNL, called Controlled Annotation, where we move away from traditional a-posteriori annotation by merging both authoring an annotations steps together(**SA1**).
- Two Controlled Language ANNotator - CLANN prototypes, each varying in expressiveness and usability(**SA1**) .
- Statistical evidence that for certain scenarios, controlled annotation can be more user friendly than a standard manual semantic annotation approach(**SA2**) .

1.4 Thesis Structure

Each chapter is preceded by a a brief introduction which connects its content to the the overall structure and research goals of the thesis. The remainder of this thesis is divided into three parts: Foundations(II), Core Research (III) and Conclusions and Future Work(IV). Each part is organised as follows:

Part II – Foundations

- Chapter 2 provides an overview of the fields of Human Language Technology (HLT) and the Semantic Web. It does not provide an exhaustive review of both fields, rather we presume that the reader has some knowledge of the Semantic Web or is expected to conduct additional reading if necessary. In addition, this chapter does not contain an exhaustive review of HLT, but rather focuses on the key technologies underpinning our research contributions which are Information Extraction(IE) and Natural Language Generation(NLG) and more specifically their intersection with the Semantic Web.
- Chapter 3 provides a thorough introduction into Controlled Natural Languages (CNLs) for the Semantic Web. In addition, we review the state of the art with respect to Semantic Annotation, ranging from manual to semi-automatic annotation tools to fully-automatic semantic annotation platforms.

Part III – Core Research

- Chapter 4 introduces the CLOnE language and discusses the, grammar, design and implementation of the language and the CLOnE environment. The chapter also discusses a comparative user evaluation with Protégé and discusses the quantitative findings. CLOnE is not a contribution of this thesis, however it does form the basis of the research output of this work.
- Chapter 5 discusses the design and implementation of the Round Trip Ontology Authoring (ROA) environment focusing on the NLG component - the ROA text generator. Again this chapter also describes a comparative user evaluation with Protégé and discusses the quantitative findings.
- Chapter 6 provides a detailed definition of *Controlled Annotation*. We describe implementations for two prototypes of Controlled Language ANNotator – CLANN, both varying in expressiveness. This chapter also describes the domain ontology, based on the domain use case which involves the authoring and annotation of project minutes and status reports. Finally, the chapter describes a user evaluation which compares both CLANN prototypes with a standard manual semantic annotation tool - Ontomat⁶ [Handschuh et al., 2002].

Part IV – Conclusions and Future Work.

- Chapter 7 summarises the contributions of this thesis and also links the results back to the original research questions described in Chapter 1. We discuss the outcome and any lessons learned and what solutions can be woven into future work.

⁶<http://annotation.semanticweb.org/ontomat/index.html>, retrieved 2011-05-08

Part II

Foundations

2 Human Language Technology and the Semantic Web

This Chapter provides an overview of the underlying semantic and language technologies of this thesis. Section 2.1 summaries the Web in brief, and discusses the Semantic Web and Linked Data to the degree that is relevant to the research output of this thesis. With respect to Human Language Technology(HLT), a definition of the term is provided and a history of its origins in Section 2.2, but we focus specifically on Information Extraction (IE) and Natural Language Generation(NLG), both of which serve as enabling technologies of our work with languages. Finally, this chapter, discusses HLT with respect to the Semantic Web focusing on Ontology Based IE (OBIE) (See Section 2.4) and Natural Language Generation from Ontologies (See Section 2.6). Finally Section 2.7 concludes the entire chapter, and provides a summary analysis with respect HLT and the Semantic Web.

References: Section 2.2 is in part based on [Bontcheva et al., 2008] and Section 2.5 is in part based on [Bontcheva and Davis, 2008].

2.1 The Semantic Web and Linked Data

2.1.1 History of the Web in Brief

The World Wide Web or called simply “WWW” is a global information system of documents interlinked via computers which are connected to the Internet. Traditionally, documents consist of hypertext [Nelson, 1965]. Hypertext functionality is realised by means of HTML – HyperText Markup Language, which can consist of formatted natural language text, digital images and media as well as structured information. It instructs

client web browsers how to display the content of a HTML document correctly. A HTML document may reference another Web document via a *hyperlink*, which when clicked on by a mouse or other input device, leads the user to the target document on the Web. Hyperlinking is achieved by embedding a machine understandable reference or URL - Uniform Resource Locator into the body of HTML markup. A URL is a global unique address or location on the Web. Hypertext forms the basis of the WWW and its discovery was intricately intertwined with simultaneous endeavours which occurred in the 1960's such as the work of Doug Engelbart [Engelbart, 1962, Engelbart and English, 1968] and Nelson's Xanadu [Nelson, 1965], and IBM's Generalised Markup Language [Goldfarb, 1996]. Early inspiration stems from Jorge Luis Borges: 1941 short story *The Garden of Forking Paths*⁷ as well as Vannevar Bush's article on the proto-hypertext device – Memex, a microfiche that halted upon request and was analogous to the associative processes of the human mind but with permanent memory [Bush and Wang, 1945].

In 1980, Tim Berners Lee, an independent contractor at the European Organisation for Nuclear Research – CERN developed ENQUIRE, which was a personal database of people and software models that explored the potential of hypertext. Each new page in the ENQUIRE system linked to an existing page. ENQUIRE was limited however to a single machine and was developed on a NEXT workstation⁸. Upon his return to CERN in 1984, Berners Lee began to work on the information presentation issues of ENQUIRE and more importantly he began to factor in the need for physicists to share data globally despite the barriers of not having common OS platforms or presentation software [Berners-Lee, 1993].

By Christmas 1990, Berners Lee had engineered all the basic tools fundamental to the Web we know today such as: the HyperText Transfer Protocol (HTTP) and HTML, the first Web browser and editor, the first HTTP server software, then known as the

⁷"The Garden of Forking Paths" (original Spanish title: "El Jardín de senderos que se bifurcan".) is a 1941 short story by Argentine writer and poet Jorge Luis Borges.

⁸The NeXT station was a high-end workstation computer developed, manufactured and sold by the company NeXT from 1990 until 1993. It ran the NeXTSTEP operating system.

CERN httpd, the first web server⁹, and the first web pages which described his project [Berners-Lee, 1993]. In order to promote the project, the CERN telephone directory was published on the Web. Prior to this CERN users were required to access a mainframe to check the telephone directory. In September 1991, Paul Kunz from the Stanford Linear Accelerator Centre (SLAC) visited CERN and was impressed by the technology. He brought the NEXT software back to SLAC, where it was adapted by librarian Louise Addis to the VMC/VAC operating system on the IBM mainframe in order to display SLAC's catalog of online documents. By the mid nineties, early adopters of the Web were university based. However, there were still no graphical browser available aside from what was available for the NEXT platform. A major milestone for the WWW was the development of the Mosaic Web Browser in 1993 by a small team at the National Centre for Supercomputing Applications (NCSA), University of Illinois at Urbana-Champaign (UIUC), which was led by Marc Andreessen [Berners-Lee, 1993].

By May 1994, the first WWW conference was organised by Robert Cailliau. It was held at CERN. Since then the conference has been held annually. In 1994, Tim Berners-Lee founded the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology with support from the Defence Advanced Research Projects Agency (DARPA) and the European Commission. It comprised of academic and commercial organisations that were willing to create standards and recommendations to improve the quality of the Web.

The period 1996-1998, saw the emergence of Web based businesses or E-commerce and the increase of start-up companies due to the novelty of the the dot-com concept. After the dot-com boom and subsequent bust from 1999-2001, several companies managed to find success by developing more compelling business models. Examples included Google's search engine, Amazon's online department store experiences, Ebay's do-it-yourself auction site and airline booking sites. In addition, this new era included the emergence of social networking sites such as MySpace and Facebook, which, although initially unpopular, have now become part of everyday culture. New methods for sharing and exchanging

⁹<http://info.cern.ch>

content in an ad hoc manner such as Weblogs and RSS began to rapidly gain acceptance on the Web. This new model for information exchange which enabled users to edit and generate website content themselves has collectively grown into what we now call Web 2.0. This has ushered in a new period of democratisation and popularisation of the Web. New sites such as Wikipedia were revolutionary in exploiting user generated content. In addition, Youtube, the video viewing website, became the most quickly popularised website in history. Beyond Web 2.0, it is believed that the Semantic Web will be the next stage of evolution for the Web.

2.1.2 The Semantic Web

The goal of the Semantic Web is to augment webpages with machine understandable meaning or *semantics* [Berners-Lee et al., 2001b]. This would involve lifting current Web content, which can be both unstructured and semi-structured, into a *Web of Data*. The Semantic Web is built on top of the the existing Web infrastructure and standards and the process of creating the Semantic Web involves the addition of several technologies which are summarised as the “The Semantic Web layer cake” (see Figure 2.1¹⁰). Exploring the layer cake in detail is beyond the scope of this thesis, as the CNL technologies developed as part of this work were targeted towards the creation of metadata and ontologies. Hence we focus on describing the fundamental data interchange language for the Semantic Web - RDF and its schema. In addition, we provide a brief overview of the ontology language OWL (See Section C.1.1).

The Resource Description Framework - RDF defines a standard method for modelling information on the Web. In 1999, the W3C published the first RDF recommendation [Lassila and Swick, 1999]. The latest recommendation was published in 2004 [Klyne and Carroll, 2004]. There are overall six published RDF recommendations¹¹.

RDF is based on the open world assumption (OWA). Hence, unlike closed world as-

¹⁰See <http://www.w3.org/2007/03/layerCake.svg>, Retrieved 31 Oct 2012

¹¹<http://www.w3.org/standards/techs/rdf>, Retrieved 2011-05-08

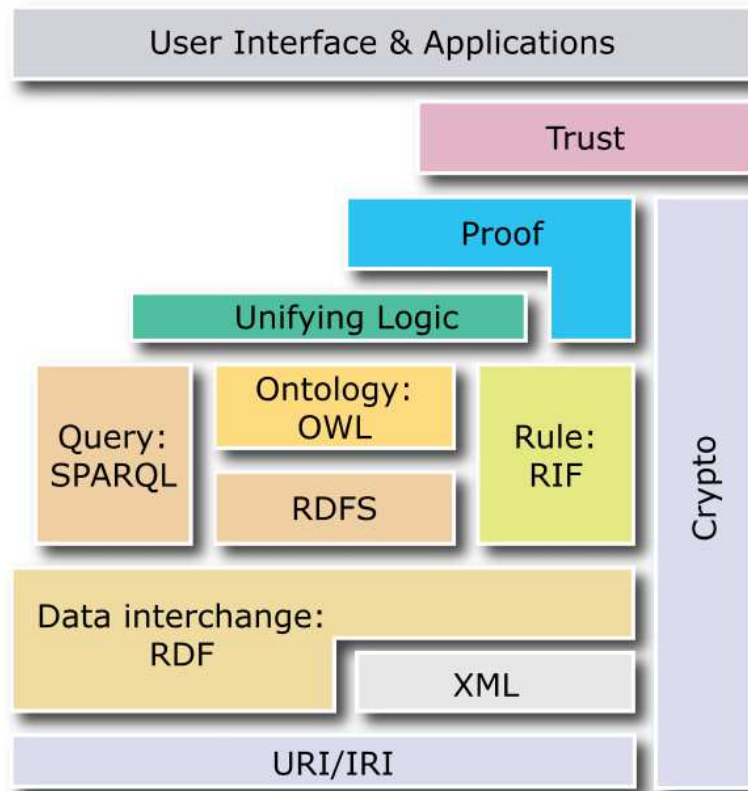


Figure 2.1: The Semantic Web layer cake.

sumption (CWA), if a statement is unknown, it is not assumed to be false. In addition, there is no unique name assumption (UNA), so different names, presented by a URI¹² may map to the same entity.

RDF permits the description of resources, which are things that can be identified on the Web even if they cannot be directly retrieved on the Web [Klyne and Carroll, 2004]. Statements are made about resources in the form of *subject, predicate, object* expressions which are commonly known as *triples*. A subject is the resource, while the predicate or property describes a relationship between the subject and the object. An example of a statement is “The sea has the colour blue”. Hence “the sea” is the subject, while “has the

¹²Uniform Resource Identifiers (URIs, aka URLs) are short strings that identify resources in the web: documents, images, downloadable files, services, electronic mailboxes, and other resources.

colour” is the predicate and finally ”blue” is the object. The object of a statement (i.e., the property value) can be another resource or it can be a literal; i.e a resource (specified by a URI) or a simple string or other primitive datatype defined by XML.

Resources are represented by URIs. Anything described by an RDF expression is a resource. A resource can refer to anything with an identity, be it an online resource such as a Webpage or a ”real world” resource such as a person or place. A resource can be identified in RDF using URIs, which can appear in any position of a triple. A property is a specific attribute, or relation used to describe a resource. A property has a specific meaning and defines constraints on permitted values as well the types of resources it can describe. A property may also describe its relationship to other properties. A literal may only appear in the object position of a triple, as a value of a property. Literals can have type information using XML schema or a language tag. Unidentified resources called blank or anonymous nodes are unnamed nodes within an RDF graph. They cannot appear as the predicate in a triple. They can be used to represent complex data, but their usage is discouraged as it poses problems when querying [Bizer et al., 2007]. A collection of RDF statements can be represented as a labeled multi-graph.

Furthermore, additional knowledge about an RDF statement can be modelled using *reification*, whereby a statement is assigned a URI and treated as a resource. Consequently additional statements can be modelled such as “Brian says that Pradeep is a Masters Student” in order to make additional assertions about a statement. Reification can have many uses, such as annotation, adding provenance data, modelling context or access rights. An alternative to using reification is named graphs, whereby a collection of RDF statements or a graph is associated with a URI in order to add context without the need to reify the triple. This is becoming much more popular as the application of reification introduces indirectness and is quite verbose and cumbersome to model.

Serialisation of RDF may be in XML form [Beckett, 2004] or in non XML form called Turtle [Berners-Lee, 2006b]. Other human friendly formats include N-Triples [Grant and Beckett, 2004], a subset of Turtle [Beckett, 2007], which is quite popular

among semantic web developers. RDFa is also another syntax format which permits the addition of a set of attribute-level extensions to HTML, XHTML and various XML-based document types for embedding rich metadata within Web documents [Adida et al., 2008]. An example of Turtle can be seen in Listing 2.1 as well as the equivalent in RDF XML in Listing 2.2. Both examples describe the following statements *Leonardo Di Caprio starred in Inception* and *Inception has a similar plot to the Matrix*.

Finally, we note that the standard recommended query language for RDF is SPARQL [Prud'hommeaux and Seaborne, 2008], although other languages exist such as RDF Data Query Language (RDQL) [Seaborne, 2004]. Others include the Sesame RDF Query Language (SeRQL) [Broekstra and Kampman, 2003] and RDF Query Language (RQL) [Karvounarakis et al., 2002]. An in-depth discussion of RDF query languages is beyond the scope of this thesis, as we are not concerned with mapping natural language for querying RDF data.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://www.example.org/> .

ex:leonardo_de_caprio_ ex:starred_in ex:inception .
ex:inception rdf:type ex:movie .
ex:inception ex:similar_plot_as ex:the_matrix .
```

Listing 2.1: RDF example represented in Turtle.

```

<?xml version="1.0" encoding="UTF-8"?>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ex="http://www.example.org/">

  <rdf:Description rdf:about="http://www.example.org/leonardo_di_caprio">
    <ex:starred_in >
      <ex:movie rdf:about="http://www.example.org/inception" />
    </ex:starred_in >
  </rdf:Description >

  <rdf:Description rdf:about="http://www.example.org/inception">
    <ex:similar_plot_as rdf:resource="http://www.example.org/the_matrix" />
  </rdf:Description >

</rdf:RDF>

```

Listing 2.2: RDF example represented in XML.

2.1.2.1 Ontologies, RDF Schema and OWL

RDF is a means for interchanging information in an homogenous manner, however additional semantics are required in order to make this information machine understandable. For instance, in order to create and associate metadata to resources, it is necessary to first develop a vocabulary on which to base the metadata. This basic vocabulary consists of terms and concepts of interest to the application domain as well as relations across concepts. This representational vocabulary can be defined as an ontology, which is a shared representation for a domain of discourse and may include definitions of classes, relations, functions and other objects [Kashyap et al., 2008b]. Ontologies are represented as OWL and RDFS within the layer cake in Figure 2.1. Studer et al define an ontology “as a formal explicit specification of a shared conceptualisation” [Gruber, 1993, Studer et al., 1998]. It is formal in that it must be machine readable, explicit in that the types of concepts used and their respective constraints must be explicitly defined. An ontology may also be

shared in that it is not private but must be accepted by some group [Studer et al., 1998].

Ontologies can be categorised into the following [Gómez-Pérez et al., 2007]:

- **Top level ontologies or Upper level ontologies**, which describe general notions and should be linked to from existing ontologies [Gómez-Pérez et al., 2007].
- **Domain ontologies**, which are reusable for a given specific domain such as law, medicine and aeronautics [Mizoguchi et al., 1995, van Heijst et al., 1997]
- **Task ontologies**, which describe the vocabulary needed for a generic task or activity such as scheduling, selling and diagnosing. Task ontologies specialise the terms in top level ontologies [Mizoguchi et al., 1995, Guarino, 1998]. The tasks defined need not belong to the same domain.
- **Domain task ontologies** define tasks which are reusable in a given domain but not across domains. They are application independent. An example would be a trip schedule [Gómez-Pérez et al., 2007].
- **Method ontologies** define the relevant concepts and relations needed to perform a reasoning process in order to execute a specific task [Tijerino and Mizoguchi, 1993].
- **Application ontologies** are application dependent and model all the knowledge necessary for a given application. They may often extend or specialise domain or task ontologies for a given application [van Heijst et al., 1997].

Lightweight Ontologies (or schemata) can be described using the RDFs standard. More expressive ontologies can additionally use the OWL standard. With respect to RDF schema or RDF(S), its purpose was to extend upon the semantic expressiveness of RDF. While RDF provides a way to express simple statements about resources, it does not provide the ability to define specific classes of resources and constraints between classes and properties. RDFS became a W3C recommendation in early 2004 [Brickley and Guha, 2004]. It essentially extends RDF with abilities to specific well defined relationships between classes. RDFS has properties such as: `rdfs:subClassOf`,

`rdfs:subPropertyOf`, `rdfs:domain` and `rdfs:range`. RDFS also defines other built in vocabularies, initially specified in the RDF model and syntax specification. RDF features can be summarised as follows [Antoniou et al., 2005]:

- classes and their features,
- binary properties between objects,
- organisation of classes and properties into hierarchies,
- types for properties such as domain and range restrictions.

While RDFS allows for the representation of some ontological knowledge, it is limited to typed hierarchies, subclass and sub-property relationships, domain and range restrictions, classes and instances. There are a number of other features, however which are missing such as: local scope for properties, disjointness of classes, cardinality restrictions, and special characteristics for relations such as transitivity, symmetry, uniqueness and inverses.

The Web Ontology Language(OWL) extends upon RDFS in this respect by adding more expressive semantics to cover the above limitations. Work began initially in 2001 on new ontological language to extend RDFS, with the result OWL, becoming a W3C recommendation in 2004 [Bechhofer et al., 2004]. OWL provides the ability to express axioms and definitions and extends upon RDFS with the aforementioned features, as well as for example, equivalence between instances and classes and properties as well as inequality between individuals.

OWL is structured into three layers of expressivity [Antoniou et al., 2005]:

1. OWL Lite, which permits the expression of definitions of axioms as well as some of the features mentioned above.
2. OWL DL which provide for support for users who want maximum expressiveness while retaining good computational properties, in particular, decidability.

3. OWL Full, which provides full expressiveness but with no computational guarantees.

In 2009, OWL 2 became a W3C recommendation and introduced new functionality beyond OWL 1.0 such as keys, property chains, richer data types and data ranges, qualified cardinality restrictions and asymmetric, reflexive and disjoint properties to name but a few [W3C OWL Working Group, 2009]. Discussing the profile and rationale of OWL 2 is beyond the scope of thesis, as the ontologies described in later chapters are either equivalent to RDFS or no more than OWL Lite in expressiveness.

Numerous vocabularies have emerged as a result of the Semantic Web initiative, of which a limited number have gained widespread adoption and recognition including: Dublin Core Metadata Initiative (DCMI), which is used to describe “core metadata for simple and generic resource descriptions” [Nilsson et al., 2008], Simple Knowledge Organisation System (SKOS) [Miles and Bechhofer, 2009], which is used to share linked data about thesauri, taxonomies, classification schemes and subject heading systems, Friend of A Friend (FOAF) [Brickley and Miller, 2005] to describe people and organisations, Semantically-Interlinked Online Communities (SIOC) [Breslin et al., 2005] to describe online communities and Description of A Project (DOAP) for describing software projects¹³.

2.1.2.2 Ontology Editing Tools

Ontology editors are applications designed to assist domain experts and ontology engineers to construct and edit ontologies. The ontologies developed by such tools can be represented in a variety of ontology languages and often offer functionality to export the the developed ontology into different ontology language file formats. We provide only some brief examples below, as a comprehensive survey of ontology authoring tools¹⁴ is beyond the scope of this thesis. For the inquisitive reader, however, we recommend

¹³See <https://github.com/edumbill/doap/wiki>, Accessed 31 Oct 2012

¹⁴http://www.w3.org/wiki/Ontology_editors, Accessed Wed 24 Jul 2013 15:08:06 IST

the reader to the following literature [Gómez-Pérez et al., 2007], [Corcho et al., 2003], [Norta et al., 2010], [Cardoso and Escórcio, 2007] and [Buraga et al., 2006]. One example is the **NeON**¹⁵ toolkit [Haase et al., 2008], which is a state of the art, open-source, multi platform ontology engineering environment, which aims to provide comprehensive support for the entire ontology engineering life cycle. The toolkit is based on the Eclipse platform and provides a set of plugins to support all aspects of ontology engineering. Another well know tool, is **Protégé**¹⁶[Knublauch et al., 2004], which is a free open source ontology editor that permits collaborative ontology engineering. It supports numerous file formats for exporting ontologies including RDF(S), OWL and XML schema. With respect to **TopBraid Composer**¹⁷, it is an example of an enterprise class modelling environment for building Semantic Web ontologies and applications. The tool is W3C compliant and offers comprehensive support for developing, managing and testing knowledge models as well as the subsequently populated knowledge bases. TopBraid also provides an API for developing semantic client/server and browser based solutions. Another example is the **Hozo** ontology development environment, which consists of Ontology Editor, Onto-Studio and Ontology Server [Kou, 2002]. Ontology Editor provides an interface for the user, in order to browse and modify the ontology as well as management of properties within the concept hierarchy. Onto-Studio on the other hand assists users to design ontologies from technical documents, while finally Ontology Server manages all builds of ontologies and models.

2.1.3 Linked Data

The term Linked Data refers to a set of best practices for publishing and linking data on the Web. These practices have been widely adopted in recent years, ultimately leading to a global data space of billions of assertions. Linked Data is about creating typed links

¹⁵<http://neon-toolkit.org/> Accessed Wed 24 Jul 2013

¹⁶<http://protege.stanford.edu/>, Accessed Wed 24 Jul 2013

¹⁷http://www.topquadrant.com/products/TB_Composer.html, Accessed Wed 24 Jul 2013 15:48:21 IST

between different data resources via the Web. This may consist of linking heterogeneous data resources in a single organisation together or linking data sources between organisations which are separated geographically. Linked Data is machine readable data which is published on the Web. The meaning of the data is explicitly defined and links to external data sources are provided, which may in turn link to other external data sources and so on. While the Hypertext Web is based on HTML, which connects documents via untyped links, Linked Data is dependent on documents containing data in RDF format.

As opposed to merely connecting documents with each other, Linked Data proposes to exploit RDF in order to make typed statements about the world, resulting in the *Web of Data*. For the novice reader, it may be unclear as to how to distinguish the Semantic Web from Linked Data. The Semantic Web conforms to the notion of a Web of machine processable data or the Web of Data. This is still the end goal, however the publishing practices of Linked Data are considered as the means or *the process* for reaching this goal [Bizer et al., 2009].

Linked Data is based on a set of rules or principles outlined by Sir Tim Berners Lee in 2006 [Berners-Lee, 2006a]. The Linked Data Principles have become a recipe for publishing and connecting data using the infrastructure of the Web while also respecting its architecture and standards. The principles are as follows:

1. use URIs as names of things.
2. use HTTP URIs so that people can look up those names.
3. provide useful information for when someone looks up a URI, using standards such as RDF and SPARQL.
4. include links to other URIs, so that clients can discover more things.

With respect to publishing, it involves three basic steps:

1. Assign URIs to the entities described by the data set and provide for dereferencing of given URIs over the HTTP protocol into RDF representations. In other words,

the URI should be able to be dereferenced in order to get the information about the thing it represents in RDF and not a web page; the URI should act as global identifier for a thing in the world.

2. Point RDF links to other data sources on the Web. This permits clients to navigate the Web of Data as a whole by following links.
3. Provide metadata about published data in order for clients to assess the quality of the published data as well as to choose between different forms of access.

The most prominent examples of the adoption and application of Linked Data is the Linked Open Data Project. It is a grass roots community effort founded in 2007 supported by the W3C Semantic Web Education and Outreach Group. The ongoing aim of this community is to bootstrap the Web of Data by lifting existing datasets into RDF according to Linked Data principles and publishing them on the Web. Participants were initially community researchers and developers, but this community has grown to include significant input from large organisations such as the BBC, Thompson Reuters and the Library of Congress. The growth has been precipitated by the open nature of the project whereby anyone can participate so long as they adhere to the principles of Linked Data. Content ranges from diverse sources such as data about locations, companies, books, persons, television, media, music, scientific publications, health and life sciences, government and much more (See Figure 2.2).

This section is not an exhaustive review of HLT for the Semantic Web rather it seeks to provide the foundations of enabling language technologies for enriching the Semantic Web. Section 2.2.1 provides an overview of HLT origins vis a vis Language Engineering(LE). We are particularly concerned with LE as it successfully converted scientific output from both Computational Linguistics(CL) and Natural Language Processing(NLP) into real world applications that are robust and scalable. Information Extraction(IE) and applied Natural Language Generation(NLG) are examples of NLP research which have evolved into HLT/LE products and disciplines. Hence, Section 2.3 defines IE and the subtasks involved as well as metrics for measuring the performance of language analysers in IE systems. In particular, we are concerned with finite-state parsing tools used in IE as they are a core enabling technology for our controlled language tools and the research contribution of this thesis. In Section 2.4 we discuss briefly Ontology Based IE (OBIE), highlighting the role traditional IE plays in enriching ontologies, and likewise how ontologies can inform the IE process. Furthermore we discuss emerging performance evaluation techniques for OBIE systems. Finally, in Sections 2.5 and 2.6, we provide an overview of NLG and briefly review emerging research with respect to NLG for ontologies and offer conclusions in Section 2.7.

2.2.1 History and Definitions

Firstly, we shall begin with a concrete definition of Language Technology also referred to as Human Language Technology (HLT) which “comprises of computational methods, computer programs and electronic devices that are specialised for analysing, producing or modifying texts and speech. These systems must be based on some knowledge of human language. Therefore language technology defines the engineering branch of computational linguistics”¹⁹. HLT is a broad term for technologies that enable communication between humans and computers. Furthermore, it is the term used for tools that mediate between the human user and the machine. The technologies use knowledge about spoken and

¹⁹<http://www.dfki.de/lt/lt-general.php>, Extracted Mon Aug 18 12:28:42 IST 2008

written language and about developing computer software to recognise, analyse, interpret and generate language²⁰.

Historically, the term HLT owes its origins to the Telematics Applications Programme, specifically the *Language Engineering* Project Directory supported by the European Commission²¹. This leads us to the term Language Engineering (LE), which is defined as “the discipline or act of engineering software systems that perform tasks involving processing human language. Both the construction process and its outputs are measurable and predictable. The literature of the field relates to both application of relevant scientific results and a body of practice”. Essentially LE is concerned with scalable and robust language processing applications, algorithmic and data/language resource reuse, performance measurement of language processing algorithms and applications and efficacy. Furthermore, LE does not concern itself with paradigm conflicts, such as empiricism vs rationalism or linguistic competence vs linguistic performance models, which continue to simmer within the NLP/CL communities. LE will apply or combine the most effective theoretical model(s) for a given task, as it is concerned with measurable and predictable engineering costs and outputs when constructing a language processing application. While the origin of the term LE can be traced back to COLING -88 [Cunningham, 1999], it was not until the early 1990’s that the term become common across Europe, as a result of the funding programme mentioned earlier, whereby the EC reported in 1996 that “Language Engineering (LE) has successfully promoted a shift from long-term research activities to more immediately feasible and industrially relevant research technology and development themes by supporting projects aiming at market opportunities in the short term and the medium term”. In the late 1990s the EC changed the title of the programme to Human Language Technologies, a term which we can regard as roughly synonymous with LE [Cunningham, 1999].

²⁰<http://www.itri.brighton.ac.uk/projects/euomap/explained.html> Extracted Mon Aug 18 12:30:47 IST 2008

²¹European Commission Directorate General XIII, Luxembourg, 1996, See http://ec.europa.eu/dgs/education_culture/valorisation/biblio-others_en.htm

2.3 Information Extraction

Information Extraction (IE) is a technology which analyses natural language texts in order to extract useful snippets of information or facts. It may take written or spoken text as input, and output fixed form, unambiguous structured data. The output can be stored in a database, spreadsheets or displayed directly to the user [Cunningham, 2005].

A recent endeavour has been the linking of IE output to ontologies and their subsequent population in order to produce a knowledge base. Ontology population is one outcome of the recent application of IE to the Semantic Web called Ontology Based Information Extraction (OBIE). The association and linking of semantic metadata to snippets of text identified by IE is called Semantic Annotation. OBIE will be described later in Section 2.4 while Semantic Annotation will be reviewed in full in Chapter 3.

IE output can also be used as a preprocessing step within an Information Retrieval (IR) engine such as Google i.e. at the indexing stage. Alternatively, the semantically typed output or annotations within an IE system can be indexed themselves thus enabling a form of Semantic Search. The system searches over semantic annotations of text snippets ; over types rather than over a bag of words. One should not confuse IR with IE. Classic IR finds relevant texts for a given query, usually in the form of keywords and returns them to the user, while IE analyses the text and presents only specific information, based on predefined user specification or schema. So while IR returns documents for a given domain, which would require subsequent manual analysis in order to create a database or spreadsheet, IE can populate a record or spreadsheet automatically. IE systems are more knowledge intensive to engineer than IR systems, and their efficacy is dependent on the extraction task, the target domain or scenario. However, as the volume of raw text continues to grow on the Web and within organisations, IE has the potential to dramatically reduce the cognitive overload and time required to read through texts retrieved via IR. In addition, the unambiguous output produced by IE has multilingual implications whereby the structured snippets outputted by an IE systems are easier to automatically translate as opposed to engaging in broad coverage machine translation of

documents returned via IR [Cunningham, 2005].

Historically, IE grew out of work in the late 1980s and 1990s in the Message Understanding Conferences(MUC) [?]. It dates back to early language computation task in the 1960's collectively know as *fact extraction*. MUC was unusual at the time of its conception in that it employed strict quantitative evaluation and injected empiricism and proper evaluation procedures into NLP research. Although, one could presume that this should have been common practice as with other sciences, NLP was largely influenced then by rationalism and linguistic theories and less so by empiricism or engineering practice.

The MUC competitions accelerated progress in IE. This has resulted in the production of present day commercial tools such as Temis²², IBM Content Analytics²³ and most recently an association to the Semantic Web and Linked Data fields with semantic annotation platforms and services such as KIM²⁴ and OpenCalais²⁵. Over the last 20 years IE has matured with work on adaptive IE, applications within the contexts of business, health care and marketing. This is also a renewed interest in opinion orientated IE with the advent of social media [Cunningham, 2005]. This has culminated with the recent high publicity of the IBM Watson Deep Question Answering project²⁶ and the emergence of *Big Data Analytics* (as the new hot topic). The beginning of the 21st century has seen IE re-branded, re-marketed or subsumed under new buzz words such as *Text Analytics*, *Content Analytics*, *Next Generation Analytics* with the most recent being *Deep Analytics*! Perhaps it is more so the case that IE has matured as a field both with respect to tools and methodologies over the last 20 years and it is now poised to deliver useful and cost-effective applications. Moreover, lessons learned from efforts such as MUC and its successors in IE research and development have built up a community that can now

²²<http://www.temis.com/>, Accessed Wed 24 Jul 2013 15:48:21 IST

²³<http://www-01.ibm.com/software/ecm/content-analytics/>, Accessed Wed 24 Jul 2013 15:48:21 IST

²⁴<http://www.ontotext.com/kim>, Accessed Wed 24 Jul 2013 15:48:21 IST

²⁵<http://www.opencalais.com/>, Accessed Wed 24 Jul 2013 15:48:21 IST

²⁶<http://www.research.ibm.com/deepqa/>, Accessed Wed 24 Jul 2013 15:48:21 IST

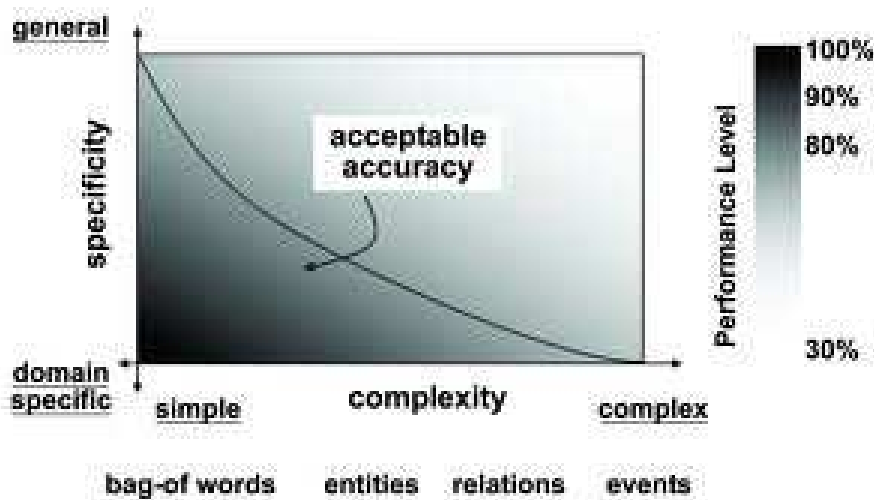


Figure 2.3: Performance trade-off relative to specificity and complexity for IE (extracted from [Cunningham, 2005], Fri 18 Jan 2013 11:24:39 GMT)

accurately predicate the cost effectiveness of the technology. This is summarised in Figure 2.3, whereby acceptable IE performance is measured as a trade off between domain specificity and task complexity. The more complex the IE task e.g. entity extraction, relation extraction or event extraction, the more specific the domain must be. Performance in IE degrades when moving towards both higher levels of complexity and specificity. Conversely, the simpler the data to be extracted, the easier algorithms can be retargeted.

In [Cunningham, 2005] the author describes how specificity is also multidimensional, where the IE task is influenced by the text and genre i.e. whether the text is edited newswire or business reports vs historical texts or user generated content or microblogging content. We move now to describe five generic tasks in IE based on [Cunningham, 2005]. They are derived from the MUC-7 competitions [?] . The definitions of each task have changed naming conventions and merged over time but the core tasks have remained consistent and consist of:

- Named Entity Recognition (NE) or (NER) - the purpose of which is find and classify names of locations, dates and organisations, amounts of money. In practice,

of course, the notion of what is a NE can be quite specific to the domain.

- Coreference resolution (CO), attempts to find identify relations of references between entities. Both NE and CO tasks were merged into a single entity detection and tracking task (EDT) in the Automatic Content Extraction program - ACE [Consortium, 2008], the successor to the MUC competitions.
- Template element construction (TE) which adds additional descriptive information to NE output based on coreference.
- Template relation construction(TR), finds relations between entities. Both TE and TR were collapsed into a relation detection and tracking task in ACE [Consortium, 2008] and are described as relation extraction (RE) within the literature.
- Scenario template production (ST), which merges TE and TR results into predefined event scenarios. It was renamed as Event Detection in ACE [Consortium, 2008]

If we take the following example fictitious text below as an example:

Eponymous Weyland corp acquires Microsoft. It was founded in 2014 by Sir Peter Weyland, industrialist, technoevangelist, who won the Nobel Prize for Science in 2024.

NE would discover that *Weyland corp* and *Microsoft* are companies and that *Sir Peter Weyland* is a person as well as the mentions of dates *2014* and *2024* and perhaps that the *Nobel Prize for Science* is an award. CO would link the pronominal *it* to its antecedent *Weyland corp*. The RE task would add information either to entity *Sir Peter Weyland* as properties such that the entity is and industrialist and a techno-evangelist in the form of *is-a* relations and perhaps even a royal title. Furthermore, it would discover that Weyland Corp had acquired Microsoft and that its owner is Peter Weyland. Finally event detection would pull the relations and entities and temporal information together based on a predefined template scenario.

2.3.1 Performance and Metrics

As mentioned earlier, the MUC conferences were succeeded by ACE and the tasks became more complex. In addition, ACE 2003 began to include hierarchical types such as entity subtypes and relation subtypes which was an important step, whereby the community has begun to recognise the need to factor the hierarchy of an ontology into IE competitions. Since then, ACE has been replaced by the Text Analysis Conference(TAC)²⁷. The original MUC/ACE tasks are still present under different categories but new tasks focus on knowledge base population. New tasks can be proposed for review and inclusion annually, which accommodates changes in the IE field. Event detection is not explicitly mentioned in TAC 2012, however the announcement of the recent of 2012 Cold Start task²⁸ has seen the inclusion of Semantic Web ontologies and query languages as well as Ontology Based IE as a task within the IE field and community.

Evaluation metrics consist of the traditional IR precision and recall metrics adapted for IE, in which precision is the number of correct answers divided by the number of answers produced (See Equation (2.1)) and recall is the number of correct answers divided by all answers in the gold standard(See Equation (2.2)).

The term gold standard or *Key* set, owes its origins to MUC. The gold standard can be a human annotated corpus for a given task or templates created manually by analysts [?]. Good practice is to conduct inter-annotator agreement with a minimum of three annotators and adjudicate the annotations into a gold standard. For more information on inter-annotator metrics, we refer the reader to [Appelt and Israel, 1999]. As Appelt reports, inter-annotator agreement can be quite revealing with respect to the task difficulty, where agreement between humans with regard to annotations can range from 60% to 80%, despite considerable manual efforts, which may include many person months. Consequently, when measuring the success of an IE task, one should factor in inter-annotator agreement scores. IE (and IR) systems tend to experience a trade off

²⁷<http://www.nist.gov/tac/>, Accessed Wed 24 Jul 2013 15:48:21 IST

²⁸http://www.nist.gov/tac/2012/KBP/task_guidelines/, Accessed Wed 24 Jul 2013 15:48:21 IST

between precision and recall. This may also be driven by needs of the IE application. More results may preferred than correct ones. so higher recall or coverage and vice versa accuracy may take precedence. So the facts returned they must be correct or have high precision even at the risk of missing possible facts. This trade-off is measured in the form of the harmonic mean of F- Measure. It is the weighted average of both precision and recall. The equation is presented below, where β is a adjustable parameter representing the relative importance of precision and recall (See Equation (2.3)) [Appelt and Israel, 1999]:

$$\text{Precision} = \frac{\text{Correct Answers}}{\text{Answers Produced}} \quad (2.1)$$

$$\text{Recall} = \frac{\text{Correct Answers}}{\text{Total Correct Answers}} \quad (2.2)$$

$$F_{\beta} = (1 + \beta^2) \cdot \frac{R \cdot P}{(\beta^2 \cdot P) + R} \quad (2.3)$$

In [Cunningham, 2005] the author argues that IE as a task has plateaued at 60% of human performance. The F-Measure varies depending on the IE task. Though this may initially come across as not very optimistic, one needs to factor in the upper bounds of inter annotator agreement. Furthermore the F-Measure varies for each IE task, with NE maintaining on average an F measure of 95% and CO at 50 -60%. The RE task can reach 75% F-Measure, while the event extraction/scenario template tasks score at 60%. We note that human inter-annotator scores for this task can be as low as 80%.

2.3.2 Approaches to developing IE systems

IE systems have been historically developed using either one of two methods: the (1)knowledge based approach or (2) the trainable/machine learning systems approach. In practice a hybrid approach is often taken. Furthermore, deciding on which approach to take depends on the IE task, available resources and time constraints. The knowledge based IE approach which is rule based/linguistic pattern based, requires basic preprocessing

and shallow natural language analysis techniques, i.e. tokenisation, sentence splitting, lemmatisation, part-of-speech(POS) tagging and will use finite state dictionary/gazetteer lookup for important domain specific trigger phrases and terms.

With respect to trainable IE systems, they can be based on statistical/probabilistic models or machine learning approaches, which are data driven. These systems learn rules/patterns from training data and interaction from users, but they require large amounts of quality training data.

As mentioned earlier, neither approach is preferred and in practice a hybrid approach is best. Furthermore, the knowledge based approach can be used to bootstrap the creation of a gold standard training set where none exists. In addition, rule based linguistic information can be converted into features for learning algorithms. For further details, with respect to choosing the best approach, we refer the reader to [Appelt and Israel, 1999]. In the work presented in this thesis, we take a rule based approach for building our CNL systems. We enforce determinism into our partial parsing tools and build a set a Controlled Language IE or CLIE tools (see Chapter 4). In the next section, we describe briefly a real world NLP framework for IE - GATE and discuss partial parsing methods using GATE's pattern matching over annotations engine - JAPE. Both GATE and JAPE play a crucial role in the development of the CNL based tools described later in this thesis.

2.3.3 GATE - Information Extraction in practice

GATE(General Architecture for Text Engineering)[Cunningham et al., 2002] is an framework for developing and deploying software components that process human language. It provides a set of NLP tools such as: a tokenizer, a gazetteer, POS(Part of Speech) taggers, chunkers and parsers - collectively called Processing Resources (PRs). Language Resources (LR)s on the other hand consist of corpora, documents, ontologies and annotation schemas. The architecture supports plugins for managing and extending PR and LRs, called CREOLE (Collection of Reusable Objects for Language Engineering). GATE also provides for a graphical user interface for developing NLP applications. It is an ex-

tremely widely used and recognised open source NLP tool for reusable experiments and a *de-facto* standard for NLP applications. GATE was chosen as the platform to carry out this research in this thesis because it is one of the few (if any) open source frameworks with ontology aware IE capabilities.

2.3.4 Partial Parsing for IE

Partial or shallow parsing is often applied to many IE tasks, in particular the NER task. IE systems do not aim for full text understanding and do not extract all possible information, rather they aim to identify and classify segments of text containing valuable information. Consequently, complete and complex parsing is not always required to perform extraction and furthermore to do so would not be cost-effective both with respect to speed and robustness of the IE system. In addition, the majority of the output produced from full parsing may ultimately be discarded. Furthermore, with respect to performance this would also be a waste of processing time.

One approach to partial parsing is applying a cascade of finite state transducers or (FST)s. An FST is a finite state machine (FSM) which maps between two sets of symbols. While a FSM defines a formal language by defining a set of strings, FST defines the relation between two sets of strings.

A major advantage is that output from an earlier transducer can be passed as output to another transducer for further use, constituting a cascade of finite state transducers. An FST can be equivalent in recognition power to a regular or Type 3 language in the Chomsky hierarchy [Chomsky, 1956]. A context free grammar or Type 2 grammar is required to recognise most fragments of English, however a cascade of FSTs can more closely approximate a context free grammar.

The intent is to produce flat syntax trees or chunks as oppose to a deep tree structure produced with deep parsing. Parse trees within FSTs are produced in a bottom up fashion and links are created between all the major constituents.

Cascaded finite state transducers form the parsing engine of the FASTUS IE system,

which was crafted for the MUC-3 competition [Onyshkevych et al., 1993]. It performed extremely well in the competition [Hobbs et al., 1996]. FASTUS was influenced by work on finite state approximations of context free grammars [Pereira and Wright, 1991]. An interesting observation was that complex verbs could be collapsed into a single category for parsing. So for example “to attack” could be paraphrased as ” to conduct an attack” or ” engage in an attack”. Rather than attempting to write rules to cater for all the examples above. Each variation was stored in a list and collapsed into a `VG-Attack` category. Hence, only one pattern was required which was subsequently exploited by later transducers within the cascade. The parsing tools used in our controlled natural language systems are similar to those used in FASTUS.

JAPE - Java Annotation Patterns Engine is the pattern matching over annotations language used in GATE. JAPE is compiled into a cascade of finite-state transducers. It provides finite state transduction over annotations based on regular expressions and is based on Common Pattern Specification Language (CPSL)[Appelt and Onyshkevych, 1998], developed by Doug Appelt, the lead developer of FASTUS, which we described earlier.

While typically regular expressions are applied to character strings, a simple linear sequence of items, in the case of JAPE matches over annotations produced from previous linguistic analysis steps within a GATE pipeline. JAPE can be applied to several language analysis tasks such as lexico-semantic patterns [Jacobs et al., 1991], finding named entities as well as Noun Phrase (NP) and Verb Phrase(VP) chunking. However, it can also act as a utility language for manipulating annotation output of other language analysers such as a deep parser or other types of processing i.e. bootstrapping and preparing a manually annotated a corpus for processing by a machine learning algorithm.

In certain cases, the matching process is non-deterministic. This is the result of varying in memory addresses of annotation data stored in the Java Virtual Machine (JVM), however the majority of patterns can be matched deterministically. One can also manipulate the control over matching in JAPE, essentially enforcing determinism, which results in a sequential transducer.

A JAPE grammar consists of a set of phases, each of which consists of a set of pattern/action rules. The phases run sequentially and constitute a cascade of finite state transducers over annotations. The left-hand-side (LHS) of the rules consist of an annotation pattern description (including standard regular expression operators such as optional (?), or match zero or more (*), one or more (+), or some specified number of times and '|' for logical or) . The right-hand-side (RHS) consists of annotation manipulation statements. Annotations matched on the LHS of a rule may be referred to on the RHS by means of labels that are attached to pattern elements. Consider the following example:

```
Phase: University
```

```
Input: Token Lookup
```

```
Options: control = appelt debug == true
```

```
Rule: University1
```

```
(  
  {Token.string == "University"}  
  {Token.string == "of"}  
  ({Lookup.minorType == city}):cityName  
):orgName  
-->  
:orgName.Organisation = {kind = "university", rule = "University1"}
```

The LHS is the part preceding the \rightarrow and the RHS is the part following it. The LHS specifies a pattern to be matched to an annotated GATE document, whereas the RHS specifies what is to be done to the matched text. In this example, we have a rule entitled `University1`, which will match text annotated with two `Token` annotations

with a `string` feature of *University* and *of*, followed by additional text annotated as a `Lookup` with `minorType` of *city*. A typical GATE IE pipeline would normally consist of a finite state gazetteer lookup of common named entities and as well as trigger phrases. `Lookup.majorType==city`, corresponds to a mention of a major city found in the task i.e. Dublin, Sheffield, Galway, Berlin. A `Lookup` corresponds to any gazetteer generated dictionary lookup annotation. In this case we are constrained to the flat taxonomy of the gazetteer to only match cities via the feature `minorType==city`. With respect to `Token` annotations, these are outputted by a standard tokeniser, executed in a earlier stage of the pipeline. The rule mixes both lexical and flat semantics, hence it is an example of a lexico-semantic pattern for named entity recognition.

Once this rule has matched a sequence of text, the entire sequence is allocated a label by the rule, and in this case, the label is `orgName`. On the RHS, we refer to this span of text using the label given in the LHS; `orgName`. Hence, this text is assigned an annotation of type `Organisation` and a `rule` feature set to `University1` and another feature `kind` with the value set to `university`. Features in GATE are arbitrary attribute value pairs. The values can be primitives in JAVA such as integers, booleans or string values or references to annotation objects or any Java object i.e. a list. They are useful for storing linguistic or semantic information, additional context information as well as debugging information, such as which JAPE rule fired to create the annotation, in this case `University1`.

The beginning of the JAPE grammar contains a phase name, e.g. `Phase:Organisation1`. JAPE grammars can be cascaded, and so each grammar is considered to be a ‘phase’. The phase name makes up part of the JAVA class name for the compiled RHS actions. The list of the annotation types to be uses in the grammar is specified using `Input:Lookup Token` because the only annotation types we use on the LHS are `Lookup` and `Token` respectively. If no annotations are defined, all annotations will be considered for matching. Although it is not visible in this case, one can embed JAVA code in the RHS of a JAPE rule. Consequently one can manipulate and even delete matched annotations or

access the entire annotation set and any annotations produced by preceding JAPE phases or any other language processors. The ability to manipulate the annotation set, can introduce a form of memory and even backtracking for each rule so the JAPE language can approximate the recognition power of a context free grammar.

With respect to options in a JAPE grammar there are the following options:

- **Control**; in this case, `appelt`. This defines the method of rule matching which is longest match
- **Debug**. When set to true, if the grammar is running in Appelt mode and there is more than one possible match, the conflicts will be displayed on the standard output.

Of particular interest is the **Control** option, of which there are the following different settings:

- `appelt`, only one rule can be fired for the same region of text, according to a set of priority rules. Priority operates in the following way:
 - From all the rules that match a region of the document starting at some point X, the one which matches the longest region is fired.
 - If more than one rule matches the same region, the one with the highest priority is fired
 - If there is more than one rule with the same priority, the one defined earlier in the grammar is fired.
- `first` - the rule fires for the first match that's found. This makes it inappropriate for rules that end in '+' or '?' or '*'. Once a match is found the rule is fired; it does not attempt to get a longer match.
- `once` - style, once a rule has fired, the whole JAPE phase exits after the first match.

- **brill** - fire all matches on a segment of text. Hence text segments could be allocated more than one entity type. No priority ordering is necessary. Brill will execute all matching rules starting from a given position and will advance and continue matching from the position in the document where the longest match finishes.
- **all** - is similar to Brill, in that it will also execute all matching rules, but the matching will continue from the next offset to the current one.

What is important is the the control options above, if combined appropriately, can enforce deterministic finite state transduction over annotations. This forms the basis for parsing for Controlled Language IE tools - CLIE in GATE, which, as we will show in Chapters 4 and 5, allows us to parse the CLOnE language and author an ontology. CLIE also serves as an enabling technology for our controlled annotation tools in Chapter 6.

2.4 Ontology based Information Extraction (OBIE)

Ontology based Information Extraction (OBIE) has recently begun to emerge as a subfield of IE. The term was roughly conceived a few years ago [Wimalasuriya and Dou, 2010], although work was carried out by Hwang on ontologies from text as early as 1999 [Hwang, 1999]. Recently there have been many publications produced concerning OBIE, including a workshop [Adrian et al., 2008]. Many of these systems are related to ongoing projects. An ontology based IE system is a system that processes unstructured or semi structured natural language text, while guided by an ontology, in order to extract useful facts or snippets of information and subsequently present the output as an ontology [Wimalasuriya and Dou, 2010]. While OBIE is a new field, there is a general consensus that it has significant potential. Although IE, and by extension OBIE, is critical for converting a large portion of unstructured text on the Web into structured facts, OBIE also has a crucial role with respect to bootstrapping the Semantic Web, which is critically dependent on the creation of semantic content. While the Linked Data movement has been extremely successful at converting non semantic web structured data into semantic

web data formats, the progress of creating of semantic content from unstructured text has been slow. As discussed by Popov et al [Popov et al., 2004], “it is difficult to imagine that the creation of semantic content will happen in a manual fashion and consequently automatic metadata generation is essential in order to make the Semantic Web real”. Hence web pages must be embellished with semantic metadata a process know as *semantic annotation* [Handschuh, 2005, Cimiano et al., 2004]. While there is considerable overlap between Semantic Annotation and OBIE, a strict distinction can be made between both of them, in that at OBIE is concerned with populating the ontology and knowledge base from text (and also, as mentioned earlier the ontology’s hierarchy is used to guide and inform the extraction process), while Semantic Annotation is concerned with associating ontological information to portions of the text i.e. associating the URI of a concept in an ontology to a *mention* of that same concept in text. *Mentions* are captured typically in IE systems as standoff-markup or annotations. Hence, OBIE will modify the ontology while Semantic Annotation will modify the text. Note that there is a *many to one* relationship between multiple mentions of an instance in ontology in text and the actual single instance in the ontology, which may have been extracted from text (this is is not limited to instances of classes, but may occur at the class level, datatype and object property levels). Typically, if an instance is not in the ontology, the portion of text contained in the discovered mention is extracted and used to populate the ontology with a new instance. Subsequent mentions or referents in text to the instance are also extracted but are stored as additional labels of the instance. Labels of instances can be used for further re-identification of the instance in text. Additionally, semantic annotation serves as quality assurance process for the ontology by:

1. tracking the provenance of a newly created instance of relation in the ontology. Hence the original source of the extraction can be traced for any errors in the ontology.
2. testing whether an ontology is, at the class level, a good representation of the target

domain.

Providing a review of OBIE systems is beyond the scope of this section, but rather OBIE systems will be reviewed in the context of automatic semantic annotation platforms in Chapter 3. We move next to provide an overview of OBIE tools in the GATE platform as they provide the backbone for the Controlled Language IE (CLIE) tools on which the CNL research for ontology authoring and semantic annotation in this work is based.

2.4.1 Ontology Based IE tools in GATE

There are several resources available in GATE to support working with ontologies as well as enriching linguistic analysis with the hierarchical properties of an ontology. They consist of the following:

- The *Ontogazeteer*²⁹ is a hierarchical gazetteer, which associates the entities from a specific gazetteer list with a class in an ontology loaded in GATE. The ontogazeteer contains mappings between lists and class URIs. When the ontogazeteer is executed on text, for a given mention of a class or instance, a **Lookup** annotation is created and a feature is attached which contains a URI reference to the corresponding ontology resource. The ontogazeteer is useful for small ontologies, where accuracy is critical and in addition the IE developer is not constrained by time, since the mapping file must be created manually.
- The *OntoRootgazeteer*³⁰ is a type of a dynamically created gazetteer. It combines with few other generic GATE resources such as the tokeniser, sentence splitter, POS tagger and morphological analyser. It is capable of producing ontology-based annotations over the given content when provided with an ontology. In order to produce annotations, it is essential to pre-process the ontology resources (e.g., classes, instances and properties) and extract their human readable lexicalisations. The names

²⁹<http://gate.ac.uk/sale/tao/#x1-32900013.3>

³⁰<http://gate.ac.uk/sale/tao/#x1-33800013.8>, Accessed Wed 24 Jul 2013 15:48:21 IST

and labels for all resources from the ontology are processed using the generic GATE processing resources mentioned above in order to create an in memory gazetteer. Similar to the Ontogazeteer, when the OntoRootGazeteer is executed over text, a `Lookup` annotation for the corresponding mention of a class or instance is created and a feature is attached which contains a URI reference to the corresponding ontology resource. The ontogazeteer is useful for small to medium size ontologies, but may over-generate lexicalisations and sacrifices accuracy over coverage. It is an extremely useful preprocessing tool if the IE developer is constrained by time and if the manual mapping process necessary to use OntoGazeteer is too time consuming.

- The *Large-Knowledge Based Gazetteer*³¹ provided by Ontotext³² provides support for ontology-aware NLP. One can load any ontology from RDF and then use the gazetteer to obtain `Lookup` annotations that have both instance and class URI features. Currently, the large KB gazetteer does not use GATE ontology language resources. Instead, it uses its own mechanism to load and process ontologies. The Large KB gazetteer grew out of the semantic search platform Ontotext KIM and was developed by the KIM team³³. In order to use the gazetteer, one must create an empty dictionary, create a configuration file to connect to the local RDF ontology or remote Sesame RDF database to be used and specify a SPARQL or SERQL query that will retrieve a subset of that ontology as a dictionary. It is extremely suitable for enabling ontology aware lookup in text, using resources from the Linked Data Cloud.
- *Ontology Aware Finite State Transduction*³⁴ combines the power of ontologies with JAPE's pattern matching mechanisms and can ease the creation of applications. It is example of using an ontology to aid the IE task. An ontology aware JAPE

³¹<http://gate.ac.uk/sale/tao/#x1-34400013.9>, Accessed Wed 24 Jul 2013 15:48:21 IST

³²www.ontotext.com, Accessed Wed 24 Jul 2013 15:48:21 IST

³³<http://www.ontotext.com/kim>, Accessed Wed 24 Jul 2013 15:48:21 IST

³⁴<http://gate.ac.uk/sale/tao/#x1-38600014.10>, Accessed Wed 24 Jul 2013 15:48:21 IST

transducer will treat all occurrences of annotations with the feature `class` differently from other features. In ontology-aware mode the matching between two class values will not be based on simple equality but rather hierarchical compatibility. For example if the ontology contains a class named ‘Politician’, which is a sub class of the class ‘Person’, then a pattern of `Entity.class == ‘Person’` will successfully match an annotation of type `Entity` with a feature class having the value ‘Politician’. If the JAPE transducer were not ontology-aware, such a test would fail. This permits the IE expert to write generic rules for pattern matching as rules that apply to several types of entities mentioned in the text can be written using the most generic class they apply to and the rules do not need to be repeated for each subtype of entity.

- The *GATE Ontology API*³⁵ hides the details of the actual back-end implementation and allows for the simple manipulation of ontologies by modelling ontology resources as easy-to-use Java objects. Ontologies can be loaded from and saved to various serialisation formats i.e. RDF/XML, N3, N-triples or Turtle syntax. The GATE ontology support roughly conforms to the representation, manipulation and inference of OWL-Lite.
- The *Ontology Annotation Tool - OAT*³⁶ is a GATE plugin, which enables a user to manually annotate a text with respect to one or more ontologies. The required ontology must be selected from a pull-down list of available ontologies. The OAT tool supports annotation with information about the ontology classes, instances and properties. In addition, it allows the user to manually populate ontologies from text. Furthermore, there is a *Relation Annotation Tool - RAT*, which is designed to annotate a document with ontology instances and to create relations between annotations with ontology object properties. It is compatible with OAT, but the

³⁵<http://gate.ac.uk/sale/tao/#x1-35300014>, Accessed Wed 24 Jul 2013 15:48:21 IST

³⁶<http://gate.ac.uk/sale/tao/#x1-35300014>, Accessed Wed 24 Jul 2013 15:48:21 IST

focus is on relations between annotations.

As mentioned earlier, this section does not provide an exhaustive review of OBIE systems, rather we survey them in Chapter 3, under automatic semantic annotation systems. However, worth noting, are recent developments with respect to ontology driven approaches using learning systems for entity and relation extraction. Regarding entity extraction, recent work uses a machine learning algorithm, modified to take advantage of the hierarchical structure of the ontology [Li and Bontcheva, 2007]. The OBIE learning algorithm is based on the Hieron large margin learning algorithm for hierarchical classification [Dekel et al., 2004]. It exploits the class label structure by learning a Perceptron model for each class and ensures that the proportion of distance between two classes is close in the tree hierarchy. The authors evaluated their systems on a corpus of almost 300 news articles within the business, international and UK political domains. The corpus was manually annotated according to the Proton ontology³⁷. As no baseline OBIE systems was available, they compared the modified Hieron algorithm to two traditional learning algorithms, Support Vector Machines - SVM and Perceptron with Uneven Margins - PAUM [Li et al., 2005]. The modified Hieron engine outperformed learning algorithms with respect to conventional F-measure.

OBIE research has tended to focus efforts on populating ontologies with concept instances, but recently methods for extracting properties between instances using ontologies as a guide have begun to emerge. In [Wang et al., 2006], the authors, provide a learning approach for ontology based relation extraction. The Support Vector Machine(SVM) algorithm is chosen for the relation extraction task [Wang et al., 2006] and GATE NLP tools are applied to the corpus (taken from the ACE2004 data), to create a set of up to ninety-four features. Using the SVM engine, the relation extraction problem is converted into a multi-class problem, so SVM, being a binary classifier is converted into one against many (a combination of binary classifiers). The features used for the learning algorithm include shallow features such as word features, part of speech (POS) categories,

³⁷<http://proton.semanticweb.org/>, Accessed Wed 24 Jul 2013 15:48:21 IST

entity subtypes and class information. Deeper features consisted of chunk parse information, full syntactic parse information and dependency trees as well as Wordnet features [Oram, 2001]. For the relation extraction task, they assume the presence of two entities as arguments per candidate relation and in addition they assume that the entities per relation are correct. They found that a linear SVN kernel gave the best precision. The performance of the systems with word features only gave an F1 measure of 31%, while when augmented with additional part of speech and entity features, F measure increased to 56.78%. This is comparable to the best reported results of 55% in the ACE 2003 competition. Entity features including the hierarchical features, which were available from the test corpus used from the ACE 2004 competition, gave the best improvements. Interestingly, the authors noticed a trade off between computational and training resources and an improvement of only 2% for deep features.

2.5 Natural Language Generation

Natural Language Generation(NLG) is the production of natural language text from some non-linguistic source such as knowledge base or database [Reiter and Dale, 2000]. NLG systems combine linguistic knowledge and application domain knowledge to automatically produce reports, support authoring aids, provide textual explanations and help messages as well as generate technical documentation. One example of an NLG system is FOG [Goldberg et al., 1994], which produces weather reports in English and French when provided input in the form of a graphical weather depiction. NLG is closely related to Natural Language Understanding (NLU), the study computer systems that understand human language. Both NLU and NLG fields combine to form the field of NLP. While one could view NLG at an abstract level as the inverse of NLU, the internal operations and tasks differ for both. As described by McDonald [McDonald and Bolc, 1988], NLU is concerned with *hypothesis management*, in other words, finding the appropriate interpretations from multiple hypothesis at any given stage within the NLU process. NLG on the other hand is concerned with *choice*, in otherwords it is goal driven, so given several

different methods available for achieving a desired goal which one should be used?

While reversible NLU/NLG systems have an intuitive appeal and have been the subject of some research [Pereira and Warren, 1980], they are in fact difficult to construct in practice. This is because important issues in NLU do not necessarily always have a direct equivalent in NLG i.e. poor spelling and grammatical errors are not of concern to NLG. In addition, the representations produced by parsers differ from the input formats required by most realisers. Hence, these fundamental incompatibilities make it difficult to build a systems that can both parse and realise natural language [Reiter and Dale, 2000]. Consequently, it is unlikely that NLG can ever be viewed as NLU in reverse.

There are a number of options to consider when deciding on whether NLG techniques are appropriate such as: whether or not graphical visualisations would provide a better user experience? Whether text is the right presentation medium? What is the degree of variation in the text? Finally, whether automation is justifiable with respect to volume, speed and consistency requirements? Analysis requirements, when building an NLG system, aside from analysing the clients and the proposed functionality, require a corpus based approach, which consists of providing text examples of output for the proposed NLG system. An initial human authored corpus is assembled consisting of output examples which are aligned to non linguistic input data. Then the corpus is analysed in terms of input data and corpus content. A target text corpus is developed and a formal functional specification is created. Factors to be considered are unchanging text (canned text), directly available data (facts in the knowledge base), computable data (knowledge that can be inferred) and unavailable data(where additional external knowledge is required).

Current NLG systems can be classified into two modes of user interaction:

- **static systems** that produce texts as a whole which are read later by users i.e. technical documentation. These systems do not receive direct user feedback.
- **Interactive** systems that generate explanations or dialogue. They track user interaction history and build up a user model and adapt to user feedback.

Another factor to take into consideration is the modality of the system i.e. whether the generated output is multimodal in that it contains graphics, sounds, gestures videos and other media? Document formatting of output is also an issue particularly when generating hypertext. This becomes even more complex when generating a combination of both multimedia and hypertext or hypermedia.

2.5.1 Generation Architectures and Tasks

With respect to generation component tasks, although researchers differ on when to include or exclude each one, the consensus is that the NLG common tasks are:

- **Content determination**, which is the process of deciding what entities, concepts and relations to choose from the knowledge base to present to the user. *Messages* are constructed from the underlying knowledge base/input data source. Messages are aggregations of data that may correspond to the meaning of a word or phrase. Messages are based on entities, concepts and relations.
- **Discourse planning**, which involves organising, ordering and structuring the propositional content into coherent units, paragraphs, documents. The task is concerned with issues such as conceptual grouping and rhetorical relations. Both content determination and discourse planning output messages in the form of *text plans*.
- **Sentence aggregation or planning**: One to one mapping from messages to sentences can result in incoherent text. Hence, messages must be combined to larger more complex sentences. The output of this task is a sentence specification or sentence plan. Sentence planning involves both combining or eliminating linguistic structures to produce more fluent/concise texts.
- **Referring expression generation** involves choosing appropriate expressions to identify entities (e.g. choosing between *the woman at the window* and the pronoun

she). A major issue is the avoidance of introducing any ambiguous referent in order to ensure that the hearer/reader recognises what entity is being talked about. However, there is a trade off between avoiding introducing ambiguity and reducing text fluidity.

- **Lexicalisation** is concerned with choosing particular words within the system lexicon for a particular language in order to express concepts, relations.
- **Linguistic realisation** is concerned with the grammatical rules for forming words - **morphological realisation** (such as *walk* + *ed* to realise the past tense of *to walk*) and **sentence aggregation** is concerned with syntax realisation, which is the rules for sentence formation. So whether the subject should proceed the verb and when both subject and verb should agree in number. Finally, **orthographic realisation** is concerned with issues such case, punctuation, font, column width and rules such as all sentences should start with an upper case letter and that sentences should merge when an abbreviation symbol is encountered.

Aside from text oriented tasks, others issues that should be taken into consideration are speech and multimedia tasks such as ordering, segmentation, media allocation and generation. For related work, with respect to multimodal generation systems, we refer the reader to [Maybury, 1995]. While researchers differ with respect to generation tasks and their place in system architecture, the general consensus is that NLG architecture can be broken down into a pipelined architecture consisting of the following components [Reiter, 1994]:

- **Content Section and Discourse Planning (Text Planning)**: This component involves starting with the communicative goal and making all the necessary choices to select content from the knowledge base. Both content selection and discourse planning are not always clearly separated. In addition, some systems perform semantic and conceptual aggregation at this stage. The most common approach to discourse planning or text planning is to use *text schemas*. It is used quite often

in applied NLG systems because of its efficiency. Text schemas encode and plan the structure of the text in a template. The output of the stage within the NLG pipeline is an intermediate representation called a *sentence plan*.

- **Sentence Planning** or micro planning is concerned with expressing the content chosen from the knowledge base as sentences. The task of sentence planning is to map conceptual structures into linguistic ones by selecting the corrected syntactic structure, aggregating sentences and paragraphs, selecting the correct content words (lexicalisation) and choosing the correct referring expression for a content word. The output at this stage is called a *sentence plan*
- **Surface Realisation:** the purpose here is to use syntactic information to convert sentence plans into actual text. A grammar of the target language is used to realise correct output i.e. subject verb agreement, handling of reflexives and generation of correct morphology.

NLG tasks use input from several knowledge sources, in order to generate correct output. The most fundamental knowledge source is the knowledge base itself which contains all the necessary information to realise the desired communicative goal. This knowledge may be domain specific or domain independent and may be derived from existing external sources such as the upper ontologies. Furthermore, the knowledge may have been crafted manually by an ontology engineer or acquired via a knowledge extraction application. Another source of knowledge can be the discourse interaction history, whereby cached generated text can be reused to avoid repetitious processing. In addition, user models can also serve as an input knowledge source, whereby the beliefs, goals, intentions and profile of the user can influence each stage in the NLG pipeline. Furthermore, linguistic knowledge, in the form of grammars, also plays a crucial role. Types of grammars are systemic grammars [Mann et al., 1983], unification grammars such as Functional Unification Grammar (FUG) [McKeown et al., 1990] and General Phrase Structure Grammar (GPSG) and GPSG augmented with referents [Sigurd, 1991], Tree Adjoining

Grammars (TAG)s [Joshi, 1987] bidirectional or reversible grammars [Wilks, 1990]. Finally, grammars can be encoded as templates (modern template systems are XML based) [Reiter, 1995, Wilcock, 2005], which have been enriched with some morphological and lexicalisation tools. Finally, lexicons also play an important role. Typically a lexicon will contain information such as part of speech categories for a lexical entry as well word senses and morphological derivational and inflectional rules for each entry. Ontology-lexica [Buitelaar et al., 2009a] will begin to play an important role here, whereby an ontology can be lexicalised and compiled into internal dictionary resources of NLG applications i.e. ontology verbalisation of controlled language [Davis et al., 2012].

2.5.2 Shallow versus Deep Generation

Shallow NLG involves the use of canned text and templates. Such NLG systems are highly domain dependent. They are efficient and fast, easy to implement by non NLG specialists and in particular the experts of the knowledge domain. However, the disadvantage is that they can become difficult to maintain. For instance, if a system is retargeted to a new domain of knowledge, linguistic resources must be redesigned from scratch, or alternatively if there is an extension to the knowledge base or a linguistic resource is required, additional templates must be written. Hence, as more templates are added to the existing database over time, they can become difficult to maintain. This is obvious when more syntactic variability is required, as more templates must be written. In addition, both text planning and sentence planning capabilities can be quite restricted in template based NLG system [Reiter, 1995]. Furthermore, shallow NLG systems have more difficulty coping with a domain shift.

Deep Generation systems on the other hand are very maintainable with regard to extending the linguistic output of the system. Usually this can be done with minimal effort in contrast to template systems. As they are based on linguistically well informed knowledge and grammars, there is a risk of over-generation, however these systems produce higher quality text output and are flexible with regard to sentence aggregation. They

can handle low level linguistic phenomena such as morphology and number agreement with ease while with shallow/template NLG systems, the ability to handle such syntactic realisation can be programmatically expensive. However, NLG expertise is required to maintain the systems, which may be difficult to obtain. Although, deep NLG systems can also offer high quality multilingual output, this may not be cost-effective to implement in particular if localised strings within a template NLG system are deemed sufficient. However, other scenarios may view quality output as a crucial requirement. Finally, deep NLG systems can play a role in enforcing document standards such as writing standards for AECAM Simplified English³⁸ or controlled languages used in technical writing.

More recent approaches to NLG involve the use of XSLT [Reiter, 1995](discussed earlier), which is essentially template based generation but has more powerful language processing capabilities. Stochastic approaches are also in use where the deep generation grammar is replaced by a stochastic language model [Oh and Rudnicky, 2000]. Finally, the third modern approach to NLG is that of hybrid NLG [Klarner, 2004b]. Hybrid NLG can take the following forms:

- i)Template/shallow based NLG with embedded elements of deep NLG or
- ii) Deep NLG with embedded elements of Shallow NLG, or
- iii)Concurrent Deep and Shallow NLG.

HyperBug [Klarner, 2004a] is an example of (iii), where tactical generation or linguistic realisation is achieved by means of either deep generation or surface generation based on a decision module. Both approaches return a surface structure, however in the case of the deep generation part, the module supplies an additional third bootstrapping quality to the system, whereby the module in addition to the outputted surface structure, returns a template for storage within the shallow NLG database and a reference to the decision module. Hence, subsequent input with a similar semantic structure can be realised faster by using a template without having to call the deep generation module again.

³⁸<http://www.asd-ste100.org/>, Accessed Wed 24 Jul 2013 15:48:21 IST

With respect to evaluating NLG systems, increased efforts towards improving evaluation methodologies are needed. The problem is largely due to less activity in the NLG field in comparison to NLU. Another problem, is the *lack of well defined input and output* for NLG systems [Wilks, 1990]. NLG systems vary in their input, domain, tasks and target media, and furthermore evaluation techniques within NLU cannot be simply transferred across to NLG [Bontcheva, 2003, Wilks, 1990]. Consequently, it is difficult to objectively compare NLG systems. NLG systems tend to be evaluated in a *extrinsic* or *black-box* manner. This type of evaluation does not focus on any particular module in the NLG system but evaluates the system in its entirety. This may involve asking users to judge the quality of the NLG output. Users may be asked to conduct a *repeated measures task based evaluation*, whereby they are requested to interact with two versions of the system and complete a set of tasks. Typically, they are provided with some background knowledge and set of multiple choice questionnaires in order to quantitatively measure the results. Other types of evaluation are *glass-box testing* where, a particular module in the NLG system is targeted for evaluation. In the context of this thesis, we apply a *black-box* technique, as the NLG component within RoundTrip Ontology Authoring (ROA) (See Chapter 5) generates controlled language, so we are not concerned with fluidity of NL output as choice with respect to lexicalisation is restricted and we do not generate referring expressions. For full NLG systems, there are numerous metrics available to measure accuracy and text fluidity, some of which are based on metrics for Machine Translation (MT) as well as string edit distance and tree edit metrics. We refer the reader to [Mellish and Dale, 1998, Bontcheva, 2003] for related work with respect to evaluation of NLG systems.

2.6 Natural Language Generation from Ontologies

Natural Language Generation (NLG) takes structured data in a knowledge base as input and produces natural language text, tailored to the presentational context and the target reader [Reiter and Dale, 2000]. NLG techniques build user context models and use them

to select appropriate presentation strategies. For example, delivering short summaries to the user's WAP phone or a longer multimodal text if the user is using their PC desktop.

In the context of Semantic Web or knowledge management, NLG can be applied to provide automated documentation of ontologies, knowledge base or metadata summarisation. Unlike human-written texts, an automatic approach will constantly keep the text up-to-date which is vitally important in the Semantic Web context where knowledge is dynamic and is updated frequently. NLG also allows for generation in multiple languages without the need for human or automatic translation [Aguado et al., 1998]. The advantage of automatically producing textual documentation from ontologies, is that it is more readable than the corresponding formal notations. Hence, users, who are not knowledge engineers, can more easily understand and use ontologies. Secondly, a number of applications have now started using ontologies to encode and reason with internally, but this formal knowledge needs to be also expressed in natural language in order to produce reports, letters, etc. In other words, NLG can be used to present structured information in a user-friendly way. There are several advantages to using NLG rather than using fixed templates where the query results are filled in: NLG can use different sentence structures depending on the number of query results, e.g., conjunction vs itemised list. Depending on the user's profile of their interests, NLG can include different types of information – affiliations, email addresses, publication lists, indications on collaborations (derived from project information). Given this variety of what information from the ontology can be included and how it can be presented, depending on its type and amount, writing templates will be unfeasible because there will be too many combinations to be covered.

This variation comes from the fact that it is expected that each user of the system will have a profile comprising of user supplied (or system derived) personal information (name, contact details, experience, projects worked on), plus information derived semi-automatically from the user's interaction with other applications. Therefore, there will be a need to tailor the generated presentations according to user's profile. NLG systems that are specifically targeted towards semantic web ontologies have started to emerge only

recently. Initial ones were based on templates, verbalising closely the ontology structure. More recent ones generate more fluent reports, oriented towards end-users, not ontology builders. In contrast to these applied NLG approaches, at the other end of the spectrum are sophisticated ones, which offer tailored output based on user models. There is a trade-off between applied approaches, which explore generalities in the domain ontology with low customisation overheads, and, on the other hand, more sophisticated, flexible and expressive systems, which, tend to be difficult to adapt by non-NLG experts. Experience shows that knowledge management and semantic web ontologies tend to evolve over time, so it is essential to have an easy-to-maintain NLG approach.

2.6.1 Generation from Semantic Web Ontologies

The ONTOGENERATION project [Aguado et al., 1998] explored the use of a linguistically oriented ontology (the Generalised Upper Model (GUM))(Bateman et al. 1995) as an abstraction between generators and their domain knowledge base (chemistry in this case). The Generalised Upper Model (GUM) is a linguistic ontology with hundreds of concepts and relations, e.g., part-whole, spatio-temporal, cause-effect. The types of text that were generated are: concept definitions, classifications, examples, and comparisons of chemical elements. However, the size and complexity of GUM make customisation more difficult for non-experts. On the other hand, the benefit of using GUM is that it encodes all linguistically-motivated structures away from the domain ontology and can act as a mapping structure in multilingual generation systems.

2.6.2 Modern Shallow Generation

Wilcock [Wilcock, 2005] provides an overview of shallow XML-based Natural Language Generation from ontologies. His work provides descriptions of XML based pipelined architectures for NLG, involving: Text Planning, Micro Planning and Surface Realisation. The research is based on practical experience which applies NLG to a spoken dialog system. The author claims that XML as a generation tool fits into the pipeline model for

NLG. He argues that powerful methods for processing XML already exist. In Wilcock's approach, XML transformations are performed on text plan trees in order to produce text specification trees using Extensible Stylesheet Language Transformations (XSLT). The author [Wilcock, 2005] emphasises that the role played by XML transformations across text plans as a consequence of using XML templates implies template based text planning and not just template based generation, whereby the text plan is passed through various stages of the NLG pipeline for processing using XSLT. The author maintains that the template based approach to text planning offers an efficient alternative to AI planning techniques traditionally used within Deep NLG, which can be expensive and complex, involving potentially exponential amounts of backtracking. However, Wilcock maintains XSLT alone is not suitable for the surface realisation of all languages, especially where complex morphological processing is required as is the case in Finnish. One can however use XSLT with extension functions for JAVA or reuse existing morphological processing resources. An interesting avenue proposed, is a hybrid combination of shallow XML-based methods in combination with traditional deeper NLG methods such as OpenCCG [Foster and White, 2004, White and Baldrige, 2003, White, 2004] (which in itself combines both unification based and statistical approaches). Such a hybrid approach is likely to fulfil needs for producing scalable, efficient and high quality methods of generating texts from ontologies.

2.6.3 NLG in CLEF - Clinical E-Science Framework

CLEF [Rector et al., 2003] focuses on integrating clinical care with biomedical research. It aims to develop methods for managing and using pseudonymised repositories of long-term patient histories, which contain and link information relating to genetic, genomatic and image information. It has applications for clinical research and can be used to support patient care. The project involves the use of HLT for IE and NLG from metadata CLEF aims to create an Electronic Patient Repository. In order for the data to be of use to a scientist or clinician, it must be easily accessible and user friendly.

Techniques from language generation are applied to produce reports from integrated information and metadata in the repository and to provide an electronic health care record for the medical specialist. In addition, the NLG contribution to CLEF aims to provide a Natural Language Window into the Knowledge base in order to aid knowledge engineers to build and maintain the knowledge base and furthermore to permit clinical experts to provide quality assurance across the the knowledge base while still being shielded by the underlying complex formalisms. The initial CLEF [Rector et al., 2003] work references WYSIWYM technology and natural language directed feedback [Power et al., 1998a] as the basis for NLG. The paper [Hallett et al., 2006] also focuses on the problems of presenting aggregated clinical data, in the form of comprehensible textual reports, from a vast and rich source of varying clinical knowledge. The authors discuss the system requirements gathered from clinicians and summarise the results as different report types, depending on user requirements. Input to the report generator is a user selected type of report or selected events in a timeline. Hence the input from the knowledge base (implemented using DAML+OIL) for generation is a chronicle, which is a highly structured representation of a patient record stored in semantic nets (no other details are provided with respect to the knowledge representational language). A chronicle contains facts and relations and relations are categorised into three types according to their role: rhetorical relations, ontological relations and attribute relations. The system follows a classical NLG architecture: Content Selector, Content planner and Syntactic realiser. The Content planner is tightly coupled with the content selector as document structure is decided when a user selects an event or collection of events. Sentence Aggregation occurs primarily at the conceptual level at the content planning stage rather than the syntactic stage. The document planner uses a combination of schemas with a bottom-up approach. Rhetorical relations holding between a simple event are often realised as a single sentence. Complex individual events are realised as individual clauses or sentences, which are connected via the appropriate rhetorical relation. Depending on the type of report, the focus may move to different events, which consequently has an effect at the

syntactic realisation stage. Other solutions to aggregation are to restrict information to events that deviate from the norm i.e. abnormal test results.

2.6.4 Summary Generation from Ontologies

This section focuses on the summary generation problem, addressed in the ONTOSUM system [Bontcheva, 2005], developed in the context of the SEKT research project. Summary generation starts off by being given a set of statements regarding instances in the ontology. (i.e., triples), in the form of RDF/OWL. Since there is some repetition, these triples are first pre-processed to remove already said facts. In addition to triples that have the same property and arguments, the system also removes triples involving inverse properties with the same arguments as those of an already verbalised one. The information about inverse properties is provided by the ontology (if supported by the representation formalism). The ONTOSUM system is implemented as a set of components in the GATE infrastructure [Bontcheva et al., 2004]. In particular, GATE makes use of its ontology support, which provides language-independent access to ontologies. The lexicalisations of concepts and properties in the ontology can be specified by the ontology engineer, be taken to be the same as concept names themselves, or added manually as part of the customisation process. ONTOSUM is parameterised at run time by specifying which properties are to be used for building the lexicon. A similar approach was first implemented in a domain- and ontology-specific way in the MIAKT system [Bontcheva and Wilks, 2004]. In ONTOSUM, it is extended towards portability and personalisation. Similar to the PEBA system, summary structuring is done using discourse/text schemas [Reiter, 1994], which are script-like structures which represent discourse patterns. They can be applied recursively to generate coherent multi-sentential text. The schemas are independent of the concrete domain and rely only on a core set of four basic properties – active-action, passive-action, attribute, and part-whole. When a new ontology is connected to ONTOSUM, properties can be defined as a sub-property of one of the four basic ones. Consequently, ONTOSUM will be able to verbalise them

without any modifications to the discourse schemas. However, if more specialised treatment of some properties is required, it is possible to enhance the schema library with new patterns that apply only to a specific property. ONTOSUM then performs semantic aggregation, i.e., it joins RDF statements with the same property name and domain as one conceptual graph. Without this aggregation step, there will be three separate sentences instead of one bullet list, resulting in a less coherent text. Finally, ONTOSUM verbalises the statements using the HYLITE+ surface realiser. The output is a textual summary.

2.7 Conclusions

This purpose of this chapter has been to review the core technologies underlying this thesis. Section 2.1 provided a brief history of the Web as well as an overview of the Semantic Web, Ontologies and Linked Data. We have discussed briefly, the history and origins of Human Language Technology (HLT) in Section 2.2. We have also provided detailed summaries of the the core technologies and techniques with respect to Information Extraction (IE) and Natural Language Generation (NLG), both of which are the key language technologies used in this thesis. We have furthermore discussed HLT with respect to the Semantic Web under the context of Ontology Based IE (OBIE)(see Section 2.4 and Natural Language Generation(NLG) from ontologies (See Section 2.6). Finally, we have reviewed key recent work with respect to both fields and their applications to the Semantic Web. With respect to OBIE, we can make the following observations:

- We note some increasing awareness within the IE community (and the larger NLP/CL community) of the potential benefits of Semantic Web technologies when building their respective language processing systems. The Semantic Web community on the other hand has been well aware of the important role NLP, particularly IE, has to play in bootstrapping the creation of semantic data i.e. semantic annotation of web pages and ontology population and learning from text. Moreover, there has been the recent acceptance of workshops at the Language Resources and Eval-

uation Conference (LREC) involving Semantic Web technologies³⁹. Conversely, at the International Semantic Web Conference (ISWC) conference series, there are now regular workshops with an NLP focus⁴⁰. Furthermore, the Linguistic Linked Data initiative is also gaining momentum⁴¹. Finally, uptake, in the IE community has become evident, with the recent inclusion of OBIE in the Text Analysis Conference (TAC) via the 2012 Cold Start task.

- Few NLP frameworks aside from GATE, provide built-in Ontology support. Although many IE systems can export their output to Semantic Web languages, IE developers are still not fully aware of how Semantic Web ontologies can inform and drive IE tasks such as NER, RE and CO. More research effort is required to investigate how the subsumption hierarchy and properties of ontologies can play a role in crafting extraction rules or training classifiers for IE subtasks. Ontology Lexicalisation, which is the enrichment of ontology labels with proper linguistic descriptions, can have a role to play in the automatic bootstrapping of IE systems with ontology aware language resources (see Chapter 7).
- Finally, many OBIE and Semantic Annotation performance metrics still have a tendency not to factor in the distance of an ontology resource in their metrics. In other words, the identification of correct entities should occur in scalar manner, rather than a binary classification, i.e. correct or incorrect value [Maynard et al., 2008]. Hence, a system classified entity may still be partially correct if it is within the class hierarchy of the correct entity.

With respect to NLG from ontologies, we make the following observations:

- NLG research in general, is still underinvested in comparison to NLU. More research

³⁹<http://www.dcs.shef.ac.uk/~diana/courses/lrec-nlp-semweb-tutorial.html>, Accessed Thu 25 Jul 2013 15:38:22 IST

⁴⁰<http://nlp-dbpedia2013.blogs.aksw.org/2013/>, Accessed Thu 25 Jul 2013 15:38:22 IST

⁴¹<http://www.ld12013.org/>, Accessed Thu 25 Jul 2013 15:38:22 IST

is needed with respect to methodologies and reusable tools and language resources for building and evaluating NLG systems. This is largely due to the continued focus within the NLP community towards NLU and MT. For instance, the first edition of the International Conference on Natural Language Generation Systems⁴² was only held in 2000 (after upgrading from a workshop series), demonstrating the size of the NLG community. There are number of factors, ranging from misunderstanding within the NLP community of the NLG task, in other words, the presumption being the NLG is simply the reverse of NLU, to the difficulty of the task itself in that it is goal driven and concerned with *choice* rather than hypothesis management.

- Despite efforts to create a reference architecture for generation systems (not discussed in this chapter) [Mellish et al., 2006], there is still no consensus on the architecture of NLG systems, let alone a commonly used reference implementation. So while GATE serves arguably as an NLP architecture for NLU/IE tasks and Moses may serve many researchers as common architecture for MT tasks [Koehn et al., 2007], no such framework is available for integrating and evaluating heterogeneous NLG components.
- While there is less activity with respect to NLG research in comparison to NLU, there is even more so less activity with respect to NLG for the Semantic Web, largely due to the relatively newness of the Semantic Web field. NLG efforts within the Semantic Web is still quite limited to ontology verbalisation. There is limited work on full NLG from knowledge bases i.e generating tailored reports or summaries, but there have been some efforts to explore the use of Semantic Web ontologies to drive the content selection task [Mellish and Pan, 2008]. Moreover, Ontology Lexicalisation can again play a role in the lexicalisation subtask for NLG (See Chapter 7). Finally, increasing activities with regard to the Multilingual Semantic Web may reinvigorate research efforts with respect to multilingual NLG [Gracia et al.,].

⁴²<http://www.cs.bgu.ac.il/~nlg2000/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

In summary, NLG and OBIE techniques and tools act as platforms for our research work on CNL for Ontology Authoring and Semantic Annotation respectively. Consequently, having provided the necessary foundations, we proceed to Chapter 3 to review both of the aforementioned fields in more detail.

3 Controlled Natural Languages and Semantic Annotation

This Chapter surveys the fields of both Controlled Natural Languages and Semantic Annotation. Section 3.1, provides a brief history and overview of Controlled Natural Languages. We also review two core CNLs which are active in the Semantic Web field, specifically Attempto Controlled English - ACE (Section 3.1.1), Rabbit (Section 3.1.2) and CNL applications of the GF (Grammatical Framework) toolkit (Section 3.1.3). Finally, Section 3.1.5 provides a summary of other CNL work. With respect to Semantic Annotation, Section 3.2, provides an overview of the field and Section 3.2.1 discusses annotation frameworks. Annotation tools are categorised into three parts: manual semantic annotation (Section 3.2.2), semi-automatic semantic annotation (Section 3.2.3), automatic semantic annotation and (Section 3.2.4). Furthermore, we provide a brief overview of related applications to semantic annotation in Section 3.2.5. Finally Section 3.3 concludes the entire chapter, and provides a summary analysis with respect to our research goals concerning both Controlled Natural Languages and Semantic Annotation.

3.1 Control Natural Languages for the Semantic Web

“Controlled Natural Languages are subsets of natural language whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity” [Schwitter, 2007]. The original concept of CNL arose during the 1930s, when a number of influential linguists and scholars devoted considerable effort to establishing a ‘minimal’ variety of English’; The purpose being to make English accessible and usable by as many individuals as possible world wide [Schwitter, 2007]. Although, as stated by

Kuhn [Kuhn, 2010a], CNLs are as old as logic itself and dates back as far as Aristotle's syllogisms [Aristotle, 350] and the work of Frege [Frege, 1879]. An early CNL, was *Basic English*, described in 1932 [Ogden, 1935] as a restricted grammar with only 850 English words. It was developed by linguist and philosopher Charles Kay Ogden as an international auxiliary language, and as an aid for teaching English as a Second Language. Another example is Special English ⁴³, first used on October 19, 1959 and still presented daily by the United States broadcasting service Voice of America (VOA). The intended audience is intermediate to advanced learners of English. *Seaspeak* is a simplified form of English [Strevens and Johnson, 1983]. Its purpose was to facilitate communication between ships whose captains' native tongues differ. It has now been formalised as Standard Marine Communication Phrases (SMCP) [Strevens and Johnson, 1983].

An early CNL within the technical domain included Caterpillar Fundamental English (CFE) [Caterpillar Corporation, 1974] . Since then, CNLs have evolved into many variations and flavours such as Smart's Plain English Program (PEP), Whites International Language for Serving and Maintenance (ILSAM) [Adriaens and Schreors, 1992] and Simplified English⁴⁴, which in 2005 became ASD(Aerospace and Defence Industries Association of Europe) Simplified Technical English⁴⁵. CNLs have found particular favour in large multi-national corporations such as IBM, Rank, Xerox and Boeing amongst others usually within the context of user-documentation production and machine translation [Adriaens and Schreors, 1992, Schwitter, 2007, O'Brien, 2003].

Traditionally, Controlled Natural Languages (CNL)s are split into two major categories: (1) CNLs that improve human readability, mainly for non-native speakers, and (2) those that constrain the text for computational treatment. In [O'Brien, 2003], O'Brien, provides a comparison of CNLs and offers the conclusion that no common core rules within her survey of CNLs can be identified. The work demonstrates that CNLs

⁴³<http://learningenglish.voanews.com/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁴⁴http://www.simplifiedenglish-aecma.org/Simplified_English.htm, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁴⁵<http://www.asd-ste100.org/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

exist in a continuum ranging from informal(human-orientated) to completely formal semantics(machine processable).

Computer Processable Language (CPL) was an early machine processable CNL, developed at Boeing and it included an interpreter, and a reasoner, with up to one thousand general and domain-specific rules. The application was to create a knowledge base for semantic retrieval of video clips based on their captions which were expressed in CPL [Clark et al., 2005].

With respect to knowledge representation via CNLs, early work included Processable English(PENG). Texts written in PENG were deterministically parsed and translated into discourse representations structure, discourse representation structures, and also into first-order predicate logic for theorem proving [Schwitter, 2002]. Furthermore, the work is influential in that it was one of the first CNLs, designed with an incremental parsing approach. It had an editor with lookahead features in order to guide the user [Schwitter et al., 2003]. With respect to CNLs for the Semantic Web, early efforts involved extending PENG in [Schwitter and Tilbrook, 2004], whereby the authors present and discuss PENG-D, a variation of PENG, which targets the CNL to a knowledge representation language (via First Order Logic (FOL)) such as RDFS or OWL.

3.1.1 Attempto Controlled English -ACE

A well known approach involving CNL translation into FOL is the popular CNL, *Attempto Controlled English*⁴⁶ (ACE) [Fuchs and Schwitter, 1996a]. It is a subset of standard English designed for knowledge representation and technical specifications, and constrained to be unambiguously machine-readable into discourse representation structures, a form of first-order logic (ACE can also be translated into other formal languages.)

ACE is a mature CNL and has been in development since 1995 for over fourteen years [Kuhn, 2010a]. It was first introduced by Fuchs and Schwitter [Fuchs and Schwitter, 1996b]. Over forty articles have been published by the Attempto group and over 500 articles listed

⁴⁶<http://www.ifi.unizh.ch/attempto/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

on Google Scholar, containing the term “Attempto Controlled English” [Kuhn, 2010a]. ACE is a general purpose CNL and is not restricted to any specific domain. The grammar of ACE is perhaps the most expressive in that it can parse a variety of syntactic phenomena in comparison to other CNLs. ACE caters for instance for relative clauses, coordinated noun phrases, coordinated adverbial and adjectival phrases, numerical and distributed quantifiers, negation, conditional sentences and some anaphoric pronouns⁴⁷

ACE was adopted as the controlled language for the EU FP6 Network of Excellence REVERSE⁴⁸ (Reasoning on the Web with Rules and Semantics) [Fuchs et al., 2006]. The Attempto Parsing Engine (APE) consists principally of a definite clause grammar, augmented with features and inheritance and is written in Prolog [Hoefer, 2004]. This tool can be tested and demonstrated with a web browser through the *APE Web-interface*⁴⁹. APE web service clients are also available⁵⁰.

REVERSE also proposed ACE OWL, a sublanguage of ACE, as a means of writing formal, simultaneously human- and machine-readable summaries of scientific papers [Kaljurand and Fuchs, 2006, Kuhn, 2006]. ACE itself however prohibits certain very natural constructions such as the use of *only* as an adverb. Since ACE OWL also aims to provide reversibility (translating OWL DL into ACE), OWL’s *allValuesFrom* must be translated into a similar construction which can be rather difficult for humans to read. Furthermore, ACE OWL does not currently support enumerations (OWL’s *oneOf*) and has limited support for datatype properties [Kaljurand, 2006]. ACE OWL also imposes several further restrictions on ACE, such as the elimination of plural nouns. Although ACE itself has a predefined lexicon, unknown words can be used if they are annotated with a part of speech tag, but this does require the user to be familiar with the lexicon

⁴⁷http://attempto.ifi.uzh.ch/site/docs/ace/6.5/ace_constructionrules.html, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁴⁸<http://reverse.net/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁴⁹<http://www.ifi.unizh.ch/attempto/tools/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁵⁰http://www.ifi.unizh.ch/attempto/documentation/ape_webservice.html, Accessed, Thu 25 Jul 2013 16:54:32 IST

[Kaljurand, 2006].

ACEView is a plugin for the Protégé editor⁵¹ [Kaljurand, 2008]. It empowers Protégé with additional interfaces based on the ACE CNL in order to create, browse and edit an ontology. The user can also query the ontology using ACE questions to access newly asserted facts from the knowledge base. ACE has also served as the basis for other applications such as interface language for a first-order reasoner [Fuchs and Schwertel, 2003], a query language for the Semantic Web [Bernstein et al., 2004], an application for the partial annotation of Webpages [Fuchs and Schwitter, 2007] and the usage of ACE for producing summaries within the biomedical domain [Kuhn et al., 2006].

A recent development is the translation of a complete collection of paediatric guideline recommendations into ACE. In [Shiffman et al., 2010], one paediatrician, one physician and one knowledge engineer assessed an extended version of ACE, to see, if it could correctly represent clinical concepts and guideline actions. Crucial Urinary Tract Infection(UTI) action statements were translated into an extended form of ACE. The three experts concluded that ACE was capable of accurately stating the clinical concepts and actions described in the guidelines. Medical terminology was an issue raised by the authors and they discuss the feasibility of incorporating medical vocabularies such as SNOWMED⁵² and UMLS⁵³ into the ACE lexicon. The authors plan to target the ACE rules for a clinical care decision support system to guide clinicians regarding best practice.

3.1.2 The RABBIT Controlled Language

The Rabbit CNL is a another well known implementation [Hart et al., 2008]. It is essentially an extension of CLOnE implementation (see Chapter 4) but is much more powerful with respect to grammar expressiveness and ontology authoring capabilities. Like

⁵¹<http://protege.stanford.edu/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁵²Systematised Nomenclature of Medicine. See <http://www.ihtsdo.org/snomed-ct/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁵³Unified Medical Language System See <http://www.nlm.nih.gov/research/umls/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

CLOnE, Rabbit is implemented using the GATE framework. Rabbit was developed by the national mapping agency in Great Britain - Ordnance Survey. Rabbit can be converted in OWL⁵⁴ to provide natural language support for ontology authoring. OWL development is not the primary objective of Rabbit and not all OWL expressions can be mapped into OWL. Rabbit is primarily a vehicle for capturing, representing and communicating knowledge in a form that is easily understood by domain experts. In [Hart et al., 2007], the authors argue that Rabbit should be considered as the authoritative source for knowledge representation. There are three broad types of sentences in Rabbit - declarations, axioms and import statements. Interestingly, a given class or concept can refer to a specific ontology in Rabbit i.e. one can refer to the animal Duck within a specific ontology - `Waterfowl` as opposed to a default ontology. So more than one ontology can be referenced in the Rabbit language [Hart et al., 2008]. Rabbit attempts to cater for property restrictions such as transitivity and symmetry, but as the authors themselves argue the such concepts are “not aligned to the way people think” and that there is no ideal solution to creating natural language equivalents to property restrictions. Arguably, these issues should be dealt with by support from the ontology engineer and not the domain expert directly.

The ROO - Rabbit to OWL Ontology authoring tool seeks to cater for the entire ontology engineering process [Dimitrova et al., 2008]. It was developed by the University of Leeds and is an open source Java based plugin for Protégé. ROO supports a domain expert in creating and editing ontologies using Rabbit. The authors argue that CNL interfaces tend to ignore the ontology construction process. The design of the ROO interface is based on Ordnance Survey’s proposed ontology development methodology called *Kanga*. Domain experts are involved in the early stages of the ontology engineering process and engage in the conceptualisation of the ontology, while the ontology engineer is involved at the end stages and focus on the logical level of the ontology. The work of [Dimitrova et al., 2008], gives a good overview of Rabbit’s expressiveness with respect to

⁵⁴<http://www.w3.org/TR/owl-features/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

Rabbit syntax patterns and their corresponding ontology mappings such as existential quantifiers, union, disjointness and cardinality. ROO is based on the experiences of the Ordnance Survey(OS) agency with creating large-scale geo-spatial ontologies. The OS identified a number of factors based on their experiences, such as: (1) the difficulty in expressing knowledge constructs in a formal language,(2) the lack of appropriate methodology for capturing the knowledge of domain experts and (3) the poor usability of existing ontology editing tools.

The ROO interface is based on the manipulation of Rabbit statements and not OWL constructs. The technical details of OWL are hidden from the end user. ROO monitors user activity as well as the state of the underlying knowledge base. It provides the appropriate contextual suggestions and assistance to users in the form of ontology construction feedback, syntax highlighting of Rabbit statements and error messages. With respect to user evaluations of ROO and Rabbit, see related work in Chapter 4.

3.1.3 Grammatical Framework -GF

A recent addition to the CNL field is GF - Grammatical Framework which is actually an implementation framework for multiple CNLs [Angelov and Ranta, 2009] and [Ranta, 2004]. The authors claim that GF can cope with a variety of CNLs as well as boost the development of new ones. In [Angelov and Ranta, 2009], the authors reverse engineer ACE for GF in order to demonstrate how portable CNLs are to the GF framework as well as how CNLs can be targeted to other natural languages. ACE is ported from English to five other natural languages. In short, the core advantage of GF is its multilingualism in that its primary task is domain specific knowledge based Machine Translation (MT) of controlled natural languages. GF follows the functional programming paradigm and began as an experimental system in 1998 at XEROX Europe⁵⁵[Dymetman et al., 2000]. The GF framework uses a logical framework based on Martin Loef's type theory [Nordstrom et al., 1990] for building semantic models of lan-

⁵⁵<http://www.xrce.xerox.com/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

guages. It adds a syntax formalism to the logical framework which defines realisations of formal meanings as concrete linguistic expressions. The semantic model is called the *abstract syntax* while the syntactic realisation functionality is called *concrete syntax*. The authors state that GF is multilingual, in that one abstract syntax, acting as an interlingua, can be (given a concrete syntax for one or more source languages) be retargeted to several languages. So the source language sentence is first parsed using a concrete grammar into an abstract syntax which is meaning preserving and reflects the semantics of a given domain (rather than the source concrete syntax). This abstract syntax can then be linearised into the target language(s). A substantial amount of linguistic competence and domain expertise is needed to define a concrete syntax for a given source/target language. Consequently, the authors developed a collection of GF resource libraries to provide a language engineering solution to this issue. The engineering effort is split between domain experts and linguists. The GF libraries now contain a collection of wide coverage grammars for over 15 natural languages. One could view GF as a general framework for developing and extending controlled languages, similar to NLP architectures such as GATE (See Chapter 2). There is increased activity with respect to the GF development and a vibrant open source community, which continues to create language resources for GF. The success is also due to the European project, MOLTO (Multilingual On-Line Translation)⁵⁶. This has boosted the uptake of GF and resulted in many more comprehensive applications. Applications of GF range from mathematical proofing, dialog systems, patent translation [España-Bonet et al., 2011], multilingual wikis and multilingual generation in the culture heritage domain [Angelov and Ranta, 2009, Dannélls, 2008]. In addition, there have been recent efforts to cater for semantic web ontologies in GF. In [Angelov and Enache, 2010] the authors develop a conversion tool for compiling axioms in the SUMO ontology [Niles and Pease, 2001] written in the KIF language [Genesereth et al., 1992] to GF abstract syntax. In addition, the authors produce CNL from the ontology and allow users to edit SUMO axioms in CNL. SUMO

⁵⁶<http://www.molto-project.eu/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

contains natural language templates for natural language generation, which were processed and covered into GF concrete syntax. SUMO permits language generation for up to 10 languages, but the templates were lacking with respect to morphological realisation for languages other than English. GF compensates for these deficits and a fraction of the English CNL generated was ported to both French and Romanian. Other work in this context involves multilingual generation from a knowledge base within the cultural heritage domain. The authors in [Dannélls et al., 2012], describe their objective of trying to build an ontology based multilingual application for museum information on the Web. The Museum Reason-able View application integrates independent data sets, which are used as a single body of knowledge for reasoning and querying evaluation. The knowledge base is built and stored using BIGOWLim [Bishop et al., 2011] as a set a of RDF triples - each resource is linked to a corresponding lexical unit in a GF lexicon. In addition, the Oxford English Dictionary and an LMF (Lexical Markup Framework) [Francopoulo et al., 2006] version of the Swedish Association Lexicon (SALDO) [Borin et al., 2008] were also imported into GF. The work is prototypical and the primary goal is to generate Wikipedia like articles in five languages. Although GF has no specific CNL, one could argue the its growing open source community may result in GF becoming the de-facto open source general framework for developing resources for engineering multilingual CNLs and in that sense it is quite similar to GATE [Cunningham et al., 2002].

3.1.4 A note on Design Principles for CNLs

With respect to designing CNLs, Kuhn in his doctoral thesis describes four principles of design which are generally accepted [Kuhn, 2010a]. They are described as:

- **Clearness**, whereby all statements in a CNL should have a clear meaning. Meaning with respect to a CNL should be described in a systematic and coherent manner and should ideally be mapped into some formal semantics or formal logic. All (or as much as possible) ambiguity should be eliminated from the CNL.

- **Naturalness** implies that statements within a CNL should be as close to natural language as possible. CNLs statements should be acceptable and intuitive to native speakers of the target controlled language. Understandability of a CNL with respect to the user should be aligned to the defined goals or meaning of the CNL.
- **Simplicity** means that a CNL should be easy to describe, teach and learn. Valid CNL statements should be easy to detect and discern by both human and machine.
- **Expressivity** indicates that ideally a CNL should have as broad a coverage as possible for the target domain. The CNL should cover as many possible situations and problems with respect to the domain.

As aptly noted by Kuhn, no CNL can truly satisfy all four principles equally and all four are often in conflict with each other. Clearness may conflict with naturalness so a natural language sentence may be unambiguous to the user but ambiguous or *unclear* to the CNL processor. Conversely, by restricting the statement in order to remove ambiguity for the machine, the statement may seem less natural to the user (For a more in depth discussion on the four design principles, we refer the reader to [Kuhn, 2010a]). The most obvious orthogonal relation is between expressivity and simplicity, whereby the more semantic coverage the developer tries to build into the CNL, she runs the risk of having a negative impact on usability and uptake of the CNL. Much of the CNL work in this thesis is focused on attempting to uncover the right balance between clearness and expressivity.

3.1.5 CNLs -Related Work

With respect to ontology driven driven generation of CNLs or *conceptual authoring*, a well-known implementation which employs the use of NLG to aid the knowledge creation process is **WYSIWYM - What you see is what you meant**) [Power et al., 1998b]. It involves direct knowledge editing with natural language directed feedback. A domain expert can edit a knowledge based reliably by interacting with natural language menu

choices and the subsequently generated natural language feedback which can then be extended or re-edited using the menu options.

Similar to WYSIWYM, **GINO** (Guided Input Natural Language Ontology Editor) provides a guided, controlled NLI (natural language interface) for domain-independent ontology editing for the Semantic Web. GINO incrementally parses the input not only to warn the user as soon as possible about errors but also to offer the user (through the GUI) suggested completions of words and sentences—similarly to the “code assist” feature of Eclipse⁵⁷ with respect to morphological realisation and other development environments [Bernstein and Kaufmann, 2006]. GINO translates the completed sentence into triples (for altering the ontology) or SPARQL⁵⁸ queries and passes them to the Jena Semantic Web framework⁵⁹. Although the guided interface facilitates input, the sentences are quite verbose and do not allow for aggregation. Static grammar rules exist for the controlled language but in addition, dynamic grammar rules are generated from the ontology itself, however this does not constitute surface realisation in the context of natural language generation, but the amendment of additional parsing rules to GINO’s grammar to guide the user. This permits the system to handle a domain shift, however this is heavily dependent on any linguistic data or RDF label data encoded the ontology.

Finally, [Namgoong and Kim, 2007] presents an ontology based CNL editor, similar to GINO, which uses a CFG (Context-free grammar) with lexical dependencies - CFG-DL to generate RDF triples. To our knowledge the system ports only to RDF and does not cater for other ontology languages.

Other related work involves the application of CNL to knowledge base querying, which represent a different task than that of knowledge creation and editing but is worth mentioning for completeness sake. Most notably *AquaLog*⁶⁰ is an ontology-driven, portable

⁵⁷<http://www.eclipse.org/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁵⁸<http://www.w3.org/TR/rdf-sparql-query/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁵⁹<http://jena.apache.org/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁶⁰<http://kmi.open.ac.uk/technologies/aqualog/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

question-answering (QA) system built for the goal of providing a natural language query interface to semantic mark-up stored in knowledge base. By using tools from the GATE framework [Cunningham et al., 2002], AquaLog translates controlled natural language queries into a triple representation called Query-Triples. GATE provides a set of linguistic annotations and additional JAPE (Java Annotation Pattern Engine) grammars are created in order to perform shallow parsing across the user’s questions. However the system is implemented using a simple triple-based intermediate representation as opposed to DRS (discourse representation structure) or fully formed FOPL (first-order predicate logic). Further processing is conducted by the Relation Similarity Service (RSS) module. The role of the RSS module is crucial as it maps CNL questions into ontology compliant queries. It structurally checks the generated query triples against the underlying ontology. If the RSS fails to discover matching relations or concepts within the KB, it requests, as last resort, the user to disambiguate the relation or concept from a given set of candidates. Furthermore this module invokes the use of string similarity metrics, lexical resources such as WordNet [Fellbaum, 1998] and domain-dependent lexicons in order to generate query-triples that are compliant with the underlying ontology [Lopez and Motta, 2004]. However, existing linguistic rules pose difficulties with respect to complex queries, requiring extension of the NLP component, which the authors plan to remedy in future work. An additional key limitation of AquaLog is that only a single ontology can be used at a time. This is addressed in PowerAqua [Lopez et al., 2006], which aims to find answers from distributed, ontology-based semantic markup. PowerAqua [Lopez et al., 2006] extends AquaLog, allowing for an open domain question-answering for the semantic web. The system dynamically locates and combines information from multiple domains.

3.2 Semantic Annotation

The process of providing semantic data is called Semantic Annotation, which involves the embellishment of existing data, i.e. the text, with semantic metadata, which can subse-

quently describe the associated text. In order for the Semantic Web to become a reality, we need, as a *primer inter pares*⁶¹, semantic data. We describe Semantic Annotation as “a process as well as the outcome of the process”. Hence it describes i) “the process of adding semantic data or metadata to content given an agreed ontology and ii) it describes the semantic data or metadata itself as a result of this process” [Handschuh, 2005]. Of particular importance here is the notion of the *addition or association* of semantic metadata to *content*. Semantic Annotation can be categorised into (i) manual semantic annotation and (ii) semi-(automatic) annotation. With respect to the modifier *semi-automatic*, we surround *automatic* with ‘(’ ‘)’, as annotation is rarely *fully automatic*. In fact semi-automatic and automatic annotation systems exist on a continuum and in general most systems which claim to be automatic require some form of human intervention in order to adapt and maintain the system. This largely a consequence of the underlining IE technology which drives semi-(automatic) annotation. In the next sections, we will discuss annotation frameworks in brief review manual and semi-(automatic) annotation systems, and finally we conclude with related applications of Semantic Annotation.

3.2.1 Annotation Frameworks

Annotea is a W3C⁶² LEAD(Live Early Adoption and Demonstration)⁶³ project under the remit of the Semantic Web Advanced Development (SWAD) ⁶⁴ [Kahan et al., 2001, Koivunen, 2005]. It describes an infrastructure for collaborative annotation of Web documents via shared metadata. Aside from semantic annotations, shared bookmarks can also be associated to documents to aid users in organising their documents under different topics. As a W3C project, it naturally has an emphasis on open standards in order to promote interoperability and extensibility. The main metadata format is RDF based

⁶¹Latin to English translation: ‘first among his/her equals’

⁶²<http://www.w3.org/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁶³<http://www.w3.org/2001/Annotea/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁶⁴<http://www.w3.org/2000/01/sw/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

annotation schema, however the document formats for annotation are limited to XML and HTML. Annotea relies heavily on XPointer⁶⁵ for locating and referring annotations in a document and annotations are kept separate from the document - a form of *stand-off markup*. XPointer is the W3C recommendation for identifying fragments in URIs. As annotations are kept separate from content. The benefit of the XPointer approach is that it supports heterogeneous document formats. In principle, XPointer can handle any robust changes to the document as long as the portion of text it is anchored too is not interfered with, however major changes to the document will disrupt Xpointer reference. This can be problematic as naturally both the document and its annotations will evolve over time.

Annotea promotes a semi-formal approach, whereby annotations may be free text comments about documents but these comments must have RDF metadata such as author, creation time etc. The metadata is typed according to the RDF schema. So while, Annotea may promote user uptake in that users are shielded to a certain degree from semantic web ontologies, the approach results in metadata loss with respect to the machine processability of the annotations. The Annotea specification was adopted by tools below such as Amaya, Annozilla and Vannotea (See Section 3.2.2), however to our knowledge there has been little activity in the project since 2003.

CREAM is an annotation framework which takes into account the context of annotations as well as the format of annotations [Handschuh et al., 2003a]. It specifies the components of an annotation system including: annotations' interface support, document format support, support for automation, document management and annotation inference servers. As with Annotea, CREAM adheres to W3C standards such as RDF, OWL and Xpointer for referencing annotations. The disadvantage is that the usage of Xpointer, (as with Annotea) restricts CREAM to XML and HTML input formats. However, the user does have the option to store annotations separately or embed them into the document. CREAM was novel in that it also took into consideration annotation for the

⁶⁵<http://www.w3.org/TR/xptr/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

deep web, which involves annotating databases based on server-side Web page markup according to the database's information structures (produced by the database owner) [Handschuh et al., 2003b]. The annotator produces client-side annotations that conform to the client ontology and the server-side markup as well as publishing the ontology and mapping rules to the database schema. Consequently, based on the mapping rules from ontology to database (derived from the annotations), a user can query the database via the published ontology. Annotation of the *deep web* is an important step as it has implications for lifting of legacy corporate data into the Semantic Web. Finally, CREAM also supports the annotation of relation metadata which is essential for building knowledge bases. S-CREAM and M-OntoMat-Annotizer are both tools based on the CREAM framework (see Section 3.2.2).

3.2.2 Manual Semantic Annotation

Most basic annotation tools permit users to create manual annotations. While they are closely related to purely textual annotation tools, an ontology is interwoven into the annotation process in some form or another. Many of these tools produce Annotea RDF markup. For instance the W3C web browser and editor **Amaya** allows for the mark up of either HTML or XML Web documents [Quint and Vatton, 1997]. Users are able to annotate and edit text using the same tool. It is mainly an authoring tool with browsing facilities. It does not facilitate any semi-automatic annotation, but has support for RDF and XPointer and Xlink⁶⁶ as well as collaborative annotation. Annozilla⁶⁷ is a browser, which supports the Mozilla browser with readable Amaya annotations. The intention of Annozilla is to use Mozilla's native facilities to manipulate annotation data - by using its built-in RDF handling to parse the annotations, and `nsIXmlHttpRequest` to submit data when creating annotations. The project seeks to make Amaya inter-operable with Mozilla. The last activity to our knowledge on the project website appears to be June 2009. **Mangrove** is an-

⁶⁶<http://www.w3.org/TR/xlink11/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁶⁷<http://annozilla.mozdev.org/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

other manual annotation tool which is aimed at user friendliness [McDowell et al., 2003]. Mangrove seeks to “entice” users to annotate HTML based on data generated from a number of semantic services such as calendar events and departmental contact information. It is a simple GUI which allows users to link a selection of tags as they highlight the text. Developments in [McDowell et al., 2004] involved the integration of Mangrove into semantic email processes and represents early steps towards *semantic email* [Scerri, 2010]. **Vannotea** seeks to bring semantic annotation to the next stage of evolution beyond by enabling users to mark up videos, images and audio [Schroeter et al., 2003]. It permits users to add MPEG-2 metadata to video, JPEG2000 to images and Direct 3D to regions of images or *mesh* files. It allows input from distributed users and for example has been deployed as collaboration annotation tool for the cultural heritage domain allowing both museum curators and indigenous groups to annotate cultural artefacts. **Co-Annotea** builds on Vannotea and permits users to annotate multiple mixed-media objects the relationships between them and enables fast and efficient ontology based tagging and correlation of multimedia collections [Hunter and Schroeter, 2008]. It seeks to address the need for to share and compare interpretations and associations across the community over distributed annotation servers. Other work with respect to Vannotea, involves embedding synchronous annotations attachments [Schroeter et al., 2006], using Jabber⁶⁸ message conferencing as well as secure collaborative annotation via *Shibboleth* identity management⁶⁹ and XACML⁷⁰ access policies [Lorch et al., 2003]. In summary, although some manual annotation tools are no longer actively supported, the Annotea model is emerging as a de-facto standard for modelling annotations and tags and has been adopted by many tools [Hunter and Schroeter, 2008].

⁶⁸<http://www.jabber.org/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁶⁹<http://shibboleth.net/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁷⁰eXtensible Access Control Markup Language

3.2.3 Semi-Automatic Semantic Annotation

Many annotation tools involve some degree of semi-automatic annotation services. The **OntoMat** Annotizer is an example of a manual annotation tool and an instance of the CREAM framework described earlier. It provides user friendly functionality for marking up and annotating web pages in a highlight, drag and drop fashion. OntoMat was extended for semi-automatic support. The extensions were called S-CREAM [Handschuh et al., 2002] and included to the adaptive IE system - Amilcare, whereby, based on initial user annotations, the system attempts to automatically suggest new annotations [Ciravegna and Wilks, 2003]. OntoMat allows deep annotation i.e. annotation web pages generated from a database [Handschuh et al., 2003b]. Extensions to OntoMat included **M-OntoMat Annotizer**, which extends the annotation tool for manual annotation of multimedia formats such as video and images [Bloehdorn et al., 2005]. It was designed for users with little multimedia experience and provides automatic extraction of low level features to describe objects within multimedia content. Furthermore, OntoMat was extended with PANKOW (Pattern based Annotation through Knowledge On the Web) functionality [Cimiano et al., 2004]. PANKOW take proper nouns from the IE phase and generates hypothesis phrases based on linguistic patterns specified in the ontology [Hearst, 1992]. So *Brian is a Researcher* would be generated as a pattern, where both ‘Brian’ and ‘Researcher’ are concepts in the ontology. The phrases are sent to a Google web service whereby phrases with the highest count are used to annotate the text with the appropriate concept. The PANKOW process follows the principle of “disambiguation by maximal evidence” [Cimiano et al., 2004] and a similar approach is taken by the system Armadillo (See Section 3.2.4).

The **SHOE (Simple HTML Ontology Extensions) Knowledge Annotator** was an early system for marking up HTML pages with SHOE resources, guided by either locally or remotely available ontologies [Heflin and Hendler, 2001]. Users are prompted for input by the annotator. **Running SHOE** allowed users to build wrappers for automatic annotation of webpages by permitting them to create lists for entities as well as regular

expressions [Heflin and Hendler, 2001]. SMORE⁷¹ is another annotation tool that allowed users to markup web documents with limited knowledge of OWL terms and syntax. It has facilities for ontology creation and authoring and the goal was to provide a flexible environment to create simple web pages simultaneously with markup. It also provides an editor which uses domain and range restrictions to present a list of eligible targets for all object property links from an instance. Development, however, with respect to both tools appear to be inactive since 2005. The COHSE Annotator is an example of another tool which produces annotations that were compliant with Annotea standards [Goble et al., 2003], though annotations are conceived as hyperlinks stored using a Distributed Links Service. The only automation reported was a word matching service for ontology terms. The annotator is deployed as a plugin for both Mozilla and Internet Explorer browsers. COHSE had a number of domain applications including supporting visually impaired users [Goble et al., 2003] and a Java Tutorial site.

3.2.4 Automatic Semantic Annotation

Automatic annotation systems are tools that provide automatic suggestions for annotations. They may require some intervention by knowledge workers or alternatively acquire annotations on a large scale. Some automatic tools must be developed by specialists while others are targeted to knowledge workers [Uren et al., 2006]. These tools factor in user interface design to reduce intrusiveness while maintaining high accuracy. They may be *supervised* or *unsupervised*. Supervised approaches require sample annotations as input for learning algorithms. The problem is that such approaches may require large samples of quality training data which is error prone and labour intensive. Unsupervised approaches attempt to tackle this problem by reducing or eliminating the required amount of training data. However, they must be implemented by specialists.

Lixto is an IE system which requires users to define wrapper rules for mapping un-

⁷¹SMORE: Semantic Markup, Ontology and RDF Editor, See <http://www.mindswap.org/2005/SMORE/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

structured resources in structured facts [Baumgartner et al., 2001]. Users can visually build wrappers by selecting the appropriate pieces for information. The system is distributed as part of the Lixto Software GmbH⁷² and was originally developed by the Technical University of Vienna.

Armadillo is an annotation tool which permits unsupervised creation of knowledge bases from large repositories such as the Web or internal legacy document repositories [Dingli et al., 2003]. The system bootstraps learning from seed examples and exploits redundant information from repositories. So web services such as Google are used to query the validity of an entity and verify its correctness. This is a similar approach to the PANKOW algorithm, described earlier [Cimiano et al., 2004]. Seeds are derived from the repository. An adaptive IE strategy is applied to generalise over examples and to uncover new facts. The documents as well as other sources are exploited to assess and confirm the quality of the annotations. No manual annotations are required. Once confirmed, a new cycle of learning can be initiated. Bootstrapping can be repeated until the user is satisfied with the quality of learning. Armadillo also uses keyword based search techniques in addition to adaptive IE. Manual patterns are also applied to the named entity extraction task. Similar to both PANKOW and Armadillo is **OntoSyphon** [McDowell and Cafarella, 2006], which is not just ontology based but *ontology driven*. Instead of trying to learn all possible information from the document, the systems focuses on parts the ontology and seeks to learn all possible information about the ontology from the Web. In addition, as opposed to just counting hits, the ontology is also taken into account when verifying content.

MnM is a system designed for creating training data for IE systems rather the direct annotations [Vargas-Vera et al., 2002]. So the mark up created by the system is not stored in RDF format. It provides for a user interface, ontology support and open APIs for linking to ontology servers and permits the integration information extraction tools. Users manually annotate the corpus and then feed the data into rules in order to induce

⁷²<http://www.lixt.com/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

wrappers. Rule writing takes a lazy-NLP approach, whereby rules are generated based on conjunction conditions of adjacent words. Rules can be corrected by performing an insertion which causes the tags to shift location based on the training information.

Melita is a user centric automatic annotation system [Ciravegna et al., 2002]. It has two annotation strategies available to the user. The first strategy uses adaptive IE based on the Amilcare system, which is a machine learning system that adapts the classification of the annotations based on user interaction [Ciravegna and Wilks, 2003]. Consequently, a large manual annotation effort is required at the initial stages, which over time as the system learns, is reduced to verification of suggested annotations. The second strategy is rule based, whereby Melita permits more specialist users to write rules based on regular expressions allowing more customisation. Interestingly documents are not selected randomly by the tool but are suggested to the user based on their expected usefulness. Amilcare was also incorporated into the **K@** system which is a legal knowledge management system with RDF semantic capabilities [Gilardoni et al., 2005].

KnowItAll is quite similar to Armadillo (discussed earlier) [Etzioni et al., 2005]. It differs however with respect to assessing the plausibility and quality of candidate extractions. It applies the PMI(Pointwise Mutual Information) measure rather the weighting the candidate based on evidence from multiple sources. PMI is used between words and phrases, estimated from web search engine hit counts and functions in a similar manner to Turney’s PMI-IR algorithm [Turney, 2001]. For example, suppose that the extraction stage proposes “Liege” as the name of a city. If the PMI between “Liege” and a phrase like ”city of Liege” is high, this gives evidence that ”Liege” is indeed a valid instance of the class City. KnowItall is does not require an initial set of seeds to bootstrap the system. Additional extensions to the system include, pattern learning, subclass extraction and list extraction, all of which have contributed to improving performance [Etzioni et al., 2005].

The **SmartWeb** project also investigated unsupervised approaches by using the ontology class and subclass labels to construct examples for learning. So instances were identifiable based on the contexts learned from the initial examples [Buitelaar and Ramaka, 2005].

AeroSWARM⁷³, is an automatic annotation tool which uses OWL ontologies based on those used by the annotator, AeroDAML [Kogut et al., 2001]. A client server version and a web based demonstrator are available, whereby the user supplies a URI and the system returns a file of annotations. However, the user must save the RDF annotations to an annotation server and view the results in an annotation friendly browser such as Amaya. Its uses AeroText for information extraction. The AeroText API used the back-end ontology as a guide to map extracted facts to RDF. The default ontology consists of two parts: The upper level uses Wordnet [Oram, 2001], while the lower level uses a knowledge base provided by the AeroText system [Kogut et al., 2001]

SemTag is an automatic semantic annotation tool based on IBM's text analysis platform Seeker [Dill et al., 2003]. Its entity extraction is based on similarity functions and factors in the contexts similar to previously provided examples. It attempts to tackle entity disambiguation by using a taxonomy based disambiguation technique. The taxonomy covers a range popular items such as movies, music, authors, sports health etc. SemTag is intended to bootstrap an annotation platform, however the tool is intended for specialists as opposed to knowledge workers as such. Semtag operates in three phases (i) Spotting, (ii) Learning and (iii) Tagging. The Spotting pass tokenises input and attempts to match tokens with labels from the ontology, while the learning pass examines a sample corpus in order to find corpus-wide distribution of terms for each node in the taxonomy. Finally the tagging pass is executed, which scans all windows from the learning pass and attempts to disambiguate matches.

KIM (Knowledge and Information Management) [Popov et al., 2004] is a automatic semantic annotation platform based on the GATE [Cunningham, 2002]. KIM takes a knowledge base approach for the extraction task, whereby the system is based on a combination of gazetteers and partial parsing over annotations. Named entities in KIM were initially embellished with semantic metadata from the KIM ontologies. Annotations are quite generic such as persons, places and locations similar to the GATE named entity

⁷³<http://ubot.lockheedmartin.com/ubot/hotdaml/aeroswarm.html>, now inactive

schema. KIM uses the PROTON⁷⁴ ontology, which is a lightweight upper-level ontology defining about 300 classes and 100 properties in OWL Light. It provides coverage of the most general concepts, with a focus on named entities (people, locations, organisations) and concrete domains (numbers, dates, etc.). In addition, the platform uses the KIM World Knowledge Base (WKB), which aims to cover the common knowledge that we normally have beyond our cultural context (e.g. country and hobby). Most applications of the KIM require extending the conceptual models with domain ontologies and the underlying knowledge base with domain specific entities and facts. Custom knowledge bases have been engineered in each of the domains where KIM has been applied. PROTON can also be aligned to DBpedia and other linked datasets. KIM pipelines are also capable of generating efficient in-memory lookup of entities based on the Linked Data Cloud as well as enrichment of semantic annotations with addition related metadata from the Cloud.

Other resources available include MIMIR⁷⁵, which is a semantic search engine that combines a fast full-text index and a high-capacity semantic repository, allowing boolean, SPARQL and annotation pattern searching in a single query [Cunningham et al., 2011]. In addition, KIM is supported by OWLIM [Bishop et al., 2011], which is a family of semantic repositories with the following characteristics: native RDF engines, implemented in Java delivering full performance through both Sesame and Jena, robust support for the semantics of RDFS, OWL 2 RL and OWL 2 QL. OWLIM claims the best scalability, loading and query evaluation performance. Aside from KIM, other GATE based semantic annotation systems include MUSE [Maynard, 2003] and h-Techsight [Maynard et al., 2005] which used GATE to inform the users of the dynamics of concepts and instances, which aid in the manual evolution of the ontology.

DBpedia Spotlight is a tool for annotating mentions of DBpedia resources in text [Mendes et al., 2011]. It attempts to link unstructured information sources to the Linked Open Data cloud through DBpedia. DBpedia Spotlight performs named entity extraction,

⁷⁴<http://proton.semanticweb.org/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁷⁵<http://gate.ac.uk/mimir/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

including entity detection and name entity (or identity) resolution. Although the system claims to use information extraction techniques, it is heavily reliant on dictionary lookup, however the entities' lists that are regenerated from DBpedia appear to disambiguate very well. Information retrieval techniques are used to disambiguate mentions based on neighbouring DPBepdia resource mentions of DBpedia resources. The system does not appear to cater for coreference of mentions, unlike GATE or KIM. It is based on the Lingpipe tools, specifically the Exact Dictionary Based Chunker ⁷⁶ which relies on the Aho-Corasick string matching algorithm [Aho and Corasick, 1975]. Spotlight is novel in that it is an excellent example of leveraging evolving linked data derived from the Web (i.e Wikipedia) for semantic annotation.

3.2.5 Other Types of Annotation

Social annotation systems such as Delicious⁷⁷ and Flickr⁷⁸ have laid the fundamentals for Web 2.0. They are extremely popular among web users due to the fact that the underlying annotation model is quite simple, usually consisting of a tag and a resource [Andrews et al., 2012]. However, due to the underlying natural language nature of the annotation model, social annotation systems are often criticised for not taking into account the explicit semantics of each tag. Problems such as synonymy, homonymy and morphological variance are not catered for and there is ambiguity over the interpretation of a tag i.e. whether *John* is a person or a photographer. Despite these limitations, such systems have had massive user uptake with Yahoo reporting in June 2011 that Flickr had a total of 51 million registered members and 80 million unique visitors⁷⁹ . In August 2011, the site reported that it was hosting more than 6 billion images and this number

⁷⁶[Alias-i.LingPipe4.0.0.http://alias-i.com/lingpipe](http://alias-i.LingPipe4.0.0.http://alias-i.com/lingpipe), Retrieved on 24.08.2010, 2008

⁷⁷<https://delicious.com/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁷⁸<http://www.flickr.com/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁷⁹<http://advertising.yahoo.com/article/flickr.html>, "Flickr - Advertising Solution". Yahoo!. After June 2011. Retrieved 20 September 2011.

continues to grow steadily according to reporting sources⁸⁰. However, formal semantics are sacrificed for usability, the result being that such systems have basic search and retrieval services [Andrews et al., 2012]. Examples of such systems include Youtube⁸¹, Last.fm⁸² and LiveJournal⁸³. We note that the Subject-Predicate-Object (SPO) model, used in semantic annotation systems is of higher structural complexity than *tags*. So a resource can have relation in the form of attributes such as “location”, “start date” and “event” all of which have values which are not resources. Most social networks such as MySpace and Facebook view the user as the resource or subject and use an attribute annotation model to represent profile data. So these systems do use some type of SPO annotation model, however the resource is not aligned to any formal conceptual model or ontology. Relations take this a step further where the annotator is expected to deal with more than one resource as a subject, hence the annotator is expected to bear a higher mental load [Andrews et al., 2012]. Examples include Facebook, MySpace, BibSonomy⁸⁴ and Upcoming.org⁸⁵. Relations are typed links between resources and a user can navigate from one resource to another via these relation links.

Other applications which involve semantic annotation are **Semantic Authoring** applications. Although these applications perform semantic annotation, the primary task is to semantically enable the authoring environment and assist in authoring the publication using previously captured annotations i.e. the tools apply annotation templates for reuse when authoring or retrieving additional information as well as tag based content visualisation. Such tools have been applied to MS Word as well L^AT_EX. They differ however with respect to their conceptual modelling. Some target domain knowledge

⁸⁰<http://news.softpedia.com/news/Flickr-Boasts-6-Billion-Photo-Uploads-215380.shtml>, Accessed Sun 27 Jan 2013 16:52:22 GMT

⁸¹www.youtube.com, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁸²<http://www.last.fm/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁸³<http://www.livejournal.com/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁸⁴<http://www.bibsonomy.org/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁸⁵<http://upcoming.yahoo.com/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

[Carr et al., 2004, Tallis, 2003] while others target rhetorical and argumentation structures and make them explicit for scientific publications i.e. SALT - Semantically Annotated L^AT_EX[Groza et al., 2011]. A thorough review of this field is beyond the scope of this thesis, but we refer the reader to [Groza, 2012], for emerging trends in Semantic Authoring.

Semantic wikis, which have become a somewhat popular way of adding semantics to user generated wiki pages. The term Semantic Wiki often implies either ontology authoring or the semantic annotation of wiki content. A traditional wiki creates links between pages without defining the kind of linkage between pages. Semantic Media Wiki [Krötzsch et al., 2006] allows a user to define the links semantically, thereby adding meaning to links between pages. Each concept or an instance has a page in Semantic Media Wiki, and each outgoing link from the page is annotated with well-defined properties as links. Relational meta-data represented in a Semantic Media Wiki always has the corresponding page as its subject. Other flavours of semantic wikis include IkeWiki⁸⁶ [Schaffert, 2006] and KiWi⁸⁷.

3.3 Conclusions

In Section 3.1, we provided a brief history of Controlled Natural Languages, while in Section 3.1.1 and Section 3.1.2, we reviewed the existing work wrt CNLs for the Semantic Web. Finally, Section 3.1.5 summarised additional related work concerning CNL. With respect to CNLs for the Semantic Web, we can make the following observations:

- Grammatical Framework, (GF) appears to be gaining momentum in the CNL research community. It is possible that GF, may take on the role of a general architecture for developing controlled languages. Furthermore, research within the CNL community is turning its attention towards multilingual controlled languages, with recent efforts to generate ACE, using GF, for several european languages.

⁸⁶<http://ikewiki.salzburgresearch.at/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁸⁷<http://www.kiwi-project.eu/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

- There has been an increased tendency towards conducting proper user evaluation for CNLs. Recall, that a CNL is type of Natural Language Interface, (NLI). In general, the tendency in the CNL research community has been to focus on how much natural language or knowledge to model in the CNL under construction, so much so that research on usability has suffered. The introduction of more complex knowledge modelling such as rules and axioms for instance can have an impact on the design of the CNL and its subsequent usability. One could argue, as does Smart et al [Smart, shed], that user could become “lost in logic”, in that the resulting complex CNL becomes no more user friendly than the ontology editor it is trying to replace. While some CNL researchers have conducted task based evaluations, there have been little or no comparative evaluations across tools. The evaluation of CLOnE against a well know ontology editor - Protégé in Chapter 4, is arguably one of the first concrete task based comparative evaluations between a CNL and an ontology editor. In general, the CNL community should invest more in conducting strong user evaluations and not to lose track of the end goal - the creation of more user friendly ontology editing interfaces.
- A major question is whether a CNL is appropriate for the task? Although, in the context of ontology authoring, CNLs like CLOnE and ACE offer an attractive alternative to ontology editors, we argue that a CNL is not a panacea for resolving ambiguity when processing natural language. This is particularly true, with respect to authoring fluid natural language texts *completely* in CNL such as technical or clinical documentation. We argue that for these scenarios, there should be a pre-existing use case for a *human orientated* CNL, in other words a restricted vocabulary or syntax for a technical domain either legal, clinical or aeronautics such as ASD Simplified Technical English⁸⁸. Without such a use case (despite it being possible to adapt a human-orientated CNL to a machine processable CNL), there would be little incentive for users to interact with it. Factors to be taken into account

⁸⁸<http://www.asd-ste100.org/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

when designing CNLs include, the knowledge creation task complexity, target user (specialist or non expert), the domain (open or specific), available corpora, sample texts, pre-existing language resources or vocabularies, ontologies, multilingualism, requirements for language generation capabilities, and finally, availability of an NLP engineer or computational linguist for development of general purpose CNLs.

- Other issues include whether to adopt a shallow or deeper NLP approach? CLOnE and RABBIT [Hart et al., 2008] are based on a suite of shallow linguistic analysis tools while Grammatical Framework (GF) [Angelov and Ranta, 2009] and Attempto Controlled English (ACE) [Fuchs and Schwitter, 1996a] are more lexicalised. Furthermore, they are both more powerful with respect to knowledge modelling. Both GF and ACE are bidirectional, which is extremely useful for surface realisation. In addition, GF, which is based on the functional language paradigm, can exploit subsumption for free and moreover has an exhaustive bank of application grammars for multiple languages. ACE on the other hand is logic based and has built-in discourse representation structures which are unification based. However, both RABBIT and CLOnE, respectively, as GATE applications, have a number of Semantic Web and Linked Data processing resources available as GATE resources [Cunningham et al., 2002]. This would permit the two CNLs to expose semantic data from text, whereas knowledge encoded in GF or ACE is represented in either of their respective functional or logic based intermediate structures. Consequently, the knowledge must be exported to Semantic Web languages for further exploitation. In addition, Semantic Web ontologies cannot easily be imported and exploited by GF or ACE. In summary, deciding on what CNL or tools to use depends very much on the complexity of both the knowledge creation task and the language modelling task of the CNL as well as the target knowledge representation language and whether there is a need to reuse existing ontologies or vocabularies.
- As research into controlled languages has been invigorated to a certain degree

by the Semantic Web initiative, Semantic Web researchers should observe lessons learned by previous work in designing CNLs. Corpus analysis and empirical approaches should be a necessary step when designing a CNL [Grover et al., 2000, Mitamura, 1999]

In Section 3.2, we moved to a definition of Semantic Annotation, while Section 3.2.1, discussed abstract frameworks for annotation that are still valid today. Section 3.2.2, Section 3.2.3 and Section 3.2.4 surveyed manual, semi-automatic and automatic annotation tools respectively, while in Section 3.2.5, we explored briefly Social Annotation and Semantic Annotation in the context of Semantic Authoring and finally Semantic Wikis. With respect to Semantic Annotation, we make the following observations:

- In general across the literature, up-to-date surveys with respect to Semantic Annotation are lacking. A comprehensive review of the state of the art of the field is needed.
- While there are plethora of evaluation tools available, there appears to be an absence in general with respect to quantitative user interface studies of manual or semi automatic annotation tools.
- Annotation ontologies should be revisited in particular with the current focus on entity linking within the Linked Data community. Annotation is more sophisticated than linking an entity mention in text to a URI i.e. it may involve the association of richer metadata to an entity mention.
- Arguably, the success of the Linked Open Data initiative may reinvigorate interest in the field of Semantic Annotation, whereby existing research on Semantic Annotation may provide further insights to the entity linking problems with respect to Linked Data.

The purpose of this chapter has been to review related work with respect to Controlled Natural Languages and Semantic Annotation. We have provided the necessary

background and clarified the state of the art with respect to both fields. We move now to describe the CNL - CLOnE in the next chapter because it forms the basis for the research contributions of this thesis.

Part III

Core Research

4 CLOnE - Controlled Language for Ontology Editing

This Chapter discusses Controlled Language for Ontology Editing (CLOnE), which forms the basis for the research contributions of this thesis. The CLOnE language and its software were originally developed by the Sheffield Natural Language Processing group at the University of Sheffield. In Chapter 5 and Chapter 6, we extend CLOnE for the purposes of exploring and experimenting with RoundTrip Ontology Authoring and Semantic Annotation respectively. We describe CLOnE, its experimental results and limitations as a prelude to our research contribution.

References: This chapter is based on my contribution for [Funk et al., 2007]. The particular contribution was a survey review of related work with respect to CNLs.

4.1 Introduction

Formal data representation can be a significant deterrent for non-expert users or small organisations seeking to create ontologies and subsequently benefit from adopting semantic technologies. Existing ontology authoring tools such as Protégé⁸⁹ attempt to resolve this, but they often require specialist skills in ontology engineering on the part of the user. This is even more problematic for domain specialists, such as clinicians, business analysts, legal experts, etc. As such professionals cannot be expected to train themselves to comprehend Semantic Web formalisms and the process of knowledge gathering; involving both a domain expert and an ontology engineer can be time-consuming and costly. Controlled Natural Languages (CNLs) for knowledge creation and management offer an

⁸⁹<http://protege.stanford.edu>, Accessed, Thu 25 Jul 2013 16:54:32 IST

attractive alternative for naive users wishing to develop small to medium sized ontologies or a first draft ontology which can subsequently post-edited by an Ontology Engineer.

We have already mentioned the Controlled Language for Ontology Editing (CLOnE) which was developed by the Sheffield Natural Language Processing group at the University of Sheffield. In Chapter 5 and Chapter 6, we extend and adapt CLOnE for the purposes of exploring and experimenting with Round Trip Ontology Authoring and Semantic Annotation respectively. Consequently, it is necessary to describe CLOnE, its experimental results and limitations as it forms the basis of our research contribution.

CLOnE is a CNL which allows users to design, create, and manage information spaces without knowledge of complicated standards such as XML, RDF and OWL, or ontology engineering tools such as Protégé. Its implementation, CLIE - Controlled Language Information Extraction, is a simplified natural language processor that allows the specification of logical data for Semantic Web technology purposes in normal language, but at the same time attains the high accuracy levels necessary for high reliability in applications. CLIE is based on GATE's existing cascade of finite state transducers (FSTs) for IE [Cunningham et al., 2002]. CLIE is configured so that it either accepts input as valid (in which case accuracy is generally 100%) or rejects it and will warn the user of needed repairs to the syntax. Because the parsing process is deterministic, the usual IE performance measures (precision and recall) are not relevant.

While the CLOnE language and the CLIE applications are themselves not overly complex in comparison to other CNLs, the main contribution of this work is a *repeated-measures task-based* evaluation of CLOnE/CLIE compared to a standard Ontology Engineering Editor -Protégé. The contribution is important as it represents the first comparative user evaluation between a CNL and an ontology editing tool.

The remainder of this chapter is as follows: Section 4.2 discusses the CLOnE CNL and its implementation CLIE, Section 4.3 describes the user evaluation and discusses our quantitative findings, while Section 4.4 compares CLOnE to related work within the context of user evaluation. Finally, Section 4.5 offers conclusions and as a prelude to

Chapter 5, discusses future work involving language generation and CLOnE .

4.2 Requirements and Implementation

4.2.1 Requirements

Taking into consideration the strengths and weaknesses of other controlled language systems discussed above, the CLOnE language and the implementation CLIE were designed to meet the following requirements:

1. CLIE requires only one programming language or runtime environment, Java 1.5.
2. CLOnE is a sublanguage of English.
3. As far as possible, CLOnE is grammatically lax; in particular it does not matter whether the input is singular or plural (or even in grammatical agreement). For example, the lines of input within each group in Table 4.1 have the same semantics. It is also case-insensitive.
4. CLOnE is compact; the user can create any number of classes or instances in one sentence.
5. CLOnE is easy to learn by following examples and a few guiding rules, without having to study formal expressions of syntax; nonetheless, the basic (declarative) implementation of CLOnE uses only 11 syntactic rules.

Table 4.1: Groups of equivalent sentences in CLOnE

Book is a type of document.	Alice is a person. Bob is a person.
Books is a type of document.	Alice and Bob are person.
Books are types of document.	Alice and Bob are persons.
Books are types of documents.	

6. Any valid sentence of CLOnE can be unambiguously parsed.

CLOnE and CLIE are limited to the to following ontology operations:

- the creation and deletion of new classes and instances
- the creation and deletion of subclasses.
- the creation and deletion of class (and instance) level properties
- and the creation and deletion of datatype properties.

The goal of building CLOnE (and CLIE) was to design a CNL to simplify ontology authoring in that the simplest solution could provide the most useful results with minimal or no training for users. If CLOnE were enhanced with to cover more complex ontology operations in OWL such as rules and axioms or other features such as cardinality and special characteristics for relations (i.e. transitivity or inverse properties), than it may become difficult to both learn and use which would be undesirable.

4.2.2 Implementation

The syntax of the controlled language is based principally on *chunks*, which are used to name classes, instances, properties and values; and *keyphrases*. POS (part-of-speech) tags and morphological analysis (lemmatisation) also play a role.

Procedurally, CLIE's analysis consists of a GATE pipeline of Processing Resources (PRs) as illustrated in Figure 5.1. The pipeline starts with a series of fairly standard NLP tools which (as implemented in GATE) add annotations and annotation features to the document.

These are followed by three PRs developed particularly for parsing CLOnE: a gazetteer of keywords and phrases fixed in the CLOnE syntax and two JAPE⁹⁰ transducers which

⁹⁰JAPE (Java Annotation Pattern Engine) is an applied language used in GATE for with writing regular expressions over annotations and subsequent annotation manipulation of bound patterns using executable Java code.

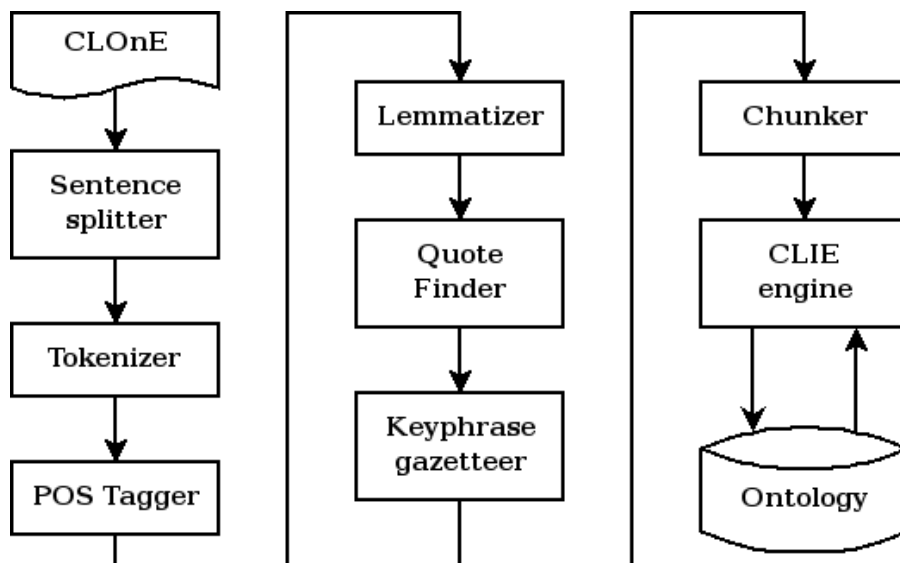


Figure 4.1: The CLIE pipeline

identify quoted and unquoted chunks. (Names enclosed in pairs of single or double quotation marks can include reserved words, punctuation, prepositions and determiners, which are excluded from unquoted chunks in order to keep the syntax unambiguous.)

4.2.3 Syntax and semantics

An input document to CLIE consists of a series of CLOnE sentences. Each each sentence consists of *chunks*, *list separators*, *prepositions*, *determiners* and *keyphrases* and ends with a full stop. A parseable sentence matches the pattern (similar to a regular expression) left-hand side (LHS) of one of the rules in the CLIE JAPE transducer and the sentence's semantics are determined by the rule's right-hand side (RHS), which contains Java code to manipulate the ontology.

We now present the full current list of CLOnE syntactic rules, for each of which is given the following details:

- an informal statement of the pattern, using variables such as CLASS and INSTANCE

(and CLASSES, for example, to indicate a list of one or more);

- one or more examples of CLOnE input in `typewriter` font; and
- an explanation of the rule’s semantics, i.e. what CLIE does when an CLOnE input matches the rule.

Many of the sentences also have negative forms with the keyphrase `Forget that` at the beginning. The negative forms (`Forget`) can be used to correct input errors (a form of “undo”-function) as well as to delete obsolete or erroneous information while editing an existing ontology.

Rule 1. There is/are CLASSES.

`Forget that there is/are CLASSES.`

`There are agents and documents.`

Create a new class immediately under the top class for each chunk in the list. If negated, delete each class named in the list.

Rule 2. INSTANCES0 is a/are CLASS1.

`Forget that INSTANCES0 is a/are CLASS1.`

`‘University of Sheffield’ is a university.`

`Alice Jones and Bob Smith are persons.`

For each chunk in list INSTANCES0, create an instance of CLASS1. If negated, delete each instance. If CLASS1 does not name an existing class, generate an error.

Rule 3. CLASSES0 is/are a type/types of CLASS1.

`Forget that CLASSES0 is/are a type/types of CLASS1.`

`Universities and persons are types of agent. Dogs are a type of mammal.`

`Forget that dogs are a type of cat.`

For each chunk in list CLASSES0, if it already exists as a class, make it a subclass

of the CLASS1 if it does not exist, create a new class as a subclass of CLASS1.⁹¹

If the sentence is negated, unlink the subclass-superclass relationship (but do not delete the subclass).

If class 1 does not name an existing class, generate an error.

Rule 4. CLASSES/INSTANCES0 have CLASSES/INSTANCES1.

Forget that CLASSES/INSTANCES0 have CLASSES/INSTANCES1.

Journals have articles. 'Journal of Knowledge Management' has 'Crossing the Chasm'.

Iterate through the cross-product of chunks in list CLASSES/INSTANCES0 and chunks in list classes/instances1. For each pair, if both are classes, create a property of the form *Domain_has_Range*. If both are instances, find a suitable property and instantiate it with those instances; if there is a class-instance mismatch or a suitable property does not exist, generate an error.

Rule 5. CLASSES0 have DATATYPE DESCRIPTION.

Forget that CLASSES0 have DATATYPE DESCRIPTION.

Projects have string names. Deliverables and conferences have dates as deadlines.

For each class named in list 0, create a datatype property of the form *Domain_has_Description*.

Rule 6. INSTANCE0 has DESCRIPTION with value VALUE.

Forget that INSTANCE0 has DESCRIPTION with value VALUE.

SEKT has name with value 'Semantically-Enabled Knowledge Technology'.
D2.2.2' has deadline with value 'M36'.

For each instance named in list INSTANCE0, find a suitable datatype property with a matching description and instantiate it with the specified data value.

⁹¹CLIE and the GATE Ontology API support multiple inheritance.

Rule 7. CLASS/INSTANCE0 is/are also called/known as SYNONYMS1.

Forget that CLASS/INSTANCE0 is/are also called/known as SYNONYMS1.

Dogs are also called canines. Bob Smith is also called Bob.

Add all the chunks in list SYNONYMS1 as synonyms of the class or instance named by chunk CLASS/INSTANCE0 . The synonyms are indexed and can be used in subsequent statements of CLOnE, although they do not affect the classes' and instances' primary names in the ontology. If negated, delete the synonyms in list SYNONYMS1 from the class or instance in chunk SYNONYMS1.

Synonyms are implemented as RDF-labels so they are saved in the OWL-Lite file that CLIE exports and can be used again when the same file is re-loaded.

Rule 8. There are CLASSES0, which have CLASSES1.

There are projects, which have workpackages and deliverables.

Create a class under the top class for each chunk in list CLASSES0, and create properties of the form *Domain_has_Range* for the cross-product of new classes in list CLASSES1 and existing classes in list CLASSES0.

Rule 9. CLASSES/INSTANCES0 are DESCRIPTION PREPOSITION CLASSES/INSTANCES1.

Forget that CLASSES/INSTANCES0 are DESCRIPTION PREPOSITION CLASSES/INSTANCES1.

Persons are authors of documents. Carl Pollard and Ivan Sag are the authors of 'Head-Driven Phrase Structure Grammar'.

Iterate through the cross-product of chunks in lists CLASSES/INSTANCES0 and CLASSES/INSTANCES1. For each pair, if both name classes, create a property of the form *Domain_Description_Prep_Range*.

If both name instances and a suitable property can be found, instantiate the property between the given instances. If there is a class-instance mismatch or one of the names cannot be dereferenced, an error message is produced.

This rule is a particularly good example of CLIE's use of information from the

ontology to interpret the input sentences.

Rule 10. **Forget everything.**

Clear the whole ontology (and start over).

Rule 11. **Forget CLASSES/INSTANCES.**

Forget projects, journals and 'Crossing the Chasm'.

For each chunk listed, delete the named class or instance. (The GATE ontology API will automatically delete subclasses and instances of named classes, and properties and property definitions referring to named classes and instances.)

4.3 Evaluation

As mentioned earlier, the main contribution of this work is a *repeated-measures task-based* evaluation of the CLOnE system compared to a standard Ontology Engineering Editor-Protégé. The contribution is important as it represents the first solid comparative user evaluation between a CNL and ontology editing tool. It is important to review the methodology and results of this work as it forms the basis for the research contributions of this thesis (See Chapters 5 and 6). For the remainder of this chapter, when we refer to “CLOnE” in our statistical results, we are referring to both the CLOnE CNL and CLIE system simultaneously.

4.3.1 Methodology

We prepared the following documents for the users.

- The pre-test questionnaire asks for background information: how much each subject already knows about ontologies, the Semantic Web, Protégé and controlled languages. We scored this questionnaire by assigning each answer a value from 0 to 2 and dividing the total by 12 to obtain a score of 0–100

- The short manual introduces ontologies and provides “quick start” instructions for both pieces of software. The manual was inspired by Protégé’s *Ontology 101: Creating your First Ontology* documentation [Noy and McGuinness, 2001]. (See Appendix A for details of the manual and CLOnE language and CLIE interface.)
- The post-test questionnaire for each tool is the *System Usability Scale*, which also produces a score of 0–100 [Brooke, 1996].
- We devised a comparative questionnaire to measure each user’s preference for one of the two tools. This form is scored similarly to SUS so that 0 would indicate a total preference for Protégé, 100 would indicate a total preference for CLOnE, and 50 would result from marking all the questions *neutral*. On the reverse side and in discussion with the facilitator, we offered each user the opportunity to provide comments and suggestions.
- We prepared two lists of ontology-editing tasks (Task list A and Task list B). Each task list was divided into three sublists covering the following task types:
 - creating classes and subclasses,
 - creating instances, and
 - creating and defining properties.

We recruited 15 volunteers with varying experience levels and asked each volunteer to complete the pre-test questionnaire, to read the manual, and to carry out each of the two task lists with one of the two tools. Approximately half the users (8 of 15) carried out Task List A with CLOnE and then Task List B with Protégé; the others (7 of 15) carried out A with Protégé and then B with CLOnE.

We measured each user’s time for each task list and in most cases (12 of 15) for each sublist. After each task list we asked the user to complete the SUS questionnaire for the specific tool used, and finally we asked the them to complete the comparative questionnaire.

4.3.2 Background

The methodology constitutes a *repeated-measures, task-based* evaluation: each participant carries out a similar list of tasks on both tools being compared.

We chose the SUS questionnaire as our principal measure of software usability because it is a *de facto* standard in this field. Although it superficially seems subjective and its creator called it “quick and dirty”, it was developed according to the proper techniques for a Likert scale [Brooke, 1996].

Furthermore, researchers at Fidelity Investments carried out a comparative study of SUS, three other published usability questionnaires and an internal questionnaire used at Fidelity, over a population of 123 subjects, to determine the sample sizes required to obtain consistent, accurate results [Tullis and Stetson, 2004]. They found that SUS produced the most reliable results across all sample sizes; they noted a jump in accuracy to 75% at a sample size of 8, but recommended a sample of at least 12–14 subjects. As a reference for interpreting the results, average SUS scores are usually between 65 and 70 [Bailey, 2006].

4.3.3 Hypothesis statements and statistical measures

With respect CLOnE SUS scores we state the our first hypothesis pair as:

4.1H₀: It is predicated that CLOnE usability (SUS) scores will not be greater than neutral ($\mu \not> 65$).

4.1H₁: It is predicated that CLOnE usability (SUS) scores will be greater than neutral ($\mu > 65$) .

To clarify, we are stricter in our interpretation of ‘neutral’ in that only SUS values of >0.65 are considered to be strong SUS scores as they are above the threshold of neutral or weak values [Bailey, 2006].

In addition, with respect to the CLOnE/Protege preference we have the second hypothesis pair:

4.2H₀: It is predicated that the CLOnE/Protégé preference score will not be greater than neutral ($\mu \not\geq 65$)

4.2H₁: It is predicated that the CLOnE/Protégé preference score will be greater than neutral ($\mu > 65$)

Before presenting and interpreting the findings from the experimental data, we briefly explain the statistical measures used in the following sections.

A *z-score* is a standard score which indicates the (signed) number of standard deviations an observation or datum is above the mean [John L. Phillips, 1996].

A *95% confidence interval* calculated from a data sample is a range which is 95% likely to contain the mean score of the whole population which the sample represents [John L. Phillips, 1996].

A *correlation coefficient* over a set of pairs of numeric data is analogous to the appearance of a scatter graph or X-Y plot. +1 signifies a perfect correlation and corresponds to a graph in which all points lie on a straight line with a positive slope; -1 signifies a perfect inverse correlation (the points lie on a straight line with a negative slope); 0 indicate a complete lack of correlation (random distribution of the points). Values $> +0.65$ and < -0.65 are generally considered to indicate strong correlations.

The formula for Pearson's coefficients assumes that the two variables are linearly meaningful; physical measurements such as length and temperature are good examples of such variables. The formula for Spearman's coefficients, on the other hand, stipulates only ordinal significance (ranking) and is often considered more appropriate for subjective measurements (such as many in the social sciences) [Connolly and Sluckin, 1971, Hildebrand et al., 1977, John L. Phillips, 1996, Calder, 1996, Simon, 2006].

4.3.4 Results and Data Analysis

As the descriptive statistics in Table 4.2 show, the SUS scores for CLOnE are generally above the baseline and distributed generally higher than those for Protégé, and scores on the comparative questionnaire are generally favourable to CLOnE. The scores are

Table 4.2: Summary of the questionnaire scores

Measure	min	mean	median	max
Pre-test scores	25	55	58	83
CLOnE SUS rating	65	78	78	93
Protégé SUS rating	20	47	48	78
CLOnE/Protégé preference	43	72	70	93

also broken down according to the tool used and the task list (A or B) carried out and calculate confidence intervals as shown in Table 4.3; these indicate that for each task list and for the combined results, the larger population which our sample represents will also produce mean SUS scores for CLOnE that are both higher than those for Protégé and above the SUS baseline.

Table 4.3: Confidence intervals (95%) for the SUS scores

Task List	Tool	Confidence interval
A	Protégé	33–65
A	CLOnE	75–93
B	Protégé	30–58
B	CLOnE	67–79
A&B	Protégé	37–56
A&B	CLOnE	73–84

With respect to

4.1H₀: It is predicated that CLOnE usability (SUS) scores will not be greater than neutral ($\mu \not\geq 65$),

we can reject the null hypothesis with $z=5$ $p < 0.5$.

With respect to:

4.2H₀: It is predicated that the CLOnE/Protégé preference score will not be greater than neutral ($\mu \not\geq 65$),

even though the preference score is favourable towards CLOnE, we cannot reject the null hypothesis with $z=1$ $p < 0.5$.

For our post-hoc data analysis, we also generated Pearson’s and Spearman’s correlations coefficients from a wide range of data; Table 4.4 shows the highlights of these calculations. In particular, we note the following points.

Table 4.4: Correlation coefficients

Measure	Measure	Pearson’s	Spearman’s
Pre-test	CLOnE time	-0.06	-0.15
Pre-test	Protégé time	-0.13	-0.27
CLOnE time	Protégé time	0.78	0.51
CLOnE SUS	Protégé SUS	-0.31	-0.20
CLOnE SUS	C/P Preference	0.68	0.63
Protégé SUS	C/P Preference	-0.62	-0.63
Pre-test	CLOnE SUS	-0.17	-0.17
Pre-test	Protégé SUS	-0.16	-0.15
CLOnE time	CLOnE SUS	0.26	0.15
Protégé time	Protégé SUS	-0.17	-0.24
CLOnE time	Protégé SUS	0.19	-0.01
Protégé time	CLOnE SUS	0.42	0.44

- The pre-test score has no correlation with task times or SUS results.
- The task times for both tools are moderately correlated with each other but there are no significant correlations between task times and SUS scores, so both tools are technically suitable for carrying out both task lists.

- As expected, the C/P preference score has a moderately strong correlation with the CLOnE SUS score and a moderately strong negative correlation with the Protégé SUS score. (The SUS scores for the two tools also show a weak negative correlation with each other.) These figures confirm the coherence of the questionnaires as a whole.

4.3.5 Sample quality

Although our sample size ($n = 15$) satisfies the requirements for reliable SUS evaluations (as discussed in Section 4.3.2), it is also worthwhile to establish the consistency of the two partitions of our sample, as enumerated in Table 4.5:

by tool order and task-tool assignment: subjects who carried out task list A on CLOnE and then B on Protégé, in comparison with those who carried out A on Protégé then B on CLOnE; and

by sample source: subjects drawn from the GATE development team, in comparison with others.

Table 4.5: Groups of subjects by source and tool order

Source		Tool order		Total
		PC	CP	
G	GATE team	4	5	9
NG	others	4	2	6
Total		8	7	15

Tool order was divided almost evenly among our sample. Although the SUS scores differed slightly according to tool order (as indicated in Table 4.3), the similar task times suggest that task lists A and B required similar effort. We note that the SUS scores for

each tool tend to be slightly lower for task list B than for A, and we suspect this may have resulted from the subjects' waning interest as the evaluation progressed.⁹² But because Table 4.3 in particular shows consistent results between CLOnE and Protégé for each task list, we believe that our study was fair.

One must also consider the possibility of biased subjects drawn from colleagues of the developers and facilitator. As Table 4.5 shows, members of the GATE team constituted 60% of the user sample. The measures summarised in Table 4.6, however, show in particular that although members of group G generally rated their own expertise higher (in the pre-test questionnaire) than those in group NG, both groups produced very similar ranges of SUS scores for each tool and of C/P preferences scores. These measures allay concerns about biased subjects: we conclude that groups G and NG were equally reliable so the sample as a whole is satisfactory for the SUS evaluation.

Table 4.6: Comparison of the two sources of subjects

Measure	Group	min	mean	median	max
Pre-test	G	25	62	67	83
	NG	33	44	42	58
CLOnE SUS	G	65	78	78	90
	NG	65	78	78	93
Protégé SUS	G	25	46	48	70
	NG	20	47	46	78
C/P Preference	G	50	71	68	90
	NG	43	74	79	93

⁹²To eliminate the possibility of this effect with the same reliability, we would need twice as many subjects, each carrying out one task list with one tool (a *between-subject* experiment, in contrast to our *within-subject* experiment).

4.3.6 Subjects' suggestions and comments

The test users made several suggestions about CLOnE.

- Several subjects complained that they needed to spell and type correctly the exact names of the classes and instances, such as “Journal of Knowledge Management” and that CLOnE is intolerant of typos and spelling mistakes. They suggested spell-checking and hinting (as provided in the Eclipse⁹³ UI) to alleviate this cognitive load.
- A related suggestion is to highlight the input text with different colours for classes, instances and properties, perhaps after the user clicks a *Check* button, which would be provided in addition to the *Run* button. The *Check* button could also suggest corrections and give error messages without affecting the ontology.
- Users complained that it was difficult to tell why some input sentences failed to have any effect, because CLOnE does not explicitly indicate unparseable sentences.
- Some suggested that CLOnE should automatically clear the input text box after each run, but they also realised this would make it more difficult to correct errors, because it is currently easy to prefix the keyword **forget** to incorrect sentences from the previous input without having to retype them.
- Some suggested making the *Run* button easier to find and putting the ontology viewer and the input box on the screen at the same time instead of in alternative panes.
- A few users said it would be useful to have an *Undo* button that would simply reverse the last input text, instead of having to **forget** all the sentences individually.

⁹³<http://www.eclipse.org/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

4.3.7 Limitations of the evaluation

There are limitations with respect to the evaluation discussed above. Most notably, the correlations were not tested for significance, hence one cannot infer beyond the sample to the general population. Secondly, although the evaluation addresses bias concerning participants who are members of the GATE team, ideally further evaluations involving CLOnE should have tighter control over bias.

4.4 Related Work

With respect to related work, we will review existing CNL research, but in the context of user evaluation. (For a thorough survey of controlled languages, we refer the reader to Chapter 3). As discussed in Chapter 3, Attempto Controlled English (ACE) is a well known CNL [Fuchs and Schwitter, 1996a]. However, to our knowledge no task based evaluations which compare ACE against another ontology authoring tool have been conducted, however much of ACE evaluation has focused on comprehension of the language itself. Recently Kuhn [Kuhn, 2010b] described an evaluation framework for CNLs based on *Ontographs*. Ontographs are a graphical notation to enable tool independent and reliable evaluation of the human understanding of a given knowledge representation language. The author categorises CNLs evaluations into (1)task-based, whereby users are provided with a specific task to complete and (2) paraphrase-based which are concerned with testing the understandability of the CNL. Ontographs serve as a common basis for testing and comparing the understandability of two different formal languages and facilitate the design of tool-independent and reliable experiments. The author claims that Ontographs are simple and intuitive. They are useful for representing simple logical forms but they do not cater for functions and are restricted to unary and binary predicates. In short, Ontographs serve to test the relative understanding of the core logic for two different formal languages.

In [Engelbrecht et al., 2009], the authors undertake a paraphrase-based evaluation to

assess whether domain experts without any ontology authoring development can author and understand declaration and axiom sentences in Rabbit. The experiment included 21 participants from the ordnance survey domain and a Rabbit language expert. The participants were given a text that describes a fictional world and were asked to make knowledge statements which were then compared to equivalent statements created by the Rabbit expert. The sentences produced by non-experts were analysed for correctness (with regard to the knowledge captured) by independent experts and were compared to those produced by the Rabbit expert. Interestingly, on average 51% of the sentences generated at least one error. Furthermore, the most common error was the omission of the quantifier “every” at the beginning of a sentence. This observation was of statistical significance. Other user errors included: confusing instances with subclass declarations, a tendency to omit intensional information as well difficulties modelling knowledge under the open world assumption. The work of [Hart et al., 2008] and [Engelbrecht et al., 2009] is important in the context of CNL evaluation in that it represents the advent of the paraphrase-based approach to evaluating CNLs.

We discussed ROO [Dimitrova et al., 2008], which was developed by the University of Leeds, is an open source java based plugin to Protege which supports a domain expert to create and edit ontologies using Rabbit. The tool is based on the experiences of the Ordnance Survey(OS) agency with creating large-scale geo-spatial ontologies. An evaluation study of ROO was conducted against ACEView [Kaljurand, 2008] where participants from the domains of geography and environmental studies were asked to create ontologies based on hydrology and environmental models respectively. Both ontology creation tasks were designed to resemble real tasks performed by domain experts at OS. Controls were put in place to eliminate bias and ontologies for both domains were also produced by the OS to compare against the ROO generated ontologies. The quantitative results were favourable. Although ACEView users were more productive (not in the statistically significant sense), they tended to create more errors in the resulting ontologies. Furthermore with respect to ROO users, their understanding of ontology modelling improves

significantly in comparison to ACEView. Interestingly, but not surprisingly, none of the ontologies produced were usable without post editing. Even though one could argue that the user experience is significantly improved with either ACE or ROO over a non CNL supported standard ontology editing tool, users would still require some formal training knowledge modelling techniques to author quality ontologies.

GINO (Guided Input Natural Language Ontology Editor) provides for guided, controlled NLI for domain-independent ontology editing for the Semantic Web. GINO incrementally parses the input not only to warn the user as soon as possible about errors but also to offer the user (through the GUI) suggested completions of words and sentences—similarly to the “code assist” feature of Eclipse and other development environments. GINO translates the completed sentence into triples (for ontology editing) or SPARQL⁹⁴ statements (for queries) and passes them to the Jena Semantic Web framework. (The JENA Eyeball⁹⁵ model checker verifies the OWL for consistency.) Although the guided interface facilitates input, the sentences are quite verbose and do not allow for aggregation [Bernstein and Kaufmann, 2006]. Furthermore, similar, to our evaluation, a small usability evaluation was conducted using SUS - System Usability Scale [Brooke, 1996], however the sample set of six was too small to infer any statistically significant results [Tullis and Stetson, 2004]. In addition, GINO was not compared to any existing Ontology editor during the evaluation. Finally, [Namgoong and Kim, 2007] presents an Ontology based Controlled Natural Language Editor, similar to GINO, which uses a CFG (Context-free grammar) with lexical dependencies - CFG-DL to generate RDF triples. To our knowledge the system ports only to RDF and does not cater for other Ontology languages. Furthermore no quantitative user evaluation is provided.

Finally, with respect to Grammatical Framework [Angelov and Ranta, 2009] although the work is very impressive, to our knowledge there have been no evaluations conducted with respect to the usability or comprehension of CNL applications for GF,

⁹⁴<http://www.w3.org/TR/rdf-sparql-query/>, Accessed, Thu 25 Jul 2013 16:54:32 IST

⁹⁵<http://jena.sourceforge.net/Eyeball/full.html>, Accessed, Thu 25 Jul 2013 16:54:32 IST

rather the evaluation work has been focused towards the machine translation context [España-Bonet et al., 2011].

4.5 Conclusion

In this chapter we have presented the CLOnE language and its implementation - CLIE, as well as empirical evidence to support their combined suitability for basic ontology editing tasks as well its comparable usability to a standard ontology editing tool.

The user evaluation consistently indicated that our subjects found CLOnE/CLIE significantly more usable and preferable than Protégé for the straightforward tasks that we assigned. (Of course we make no claims about the more complicated knowledge engineering work for which Protégé but not CLIE is designed and intended for.). CLIE was designed to create small to medium sized ontologies or an initial first draft of an ontology. The point being that the majority of knowledge in an ontology will be taxonomic as well relationships between classes, leaving the more complex axiom and rules to the ontology engineer. The main contribution of this work is the evaluation of CLOnE/CLIE compared to a standard Ontology Engineering Editor - Protégé, which represents the first comparative user evaluation between a CNL and an ontology editing tool.

The participants made several interesting and useful suggestions for improvements to CLOnE/CLIE many of which we already envisage developing in future work. In particular, the notion of embedding CLOnE or CNL in semantic wikis or other user interfaces which would eliminate some of the constraints imposed by running it in the GATE GUI, which is really intended for developing language engineering applications (such as CLIE itself) rather than for interactively editing language resources (such as documents and ontologies). Other work involves to adaptation of CLOnE for semantic annotation, already mentioned in Section 1.1.2 , but which is explored in detail in Chapter 6. In this chapter, we have discussed using CLOnE to generate ontologies from input text. The reverse of the process involves the generation of CLOnE from an existing on ontology by Natural Language Generation (NLG), specifically shallow NLG. Combining both NLG

and CLIE enables a RoundTrip Ontology Authoring(ROA) environment: a user can start with an existing imported ontology or one originally produced using CLOnE, (re)produce the Controlled Language using the text generator, modify or edit the text as required and subsequently parse the text back into the ontology using the CLOnE environment. The process can be repeated as necessary until the required result is obtained. The purpose of introducing NLG is to ease the learning curve associated with writing CLOnE. On this note, we move to Chapter 5, to explore this further.

5 Round Trip Ontology Authoring

This Chapter discusses the design and implementation of the Round Trip Ontology Authoring (ROA) environment (See Section 5.2). We focus in particular on text generator component (See Section 5.2.3). The chapter provides the first comparative task-based evaluation between a generation driven Controlled Natural Language (CNL) tool and a standard ontology editor - Protégé [Knublauch et al., 2004] and discusses the quantitative findings (See Section 5.3).

References: This chapter is based mainly on [Davis et al., 2008].

5.1 Introduction

The previous chapter presented CLOnE - *Controlled Language for Ontology Editing* which allowed naive users to design, create, and manage information spaces without knowledge of technical standards (such as XML, RDF and OWL) or ontology engineering tools. The CLOnE software - CLIE (Controlled Language IE) is based on GATE's existing tools for IE (Information Extraction) and NLP (Natural Language Processing) (Henceforth, when we refer to CLOnE, we are also including CLIE) [Cunningham et al., 2002].

CLOnE was evaluated using a *task-based* methodology in comparison with a standard ontology editor – Protégé. CLOnE performed favourably with test users in comparison to Protégé. Despite the benefits of applying Controlled Natural Language (CNL) technology to ontology engineering, a frequent criticism against its adoption, is the learning curve associated with following the correct syntactic structures and/or terminology in order to use the CNL properly. Adhering to a controlled language can be, for some non expert

users, time consuming and annoying. These difficulties are related to the *habitability* problem, whereby users do not really know what commands they can or cannot specify to the NLI (Natural Language Interface) [Thompson et al., 2005]. Where the CLIE system uses natural language analysis to unambiguously parse CLOnE in order to create and populate an ontology, the reverse of this process, NLG (Natural Language Generation), involves the generation of the CLOnE language from an existing ontology. The text generator and CLOnE authoring processes combine to form a RoundTrip Ontology Authoring(ROA) environment: a user can start with an existing imported ontology or one originally produced using CLOnE, (re)produce the CNL using the text generator, edit the text as required and subsequently parse the text back into the ontology using the CLOnE environment. The process can be repeated as necessary until the required result is obtained. Building on previous methodology(Chapter 4, Section 4.3.1), we undertook a repeated-measures, task-based evaluation, comparing the RoundTrip Ontology Authoring process with Protégé. In the previous chapter, we required a reference guide in order to use the controlled language. In this chapter, however, we investigate whether the substitution of NLG can reduce the learning curve for users, while simultaneously improving upon existing results for basic ontology editing tasks.

A well-known implementation which employs the use of NLG to aid the knowledge creation process is WYSIWYM (*What you see is what you meant*)(See Chapter 3 and [Power et al., 1998b]). It involves direct knowledge editing with natural language directed feedback. A domain expert can edit a knowledge based reliably by interacting with natural language menu choices and the subsequently generated natural language feedback, which can then be extended or re-edited using the menu options. The work is conceptually similar to RoundTrip Ontology Authoring, however the process is more sophisticated as natural language generation occurs as a feedback to guide the user, whereas ROA generates the CNL text out as a whole. Hence, the directed knowledge editing is not made explicit. While user evaluations with respect to WYSIWYM have been favourable (See Section 5.4), the evaluations do not compare the system to a standard ontology

editing tool. Consequently, the main research contribution of this chapter is **the first comparative task-based evaluation between a generation driven CNL tool and a standard ontology editor** - Protégé. The specific thesis contributions made by the author with respect to ROA are:

- extending the CLOnE grammar to cater for simple relations such as: **Brian studies at DERI**.
- engineering of the CLIE Java backend to cater for the new GATE 4.0 ontology API.
- reengineering of the ROA text generator cater for the new GATE 4.0 ontology API.
- modification of the ROA text generator XML template for CLOnE output.
- lead authorship with respect to the publication, on which this chapter is based [Davis et al., 2008].
- design and execution of evaluation and statistical analysis presented in Section 5.3.

The remainder of this chapter is organised as follows: Section 5.2 discusses the design and implementation of the ROA pipeline focusing on the NLG component - the ROA text generator, Section 5.3 presents our evaluation and discusses our quantitative findings. Section 5.4 discusses related work. Finally, Section 5.5 offers conclusions and future work, with a particular focus on the application of CNL to semantic annotation, which is explored in the next chapter.

5.2 Design and Implementation

In this section, we describe the overall architecture of the Round Trip Ontology Authoring (ROA) pipeline which is implemented in GATE [Cunningham et al., 2002]. We discuss briefly extensions to existing CLOnE components of ROA, but focus the attention of this section towards describing the CLOnE text generator, the algorithm used and the XML configuration file containing templates needed to configure the controlled language output of the generator.

5.2.1 RoundTrip Ontology Authoring (ROA) and CLOnE

ROA builds on and extends the existing advantages of the CLOnE software and input language, which are described below:

1. ROA requires only one interpreter or runtime environment, the Java Runtime Environment.
2. ROA like CLOnE, uses a sub-language of English.
3. As far as possible, CLOnE is grammatically lax; in particular it does not matter whether the input is singular or plural (or even in grammatical agreement).
4. Like CLOnE, ROA can be compact; the user can create any number of classes or instances in one sentence.
5. ROA is more flexible and easier to learn by using simple examples of how to edit the controlled language generated by the text generator in order to modify the ontology. It reduces the need to learn the Controlled Language by following examples, style guides or CLOnE syntactic rules. Instead, a user can create or modify various classes and instances in one (generated) sentence or (using simple copy and paste) create new properties between new or existing classes and instances.
6. The CLOnE grammar within ROA has been extended to handle simple verbs and phrasal verbs.
7. Like CLOnE any valid sentence of ROA can be unambiguously parsed.
8. The advantage of the GATE Ontology API allows users to import existing ontologies for generation, subsequent editing in ROA and export the result to different ontology formats.
9. SimpleNLG [Gatt and Reiter, 2009] has been added into the ROA text generator to lexicalise unseen properties.

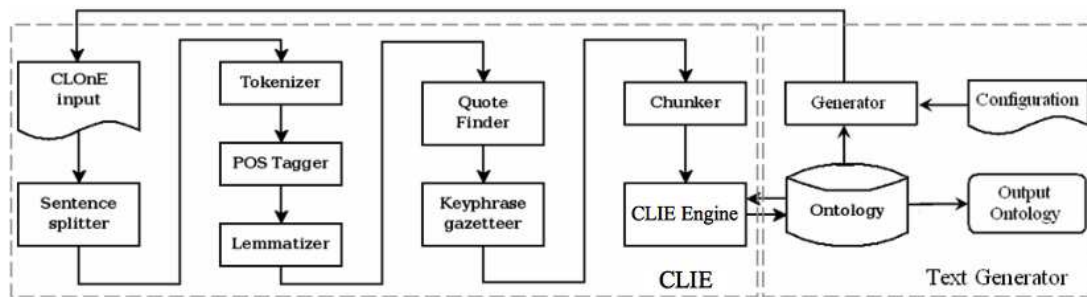


Figure 5.1: The ROA RoundTrip Ontology Authoring pipeline

Procedurally, CLOnE’s analysis consists of the ROA pipeline of processing resources (PRs) shown in Figure 5.1 (left dotted box). This pipeline starts with a series of standard GATE NLP tools which add linguistic annotations and annotation features to the document. These are followed by three PRs developed particularly for CLOnE: the gazetteer of keywords and phrases fixed in the controlled language and two JAPE⁹⁶ transducers which identify quoted and unquoted chunks. Names enclosed in pairs of single or double quotation marks can include reserved words, punctuation, prepositions and determiners, which are excluded from unquoted chunks in order to keep the syntax unambiguous. The last stage of analysis, the CLIE JAPE transducer, refers to the existing ontology in several ways in order to interpret the input sentences. Table 5.1 below provides an excerpt of the grammar rules of the CLOnE language (including the extension for verbs). We refer the reader to 4.2.3 for additional rules and examples.

5.2.2 Round Trip Ontology Authoring (ROA) - Interface

At startup, both the CLIE (Controlled Language for Information Extraction) tools and ROA text generator are loaded, including the initial textual data. The user will have a

⁹⁶GATE provides the *JAPE* (Java Annotation Pattern Engine) language for matching regular expressions over annotations, adding additional annotations to matched spans, and manipulating the match patterns with Java code.

Table 5.1: Excerpt of CLOnE grammar with examples

Sentence Pattern	Example	Usage
Forget everything.	Forget everything.	Clear the whole ontology corpus to start with the new ontology.
(Forget that) There is/are <CLASSES>.	There are researchers, universities and conferences.	Create or delete (new) classes.
(Forget that) <INSTANCES> is a/are <CLASS>.	Ahmad Ali Iqbal and Brian Davis are 'Ph.D. Scholar'.	Create (or delete) instances of the class.
(Forget that) <SUB-CLASSES> is/are a type/types of <SUPER-CLASS>.	'Ph.D. Scholar' is a type of Student.	Make subclass(es) of an existing super-class. 'Forget that' only unlinks the the subclass-superclass relationship.
(Forget that) <CLASSES/INSTANCES> <VERB PROPERTY> <CLASSES/INSTANCES>.	Professor supervises student.	Create the property of the form <i>Domain_verb_Range</i> either between two classes or instances.

window like the one shown in Figure B.2 to work with and can click on the buttons or tabs across the top to bring up the following panes.

Messages This pane explains in detail what CLIE has just done and includes error messages. The user can distinguish the error messages by the word *WARNING*, and probably ignore the *INFO* messages.

Text input In this pane (shown in Figure 5.2) the user can edit statements in CLOnE. To clear any input, the user can select all the text with the mouse and press the backspace key. It is recommended to edit individual parts. This can be achieved as

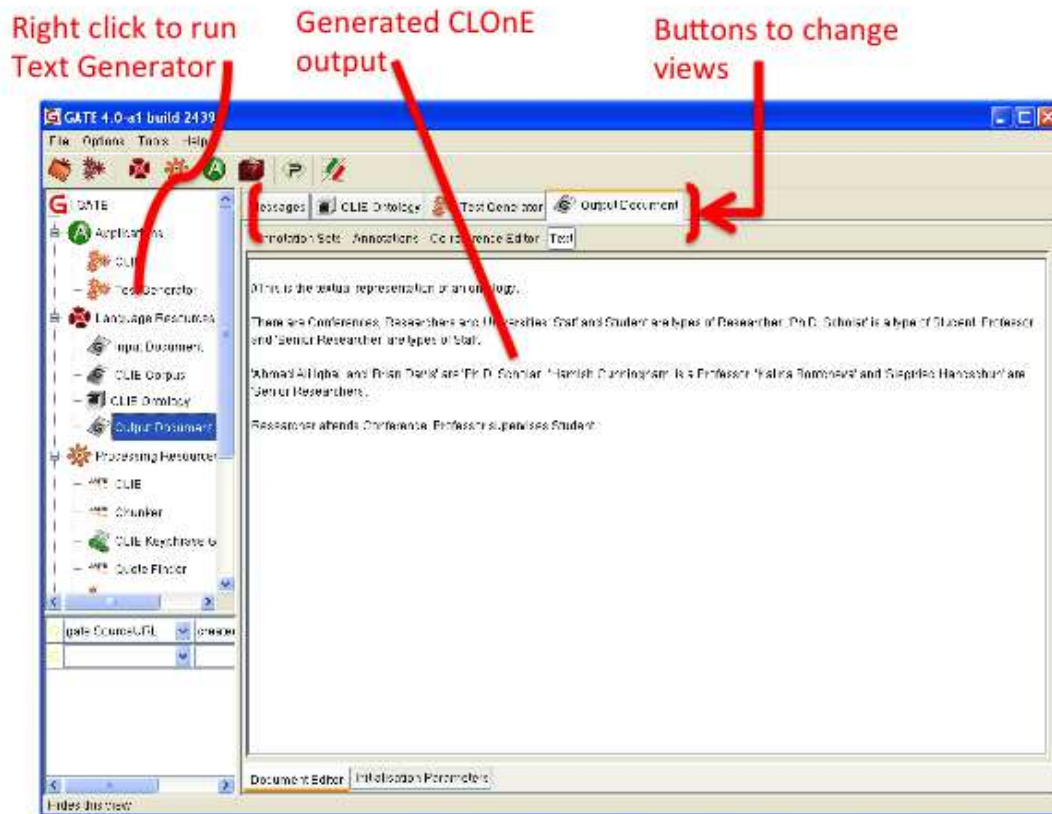


Figure 5.2: Generated CLOnE output

one would edit text normally using the arrow and backspace keys and the mouse.

Ontology This pane (shown in Figure 5.3) shows you the state of the ontology, represented by a class and instance diagram (top left), a general list of properties (below), and information about the selected class or instance (right). Click on classes or instances to change the information in the right-hand section. Before beginning any tasks, the user might wish to look at the initial ontology to become familiar with it.

To enable RoundTrip Ontology Authoring on your current input text, right click on the word *CLIE* near the top of the left-hand pane and click *Run* in the menu that appears. For further details with respect to the ROA user interface, we refer the reader to Appendix B.

5.2.3 Text generation of CLOnE

The text generation component in Figure 5.1 (right dotted box) displayed in the ROA pipeline is essentially an Ontology Verbaliser. Unlike some NLG systems, the communicative goal of the text generator is not to construct tailored reports for specific content within the knowledge base or to respond to user specific queries. Hence no specific content selection subtask or "choice" is performed since our goal is to describe and present the Ontology in textual form as unambiguous subset of English - the CLOnE language for reading, editing and amendment. We select the following content from the Ontology: top level classes, subclasses, instances, class properties, their respective domain and ranges and instance properties. The text generator is configured using an XML file, whereby text templates are instantiated and filled by the values from the Ontology. This file is decoupled from the text generator PR. Examples of two templates used to generate top level classes and class properties are displayed in Figure 5.4. The text generator (See **Generator** in Figure 5.1) is realised as a GATE PR and consists of **three** stages:

Stage 1 within the text generator converts the input ontology into an internal GATE ontological resource and flattens it into RDF style triples. This is executed in a breadth-first manner—so lists are created where super-classes always precede their corresponding subclasses—in the following order: top-level classes, subclasses, instances, class properties, and instance properties.

Stage 2 matches generation templates from the configuration file based on textual order (See Figure 5.4) with the triples list derived from the Ontology in Stage 1. A generation template has three components: (1) an *in* element containing a list of triple specifications, (2) an *out* element containing phrases that are generated when a successful

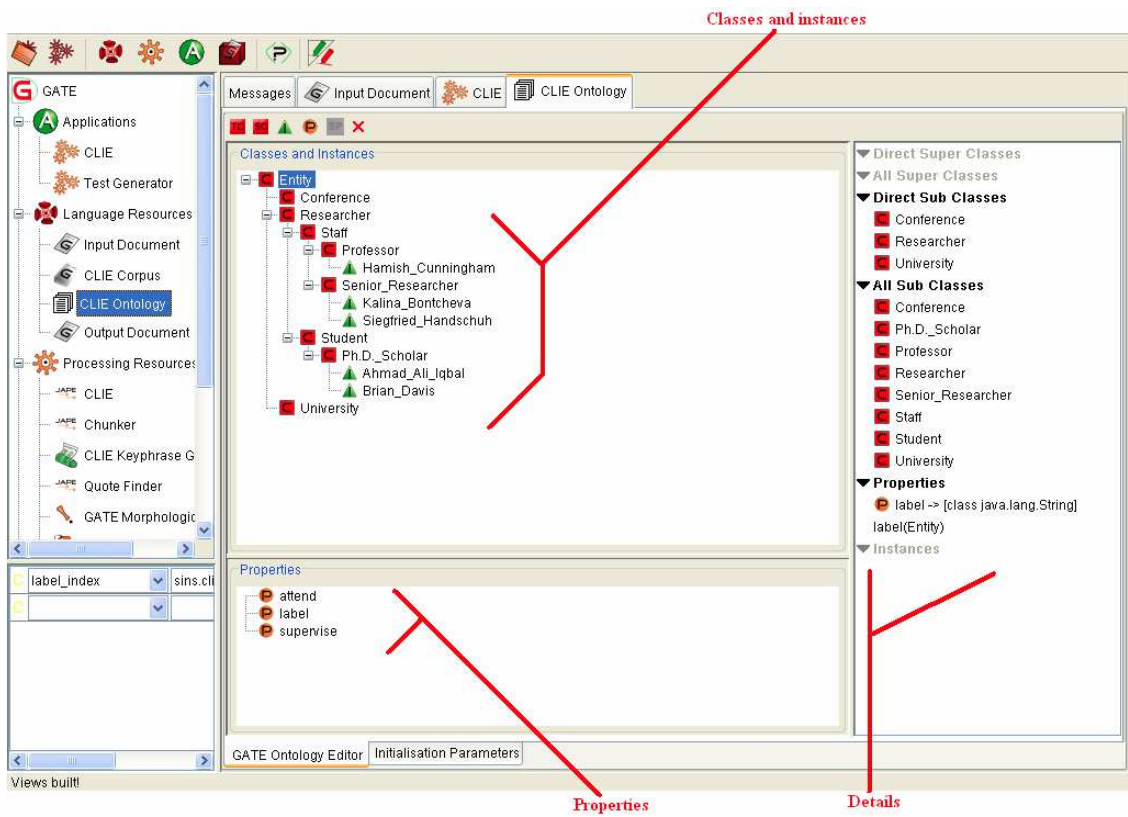


Figure 5.3: ROA Ontology viewer

match has occurred and (3) an optional `ignoreIf` element for additional triple specifications that cause a match specified in the `in` element to be ignored if the conditions are satisfied. The triple specifications contained within the `in` portion of the template can have subject, property and object XML elements. The triple specifications act as restrictions or conditions, such that an input triple generated from the Ontology must match this template. If more than one triple is included in the `in` element they are considered as a conjunction of restrictions, hence the template will only match if one or more actual triples for all triple specifications within the `in` element are found. One triple can reference another, i.e., a specification can constrain a second triple to have the same object as the subject of the first triple. Only backward referencing is permitted since the triples are matched in a top down fashion according to their textual ordering. An example of referencing can be seen in line 188 of the `out` element of the template shown in Figure 5.4 for generating class properties.

In **Stage 3** the `out` section of the template describes how text is generated from a successful match. It contains phrase templates that have text elements and references to values matched within the `in` elements. Phrases are divided into singular and plural forms. Plural variants are executed when several triples are grouped together to generate a single sentence (Sentence Aggregation) based on a list of Ontology objects (i.e., **There are Conferences, Students and Universities**). Text elements within a template are simply copied into the output while reference values are replaced with actual values based on matching triple specifications. We also added a small degree of lexicalisation into the Text Generator PR, whereby, for example, an unseen property, which is treated as a verb is inflected correctly for surface realisation i.e. **study** and **studies**. This involves a small amount of dictionary look-up using the SimpleNLG Library [Gatt and Reiter, 2009] to obtain the third person singular inflection **studies** from **study** to produce **Brian Davis studies at NUIG**. A new object of type `Lexicon` is created and the base or infinitive form of the verb **study** can be passed as parameter to the method `getPresent3SG(String lemma)`. In addition, we might wish to concatenate a preposition such as **at** or **with** in

order to produce a phrasal verb such as `studies at` or `studies with`.

```
Lexicon lex=new Lexicon();  
verb = lex.getPresent3SG(lemma);  
result = verb.concat("_"+prep);
```

The out elements of the generation template also provide several phrase templates for the singular and plural sections. These are applied in rotation to prevent tedious and repetitious output.

Stage 2 also groups matches together into sets that can be expressed together in a plural form. For this to proceed, the required condition is that the difference between matches, occurs in only one of the references used in the phrase templates, i.e., if singular variants would only differ by one value. A specialised generation template with no `in` restrictions is also included in the configuration file. This allows for the production of text where there are no specific input triple dependencies.

```

56 <!-- Template for all the other top classes -->
57 <template>
58   <in>
59     <triple id="t1">
60       <property ns="rdf" name="type"/>
61       <object ns="owl" name="Class"/>
62     </triple>
63     <triple id="t1.subject">
64       <subject ref="t1.subject"/>
65       <property ns="rdfs" name="subClassOf"/>
66       <object name="Entity"/>
67     </triple>
68   </in>
69   <out>
70     <singular>
71       <phrase>There are
72       <ref ref="t1.subject" number="plural"/>.
73     </phrase>
74   </singular>
75   <plural>
76     <phrase>There are
77     <ref ref="t1.subject" number="plural"/>.
78   </phrase>
79 </plural>
80 </out>
81 <ignoreIf>
82 </ignoreIf>
83 </template>

166 <!-- Template for defining class properties -->
167 <template>
168   <in>
169     <triple id="t1">
170       <property ns="rdf" name="type"/>
171       <object ns="rdf" name="Property"/>
172     </triple>
173     <triple id="t2">
174       <subject ref="t1.subject"/>
175       <property ns="rdfs" name="domain"/>
176     </triple>
177     <triple id="t3">
178       <subject ref="t1.subject"/>
179       <property ns="rdfs" name="range"/>
180     </triple>
181   </in>
182   <out>
183     <singular>
184       <phrase>
185         <ref ref="t2.object" number="singular"/>
186         <ref ref="t1.subject" number="singular"/>
187         <ref ref="t3.object" number="singular"/>.
188       </phrase>
189     </singular>
190     <plural>
191       <phrase>
192         <ref ref="t2.object" number="singular"/>
193         <ref ref="t1.subject" number="singular"/>
194         <ref ref="t3.object" number="plural"/>.
195       </phrase>
196     </plural>
197   </out>
198 </template>

```

Figure 5.4: Example of a generation template

5.3 Evaluation

5.3.1 Methodology

Our evaluation methodology is based on the criteria previously used to evaluate CLOnE (See Section 4.3.1), so that we can fairly compare the earlier results using the CLOnE software with the newer RoundTrip Ontology Authoring (ROA) process. The methodology involves a *repeated-measures, task-based* evaluation: each subject carries out a similar list of tasks on both tools being compared. Unlike our previous experiment, the CLOnE reference guide list and examples are withheld from the participants, so that we can measure the benefits of substituting the text generator for the reference guide and determine its impact on the learning process and usability of CLOnE. Furthermore, we used a larger sample size and more controls for bias. All evaluation material and data are available online for inspection, including the CLOnE evaluation results for comparison in Appendices A and B. The evaluation contained the following:

- A pre-test questionnaire asking each subject to test their degree of knowledge with respect to ontologies, the Semantic Web, Protégé and Controlled Languages. It was scored by assigning each answer a value from 0 to 2 and scaling the total to obtain a score of 0–100.
- A short document introducing Ontologies, the same ‘quick start’ Protégé instructions as used in Chapter 4, Section 4.3.1 (partly inspired by Protégé’s *Ontology 101* documentation [Noy and McGuinness, 2001]), and an example of editing CLOnE text derived from the text generator. The CLOnE reference guide and detailed grammar examples used in for the previous experiment in Chapter 4, Section 4.3.1 were withheld. The participants were allowed to refer to an example of how to edit generated Controlled Language but did *not* have access to CLOnE reference guide. (See Appendices A and B for details of the manual and CLOnE language and ROA interface respectively.)

- A post-test questionnaire for each tool, based on the *System Usability Scale* (SUS), which also produces a score of 0–100 to compare with previous results [Brooke, 1996].
- A comparative questionnaire similar to the one used in Chapter 4, Section 4.3.1 was applied to measure each user’s preference for one of the two tools. It is scored similarly to SUS so that 0 would indicate a total preference for Protégé, 100 would indicate a total preference for ROA, and 50 would result from marking all the questions *neutral*. Subjects were also given the opportunity to make comments and suggestions.
- Two equivalent lists of ontology-editing tasks, each consisting of the following sub-tasks:
 - creating two subclasses of existing classes,
 - creating two instances of different classes, and
 - either (A) creating a property between two classes and defining a property between two instances, or (B) extending properties between two pairs of instances.

For both task lists, an initial ontology was created using CLOnE. The same ontology was loaded into Protégé for both tasks and the text generator was executed to provide a textual representation of the ontology for editing purposes(see Figure 5.5), again for both tasks.

For example, Task List A is as follows.

- Create a subclass *Institute* of *University*.
- Create a subclass *Workshop* of *Conference*.
- Create an instance *International Semantic Web Conference* of class *Conference*.
- Create an instance *DERI* of class *Institute*.



Figure 5.5: Text Generated by ROA

- Create a property that *Senior Researchers supervise Student*.
- Define a property that *Siegfried Handschuh supervises Brian Davis*.

5.3.2 Sample quality

We recruited 20 volunteers from the Digital Enterprise Research Institute, Galway⁹⁷. The sample size ($n = 20$) satisfies the requirements for reliable SUS evaluations, according to [Tullis and Stetson, 2004]. We recruited subjects with an industrial background (**I**) and participants with a research background (**R**). See (in Table 5.6) for details. In addition we attempted to control bias by selecting volunteers who were either:

- Research Assistants/Programmers/Post-Doctoral Researchers with an industrial background either returning (or new) to Academic Research respectively(**I**),
- Postgraduate Students who were new to the Semantic Web and unfamiliar with Ontology Engineering(**R**),

⁹⁷<http://www.deri.ie>, Retrieved 2008-05-22

- Researchers from the E-learning and Sensor Networks lab but not from the Semantic Web Cluster(**R**),
- Researchers with no background in Natural Language Processing or Ontology Engineering(**R**) or
- Industrial Collaborators (**I**).

In all cases, we tried to ensure that participants had limited or no knowledge of GATE or Protégé. First, subjects were asked to complete the pre-test questionnaire, then they were permitted time to read the Protégé manual and Text Generator examples, and lastly they were asked to carry out each of the two task lists with one of the two tools. (Half the users carried out task list A with ROA and then task list B with Protégé; the others carried out A with Protégé and then B with ROA.) Each user's time for each task list was recorded. After each task list the user completed the SUS questionnaire for the specific tool used, and finally the comparative questionnaire. Comments and feedback were also recorded on the questionnaire forms.

5.3.3 Hypothesis statements

With respect ROA SUS scores we state the our first hypothesis pair as:

4.1H₀: It is predicated that ROA usability (SUS) scores will not be greater than neutral($\mu \not> 65$).

4.1H₁: It is predicated that ROA usability (SUS) scores will be greater than neutral ($\mu > 65$).

To clarify, we are stricter in our interpretation of neutral' in that only SUS values of >0.65 are considered to be strong SUS scores as they are above the threshold of neutral or weak values [Bailey, 2006].

In addition with respect to ROA/Protégé preference scores, we have the second hypothesis pair:

4.2H₀: It is predicated that the ROA/Protégé preference score not be greater than neutral ($\mu \not\geq 65$).

4.2H₁: It is predicated that the ROA/Protégé preference score will be greater than neutral ($\mu > 65$).

Finally, with respect to the ROA/Protégé times we have the third hypothesis pair:

4.3H₀: It is predicated that the ROA times will not differ significantly than the Protégé times.

4.3H₁: It is predicated that the ROA time will differ significantly than the Protégé times.

5.3.4 Results and Data Analysis

Table 5.2 and Table 5.3 summarise the main measures obtained from our evaluation. We used SPSS⁹⁸ to generate all our statistical results. In particular the mean ROA SUS score is above the baseline of 65–70% while the mean SUS score for Protégé is well below the baseline [Bailey, 2006]. In the ROA/Protégé Preference (R/P Preference) scores, based on the comparative questionnaires, we note that the scores also favour on average ROA over Protégé. Confidence intervals are displayed in Table 5.4.⁹⁹

With respect to

4.1H₀: It is predicated that ROA usability (SUS) scores will not be greater than neutral ($\mu \not\geq 65$),

we can reject the null hypothesis with $z=3.5$ $p < 0.5$.

Additionally, with respect to:

4.2H₀: It is predicated that the ROA/Protégé preference score will not be greater than neutral ($\mu \not\geq 65$),

we can reject the null hypothesis with $z=2$ $p < 0.5$.

⁹⁸SPSS 2.0, <http://www.spss.com>, Retrieved 2013-05-22

⁹⁹A data sample's *95% confidence interval* is a range 95% likely to contain the mean of the whole population that the sample represents [John L. Phillips, 1996].

Table 5.2: Summary of the questionnaire scores

Measure	min	mean	median	max
Pre-test scores	17	42	42	75
ROA SUS rating	48	74	70	100
Protégé SUS rating	10	41	41	85
R/P Preference	40	72	79	95

Table 5.3: Summary of the Task per Tool times

Task	Tool	min	mean	median	max
A	ROA	4.08	6.25	6.12	9.90
A	Protégé	3.57	7.05	7.33	10.38
A	Both	3.57	6.65	6.23	10.38
B	ROA	2.15	4.27	4.43	6.20
B	Protégé	4.45	9.22	10.00	11.82
B	Both	6.75	6.75	6.12	11.82
A&B	ROA	2.15	5.27	5.47	9.90
A&B	Protégé	3.57	8.13	8.15	11.82

We also conducted one way repeated measures ANOVA between ROA and Protégé times respectively. With respect to:

4.3H₀: It is predicated that the ROA times will not differ significantly than the Protégé times,

we can reject the null hypothesis as the compared groups differed significantly, $F(1,19) = 42.36$, $p < 0.5$, multivariate partial eta squared = .70.

For post-hoc data analysis, we generated Pearson's and Spearman's correlations coefficients [John L. Phillips, 1996, Connolly and Sluckin, 1971]. Table 5.5 displays the coef-

Table 5.4: Confidence intervals (95%) for the SUS scores

Tool	Confidence intervals		
	Task list A	Task list B	Combined
Protégé	28–55	29–51	32–49
ROA	63–77	69–84	68–79

ficients. In particular, we note the following results:

- The pre-test score has a weak negative correlation with ROA task time.
- There are no correlations with pre-test score and the ROA SUS score.
- The pre-test score has a weak negative correlation with the Protégé SUS score.
- There are no correlations with pre-test score and the Protégé time.
- In previous results in comparing CLOnE and Protégé, the task times for both tools were more positively correlated with each other while in the case of ROA and Protégé, there correlation has being weakened by a significant 32% of its original value (of 78%) reported for CLOnE in Section 4.3.1, indicating that the users tended not spend the equivalent time completing both ROA and Protégé tasks.
- There is a moderate correlation with Protégé task time and Protégé SUS scores.
- There is a strong negative correlation of -0.65 between the ROA task time and the ROA SUS scores. Our previous work reported no correlation between the CLOnE task time and CLOnE SUS time. A strong negative or inverse correlation implies that users who spent less time completing a task using ROA tended to produce high usability scores - favouring ROA. More importantly, we noted that the associated probability reported by SPSS, was less then the typical 5% cut-off point used in social sciences.

Table 5.5: Correlation coefficients

Measure	Measure	Pearson's	Spearman's	Correlation
Pre-test	ROA time	-0.41	-0.21	weak -
Pre-test	Protégé time	-0.28	-0.35	none
Pre-test	ROA SUS	-0.02	-0.00	none
Pre-test	Protégé SUS	-0.32	-0.29	weak -
ROA time	Protégé time	0.53	0.58	+
ROA time	ROA SUS	-0.65	-0.52	-
Protégé time	Protégé SUS	0.53	0.56	+
ROA time	Protégé SUS	-0.14	-0.10	none
Protégé time	ROA SUS	-0.02	-0.09	none
ROA SUS	Protégé SUS	0.04	-0.01	none
ROA SUS	R/P Preference	0.58	0.56	+
Protégé SUS	R/P Preference	-0.01	0.10	none

Hence the correlation, revealed for the data that user preference for RoundTrip Ontology Authoring over Protégé and decreased time spent completing ontology editing tasks were significantly related, $r=-.65$, $n=20$, $p < .05$, two tails. (Note, alpha was actually adjusted to .004 from .05 as per the Bonferroni correction¹⁰⁰ to cater multiple correlations) however p was still below this value).

- The R/P Preference score correlates moderately with the ROA SUS score, similar to previous results, but no longer retains a significant inverse correlation with the Protégé SUS score. The reader should note the R/P Preference scores favour ROA over Protégé.

We also varied the tool order evenly among our sample. As noted in the previous chap-

¹⁰⁰See http://en.wikipedia.org/wiki/Bonferroni_correction, Retrieved 2013-05-22

ter, specifically Section 4.3.1, once again the SUS scores differ slightly according to tool order (as indicated in Table 5.4). Previous SUS scores for Protégé tended to be slightly lower for B than for A, which we believe may have resulted from the subjects’ decrease in interest as the evaluation progressed. While in previous results there was a decrease in SUS scores for CLOnE (yet still well above the SUS baseline), in the case of ROA however, the SUS scores increased for task B (see Table 5.4), implying that if waning interest was a factor in the decrease in SUS scores for CLOnE, it does not appear to be the case for ROA. What is of additional interest is that group **I**, subjects with industrial background scored on average 10% higher for both ROA SUS and ROA/Protégé, which implies that Industrial collaborators or professionals with an Industrial background favoured a natural language interface over a standard ontology editor even more than Researchers.

Table 5.6: Groups of subjects by source and tool order

Source		Tool order		Total
		PR	RP	
R	Researcher	5	7	12
I	Industry	5	3	8
Total		10	10	20

5.3.5 User Feedback

The participants also provided several suggestions/comments about ROA.

- “RoundTrip Ontology Authoring becomes much easier, once the rules are learnt”. (This is very interesting considering that no syntax rules, extended examples or restricted vocabulary list were provided).
- Use of inverted commas should be used only once and afterwards, if same the class /instance is reused, the system should automatically recognise it as the previous

Table 5.7: Comparison of the two sources of subjects

Measure	Group	min	mean	median	max
Pre-test	R	17	38	38	58
	I	17	47	50	75
ROA SUS	R	48	69	70	82
	I	65	80	80	100
Protégé SUS	R	10	30	28	52
	I	12	48	49	85
R/P Preference	R	40	68	72	88
	I	65	78	78	95

word.

- Many participants suggested displaying the ontology pane on the right hand side of the text pane, where test users edit the text instead of moving between two separate panes.
- Some users suggested dynamic ontology generation, once a user finishes typing a sentence, the changes should be displayed automatically in the ontology pane.
- Similar suggestions to the previous evaluation were provided for user auto-completion, syntax highlighting, options about available classes, instances or property names and keywords should be displayed, a similar concept to modern Word Processor or programming IDEs such as Eclipse.
- Some participants with an industrial background demonstrated concern regarding scalability and ROA using with a larger business related ontology and suggest capabilities for verbalising a portion of the ontology tree within the Ontology viewer, using text generation for subsequent editing.

- Some test users appreciated the singular/plural forms and sentence handling of ROA (e.g., study, studies).

5.4 Related work

With respect to related work, we will focus our review on tools which generate CNL. For a more thorough review of Controlled Natural Languages, we refer the reader to Chapter 3.

ACE OWL, a sublanguage of Attempto Controlled English (ACE), proposes a means of writing formal, simultaneously human- and machine-readable summaries of scientific papers [Kaljurand and Fuchs, 2006, Kuhn, 2006]. Similar to RoundTrip Ontology Authoring, ACE OWL also aims to provide reversibility (translating OWL DL into ACE). The application NLG to ACE OWL is for the purposes editing existing ACE text [Kaljurand and Fuchs, 2007]. The paper discusses the implementation of a shallow NLG system - an OWL Verbaliser, focusing primarily on the OWL to ACE rewrite rules, however no evaluation or quantitative data is provided in an attempt to measure the impact of NLG in the authoring process.

As mentioned at the beginning of this chapter, a well-known implementation which employs the use of NLG to aid the knowledge creation process is WYSIWYM (*What you see is what you meant*) (See Chapter 3 and [Power et al., 1998b]). It involves direct knowledge editing with natural language directed feedback. A domain expert can edit a knowledge base reliably by interacting with natural language menu choices and the subsequently generated feedback, which can then be extended or re-edited using the menu options. The method is also known as *Conceptual Authoring* [Hallett et al., 2007]. One application of WYSIWYM has been in the context of querying and creating metadata for the Semantic Grid within the social sciences domain [Hielkema et al., 2008]. The research is supported by the PolicyGrid project [Hielkema et al., 2007b] which aims to investigate how best to support social scientists via the Semantic Grid. Non-experts in the form of social scientists attempt to create and share resources as well as their associated RDF metadata using WYSIWYM [Hielkema et al., 2007a]. Users are presented with text which

is an expansion point or anchor for an object mention, which triggers a menu listing all possible properties associated with that object. The WYSIWYM interface is built using the Google Web toolkit and the information the user is editing is stored in a semantic graph. The domain ontology is consulted throughout the graph construction, specifically the domain and range of properties in each anchor menu. Finally, the semantic graph is translated into RDF triples. The language generation component maps the semantic graph to HTML text. In addition, the tool also attempts to identify incoming sentence types for transformation into templates. Sentence generations from a property in the ontology are produced and the user may select the desired sentence and edit, if needed. Once the user is satisfied with the selection the corresponding dependency tree is stored in the lexicon for realising future instances of the property.

An evaluation of WYSIWYM was carried out with 16 researchers and PhD students from the social sciences domain. Users were shown a six minute background video which described the main functionalities of the WYSIWYM interface [Hielkema et al., 2008]. Descriptions of four resources (documents to associate metadata to) were provided to the users. These descriptions were described as paragraphs of English. The goal being to reproduce the descriptions using the WYSIWYM tool. Each subject also received the descriptions in varied order. Four descriptions were given, which were further decided in eight to ten sub tasks. The successful completion of certain sub tasks was dependent on the preceding subtask. Task completion times, number of operations as well as errors, including “avoidable” errors (which imply the result of an error introduced from a previous subtask), were measured. The results were encouraging, where users mean completion times decreased significantly. Hence, users gained speed over time. In addition, user feedback was positive, however the results were less positive in comparison to an earlier evaluation of WYSIWYM [Hallett et al., 2007], whereby users completion of tasks was less accurate [Hielkema et al., 2008]. Note that the domain ontology was medical as it was in the context for the CLEF¹⁰¹ project. Furthermore, the evaluation involved composing

¹⁰¹<http://www.clinical-escience.org/>, Retrieved 2008-05-22

SQL queries to a relational database. More importantly, users from the social sciences field reported that they were overwhelmed by the large number of options available i.e. thirty properties per one object. CLEF was also developed for the well structured domain of medicine while social sciences tends to be more varied with many different theories and approaches. Consequently, the underlying domain ontology can have a large a significant impact on usability.

The advantage of WYSIWYM is that it eliminates possibilities of interpretation error, and, similar to ROA, training requirements are minimal. Originally, WYSIWYM was targeted to the instance level but recent efforts have targeted it towards authoring at the class and axiom level [Power et al., 2009], where sentences denote axioms and major sentence constituents denote classes. With respect to the initial prototype, classes may be count and proper nouns whereas properties may be represented as transitive verbs.

Another WYSIWYM inspired CNL is OWL Simplified English [Power, 2012]. It is a finite state language for ontology editing. It seeks to maximise on Kuhn's notions of *simplicity* and *naturalness* (See Chapter 3) at the expense of *expressiveness*. The argument for the finite state approach is that the majority of OWL expression created by ontology developers are invariably right branching and hence can be recognised by a finite state grammar. Based on previous studies of ontology corpora, the authors show how the individuals, classes and properties tend to have distinct part of speech (POS) tags. Individuals or instances tend to be either proper nouns, common nouns or numbers, while classes are composed mostly of common nouns, adjectives and proper nouns. Finally, properties tend to open with a verb or auxiliary verb in the present tense. In their paper [Power, 2012], they describe a finite state network that is capable of interpreting the CNL sentences in the grammar with minimal knowledge of content words. Similar the work in this thesis (See Chapter 6) OWL Simplified English permits the acceptance of some technical phrases that violate normal English. The language can capture ontology operations such as simple negation, cardinality, object intersection but aims to reduce or eliminate structural ambiguity. In order to support their theory that

ontology developers in general tend to favour right branching statements, they conducted analysis on an ontology corpus of 500 ontologies containing some 500,000 axioms. They derived class patterns of low complexity levels and conducted analysis of the corpus and concluded that there are some 300 axioms that cannot be verbalised in their CNL. These results apply to ontology classes constructed from intersection and restriction. As more complexity, such as union and compliment is added, the portion of unverbalsed axioms rises. Hence, there is the risk of reduced coverage of expressivity within the CNL, however their analysis confirms that ontology developers tend to favour right branching statements. OWL Simplified is similar to CLOnE in that it is implemented finite state parsing techniques and likewise *simplicity* and *naturalness* are preferred at the expense of *expressiveness*. We include OWL simplified English in this chapter as the interface, under construction, is a WYSIWYM based interface.

As mentioned in the previous chapter, *GINO* (Guided Input Natural Language Ontology Editor) provides a guided, controlled NLI (natural language interface) for domain-independent ontology editing for the Semantic Web. However, unlike ROA, a full textual description of the Ontology is not realised [Bernstein and Kaufmann, 2006].

While much of the applications described involve verbalising ontologies as CNLs, the work of [Stevens et al., 2011] takes this a step further by attempting to verbalise coherent text paragraphs. The approach tries to automatically provide text based definitions from an ontology. Rather than verbalising axioms as individual sentences the approach groups axioms into common structures and attempts to generate a coherent text in the form of a paragraph of English text. The natural language output is generated from the Experiment Factor Ontology, which is an application ontology used to describe experimental variables in functional genomics data [Malone et al., 2010]. EFO uses OWL to produce a rich axioms description of classes in the domain. Surveys were conducted involving the developers of EFO, who were sufficiently satisfied with the output of the generated definitions such that they were incorporated into the EFO. While the generated definitions do not substitute for hand-written textual definitions, they are a good

starting point. With respect to generating paragraphs, the justification for taking this approach is informed by related work in psycholinguistics, in that unordered collections of sentences are difficult to understand. Text comprehension depends on inferences made by the reader based on the organisation of sentences into structured units or paragraphs. So one concept in an ontology can correlate to one paragraph. Issues which arise from such an approach are grouping and aggregation as well as uncovering the optimal balance between preserving OWL semantics and producing readable English. The authors conducted two informal surveys by verbalising fifty cells lines from EFO. Participants were asked to judge the readability of ten text definitions. While results of the surveys were not statistically significant, the feedback received, suggested that text descriptions of the definitions were useful in the context of an ontology which contained sparse definitions. Feedback from users indicated that making semantics explicit in initial verbalisations was obtrusive as well as the explicit verbalisation. The system has some limitations in that the coverage is restricted to a subset of OWL which excludes inverse properties, enumeration of classes with multiple arguments and datatypes. The NLG approach is part of their overall SWAT (Semantic Web Authoring Toolkit)¹⁰² tools suite. The SWAT tools take document structuring to a more sophisticated level than other ontology verbalisers [Stevens et al., 2011].

In addition, the system is implemented in SWI Prolog a language that is very suitable for NLP tasks but can under-perform [Gazdar and Mellish, 1989]. The process for generating descriptions has five phases: translation or transcoding from OWL into Prolog structures, construction of a lexicon for atomic entities, selecting relevant axioms for describing each class, aggregating axioms with a similar structure and surface realisation of sentences from input axioms including aggregation. The evaluation in [Stevens et al., 2011] also explored the generation of varied form of properties used in EFO, as well as variations with or without aggregation and various degrees of elision of repeated noun phrases and ‘something that’ phrases.

¹⁰²<http://mcs.open.ac.uk/nlg/SWAT/editor.html>, Retrieved 2011-05-22

A related evaluation involves the application of the same system to generate organised text similar to an online encyclopaedia or glossary. In [Williams et al., 2011], the authors investigate whether such a hierarchical organisation improves on text comprehension and navigation. The same implementation, based on Definite Clause Grammars (DCG)s in Prolog is used. The DCGs have coverage for the majority of OWL-DL. The evaluation was conducted over fifty seven participants across special interest groups in the Association of Computational Linguists (ACL)¹⁰³, specifically SIGGEN¹⁰⁴ and SIGDIAL¹⁰⁵. The purpose was to assess whether the paragraph organisation despite its longer length was able to help people understand and navigate the text. The domain ontology chosen was that of spider anatomy. One group was provided with the encyclopaedia style hypertext, while the other group was provided with a list of verbalised unaggregated sentences. Subjects were asked for feedback regarding the usability of the navigation using a likert scale based questionnaire. In addition, they were asked to provide feedback on the various textual features such as headings, typology subheadings, description subheadings, hyperlinks within entries etc. Finally, their knowledge of the spider ontology was also tested. In summary, the evaluation indicated that the hierarchical organisation of texts is more user friendly than unorganised text with no loss in performance.

Finally, while RABBIT [Engelbrecht et al., 2009](see Chapter 3) is quite a similar implementation to CLONE, there are no generation capabilities in the RABBIT system. Likewise, there have been a lot of NLG efforts in GF, with respect generating tailored summaries for semantic repositories [Angelov and Ranta, 2009, Dannélls, 2008]. However, the notion of round tripping for ontology authoring in GF has not been evaluated, although GF like ACE is bidirectional because of its declarative nature.

¹⁰³<http://aclweb.org/>, Retrieved 2013-05-22

¹⁰⁴<http://www.siggen.org/>, Retrieved 2013-05-22

¹⁰⁵<http://www.sigdial.org/>, Retrieved 2013-05-22

5.5 Conclusion & Discussion

The main research goal of this chapter is to assess the effect of introducing Natural Language Generation (NLG) into the CLOnE Ontology authoring process to facilitate RoundTrip Ontology Authoring. The underlying basis of our research problem is the *habitability* problem (See Section 5.1): How can we reduce the learning curve associated with Controlled Languages? And how can we ensure their uptake as a Natural Language Interface (NLI)? Our contribution is empirical evidence to support the advantages of combining of NLG with ontology authoring, a process known as RoundTrip Ontology Authoring (ROA). In summary, with respect to the research questions **OA1**, **OA2** and **OA3** outlined in Chapter 1, Section 1.2, this chapter makes the following contributions:

- **OA1:** Statistical evidence that ROA is more user friendly than a standard ontology editor for basic ontology editing tasks.
- **OA2:** Statistical evidence that users take less time to complete ontology editing tasks using ROA in comparison to Protégé.
- **OA3:** Evidence that ROA is more user friendly than CLOnE for basic ontology editing tasks.

The readers should note that we compared Protégé with ROA because Protégé is the standard tool for ontology authoring. In the previous chapter, we compared CLOnE with Protégé. Hence, in order to compare ROA with CLOnE, it was necessary to repeat the experiment and use Protégé as the baseline. We make no claims that Protégé should be replaced with ROA, the point is that ROA can allow for the creation of a quick easy first draft of a complex ontology by domain experts or the creation of small to medium sized ontologies by novice users. Furthermore, a large percentage of an initial ontology would naturally consists of taxonomic relations and simple properties/relations.

Our user evaluation consistently indicated that our subjects found ROA (and continue to find CLOnE) significantly more usable and preferable than Protégé for simple ontology

editing tasks (See hypotheses 4.1H₁ and 4.2H₁). Furthermore, users spent significantly less time completing ontology editing using ROA in comparison to Protégé (See hypothesis 4.3H₁) .

In addition, our evaluation differs, in that we implemented more tighter restrictions during our selection process, to ensure that users had no background in NLP or Ontology Engineering. Furthermore, 40% of our subjects with an industrial background, scored ROA 10% higher then researchers indicating that a NLI to a Ontology Editor might be a preferred option for Ontology development within industry.

In detail, this evaluation differs from Chapter 4, Section 4.3.1 by two important factors: (1) we *excluded* the CLOnE reference manual from the training material provided in the previous evaluation; and (2) we introduced a text generator, verbalising CLOnE text from a given populated ontology and asked users to edit the ontology, using the generated CLOnE text based on an example provided. In our post-hoc analysis, we observed two new significant improvements in our results: (1) the previous evaluation indicated a strong correlation between CLOnE task times and Protégé task times, this correlation has significantly weakened by 32% between ROA and Protégé task times. Hence, where users previously required the equivalent time to implement tasks both in CLOnE and Protégé, this is no longer the case with ROA (the difference being the text generator); and (2) our previous evaluation indicated no correlation between either CLOnE/Protégé task times and their respective SUS scores. However, with ROA, the data revealed that user preference for RoundTrip Ontology Authoring over Protégé and decreased time spent completing ontology editing tasks were significantly related, $r=-.65$, $n=20$, $p < .05$, two tails. Furthermore, ROA tended to retain user interest, which CLOnE did not. While Protégé is intended for more sophisticated knowledge engineering work, this is not the case for ROA. Scalability, both in performance and usage, was also an issue raised by our test subjects. From a performance perspective, when loading large Ontologies, we do not foresee any major issues. ROA was recently ported to a newer release of GATE which contains a completely new ontology API that utilises the power of OWLIM -

OWL in Memory, a high performance semantic repository developed at Ontotext¹⁰⁶. Finally, from a user perspective, authoring memory frequently used in translation memory systems or text generation of selective portions of the ontology (using a Visual Resource) could significantly aid the navigation and authoring of large ontologies. For more details, however, with respect to future work and ROA, we refer the reader to Chapter 7.

We have already mentioned adapting CLOnE for semantic annotation, whereby the task differs from ontology authoring in that semantic annotation is concerned with the association of semantic metadata to text. We move now to Chapter 6, which explores our notion of CNL from semantic annotation, two prototypes and their respective user evaluations.

¹⁰⁶<http://www.ontotext.com/owlim/>, Retrieved 2011-05-22

6 Towards Controlled Natural Language for Semantic Annotation

This Chapter discusses the application of Controlled Natural Language (CNL) to the process of Semantic Annotation or *Controlled Annotation*. It describes implementations for two prototypes of Controlled Language ANNotator – CLANN, both varying in expressiveness (See Section 6.2). We also describe the domain ontology, based on the domain use case which involves the authoring and annotation of project minutes and status reports (See Section 6.2.1). Finally, the chapter describes a user evaluation which compares both CLANN prototypes with a standard manual semantic annotation tool - Ontomat [Handschuh et al., 2002] (See Section 6.3).

References: This chapter is based mainly on [Davis et al., 2010].

6.1 Introduction

The previous two chapters (See Chapters 4 and 5) discussed Controlled Natural Languages (CNL)s in the context of ontology authoring. Ontologies form the crucial backbone of the Semantic Web. The process of creating, extending and adapting ontologies can be described as the *knowledge meta process* [Sure, 2003]. The subsequent creation of metadata to describe a web resource using an ontology is called the *knowledge process* [Sure, 2003]. The actual method for the creation of metadata and its subsequent association to a resource is called Semantic Annotation, which is one of the core challenges for building the Semantic Web.

Semantic Annotation as a process may be either manual, semi-automatic or automatic

(See Chapter 3, Section 3.2). Manual semantic annotation is a complex and laboured task, which is both time-consuming, expensive and often requires specialist annotators or the subsequent training of such annotators. This may require (an arguably unnecessary) exposure to formal ontological description. Such formal data representation can act as a significant deterrent for non-expert users or organisations seeking to annotate resources as part of their daily activity, thus allowing them to fully benefit from the adoption of Semantic Web technologies. While (Semi)-automatic annotation tools attempt to remove this constriction, which is commonly known as the *knowledge acquisition bottleneck*, their application often requires access to specialists who can combine Natural Language Processing(NLP)/Machine Learning(ML) and Semantic Web ontology languages. Such specialists are costly and rare and furthermore the creation or acquisition of quality language resources to bootstrap such approaches may require significant investment, which for a small to medium enterprises may not be justifiable. Consequently, this challenges researchers to develop user-friendly manual annotation environments to support the knowledge acquisition process.

CNLs offer an incentive to the novice user to annotate, while simultaneously authoring, her respective documents in a user-friendly manner, yet at the same time shielding her from the underlying complex knowledge representation formalisms of ontology languages. A natural overlap exists between tools used for both ontology creation and semantic annotation. Despite such efforts, very little research has focused on applying CNLs to semantic annotation. The specific thesis contributions made by the author with respect to this chapter are:

- conceptualisation and definition of controlled annotation
- grammar design and engineering of Controlled Language ANNotator (CLANN) Type I and Type II (See Section 6.2).
- lead authorship of the publication on which this chapter is based [Davis et al., 2010].
- design and execution of user evaluation and analysis (See Section 6.3).

The reader should note that there is a subtle *difference* between the process of ontology creation and population and that of Semantic Annotation. We describe Semantic Annotation as “a process as well as the outcome of the process”. Hence it describes i) “the process of adding semantic data or metadata to content given an agreed ontology and ii) it describes the semantic data or metadata itself as a result of this process” [Handschuh, 2005]. Of particular importance here is the notion of the *addition or association* of semantic metadata to *content*. Semantic annotations are typically indexed into a semantic search engineer for more efficient search and retrieval. The benefits of creating semantic annotations associated to CNL statements is that we can retrieve the context surrounding a CNL statement. Although, the CNL statement is stored as a fact in the knowledge base, its surrounding context is not. The same fact may have multiple contexts across documents. Traditional CNLs are not concerned with the surrounding context of a fact derived from a CNL statement.

6.1.1 Annotation as Authoring

As with any annotation environment, a major drawback is that in order to create metadata about a document, the author must *first* create the content and *second* annotate the content, in an additional *a posteriori*, annotation step. In the context of our application of CNL to semantic annotation, we seek to merge both authoring and annotation steps into one.

6.1.2 Towards a definition of Controlled Language for Semantic Annotation

We hereby refer to the application of CNL to semantic annotation as controlled (natural language) annotation which reflects how traditional CNL intersects but also differs from our approach, whereby:

Controlled (Natural Language) Annotation or Controlled Annotation is the application of CNL technologies to the process of semantic annotation. Controlled Annotation aims to reduce or eliminate ambiguity with respect to the semi-automatic/manual semantic

annotation of textual resources. It may include the creation of semantic data or metadata from machine processable content as in traditional CNL or apply CNL techniques that act as an interface to associate semantic data or metadata with free or uncontrolled text. Unlike traditional CNL, content in Controlled Annotation can be independent of the process.

The above definition is clarified based on the following categories. Each one is elaborated on below:

- **Degree of Control:** This refers to the degree of *restriction* with respect to the language used to author and annotate content. CNLs (although this may seem counterintuitive) in practice actually vary with respect to their degree of restriction on vocabulary and syntax. Usually a balance between reducing or removing ambiguity and user friendliness is sought. Controlled Annotation is similar in this respect in that a sentence may be syntactically constrained but may be more lenient with regards to the type of vocabulary that can be used i.e. (nouns or proper nouns), but CNL sentences or CNL snippets, (as we shall see in the section on design and implementation), may be embedded in free text. This overlaps with what Kaufmann describes as the **Formality Continuum** for Natural Language Interfaces [Kaufmann, 2007] and furthermore relates to Kuhn’s CNL design principle of *Clarity* [Kuhn, 2010a].
- **Knowledge Capture:** While CNLs usually focus on the creation of intensional, axiomatic or assertional knowledge statements, controlled annotation focuses on the creation of facts in the knowledge-base as well as the retention of provenance data and the maintenance of links or pointers to the original content. The creation of some intensional knowledge maybe permitted but this is reserved usually for the more confident or specialist user. In fact one may wish to prevent a casual user from altering an ontology at the class level as this could result in corruption of the ontology. On the other hand, one can argue that an annotation process with such ontology authoring features could serve as a quality assurance step for the ontology.

In other words, if an ontology is truly well designed and representative of the domain vocabulary, there should be very little need to alter it at the class level. The creation of rules or axioms during annotation is not common practice. Consequently, in order to apply controlled annotation to a textual resource, the annotation environment must be pump primed with an ontology consisting, at minimum, of intensional or axiomatic knowledge. This relates to Kuhn’s CNL design principle of *Expressivity* [Kuhn, 2010a].

- **Provenance:** This refers to the creation of metatadata to record data about the metadata/knowledge which has been captured: i.e. Where an RDF statement has come from? Who made it? When was it created? Which document and where is it located in the original (textual) resource - position and offset within the document or webpage, the name of the resource, date of creation and author. CNLs are traditionally not concerned with provenance, however Controlled Annotation is, since annotations will be indexed for semantic search.
- **Comprehension and Context:** A subtle difference between traditional Controlled Annotation and CNL is that for Controlled Annotation content can be independent of the process. We argue that content, which Controlled Annotation has been applied to, is independent of any facts that have been captured from it. Consequently, controlled annotated content is arguably implicitly understood by the average user regardless whether they are aware or not that the content was originally authored and annotated in a controlled manner. So a controlled annotated document will look reasonably like any other document on the Web, while on the other hand, a document written in CNL (although written in “natural language”) would still read as a collection of knowledge statements in natural language, which can be counterintuitive to the casual user. Hence, a CNL document should be viewed as a document in its own right i.e. subject to annotation and indexing into a semantic search engine. The benefit of retrieving (via semantic search) the por-

tion of text containing an annotated CNL statement is that the user can retrieve the surrounding *context*. This leads to a better information retrieval experience for the user. Although a processed CNL statement may not change in the knowledge base, its context will vary across documents even if that context contains other CNL statements. A related issue, argued by Smart et al [Smart, shed] is that despite the fact that knowledge statements can be described in CNL, users will still have the tendency to become “lost in logic”. Although a user is shielded by the underlying formalisms of some logic notation, they will still require some minimal introduction to formal logic to comprehend certain knowledge statements in CNL. However, this is an inevitable consequence of attempting to author an ontology from scratch, in that knowledge statements at the class and perhaps axiom level are necessary before instance population can proceed. Controlled Annotation is however, from a knowledge modelling perspective, a less complex task as it operates primarily at the instance level. In summary, Controlled Annotation sacrifices *Expressivity* over *Clearness* [Kuhn, 2010a].

- **Target User:** Fundamentally, the tasks differ when considering controlled annotation and CNLs for ontology authoring. The purpose of annotation is to associate metadata to content, which will subsequently be discovered by a semantic agent active on the Web. The provenance metadata will point to the original resources from which it was derived. The agent will retrieve this content for the human user. In short, the final benefactor is the human. In contrast, with respect to CNL for ontology authoring, the human benefits more indirectly. Although CNLs act as a user friendly interface to the ontology, it is the semantic agent which exploits the CNL generated machine-processable vocabulary. Hence the ontology is applied to semantic metadata by the agent to perform reasoning and search over semantically annotated Web resources. With respect to Controlled Annotation we argue that the target user is a typical Social Web user who is comfortable with CMS systems or Wikis. No knowledge of Semantic Web formalisms is expected. In contrast, with

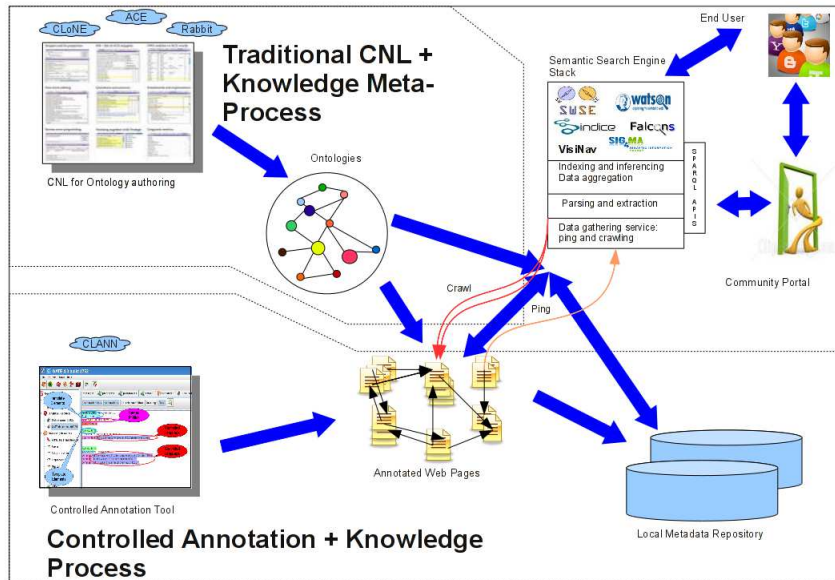


Figure 6.1: Controlled Annotation and the Semantic Web *Information Food Chain*

respect to CNLs, the user is a domain expert who will require some basic foundation in ontologies and formal logic in order to author an ontology in CNL. Figure 6.1 below describes the role of controlled annotation with respect to CNL. This visualisation incorporates what [Sure, 2003] describe as the duality of ontology and metadata, in other words (i)the knowledge meta-process - the development of an ontology and (ii) the knowledge process - the subsequent creation of a knowledge base. CNLs for ontology authoring assign themselves to the role of the knowledge meta-process - the creation, continued extension and adaptation of the ontology, while Controlled Annotation is concerned with the knowledge process - the steps for the creation and processing ontology based metadata. Figure 6.1 is also based on a variation of the *information food chain for the Semantic Web* described in [Decker, 2002].

6.1.3 Two approaches to Controlled Annotation - CLANN I and CLANN II

This chapter describes the design and implementation of **two** user friendly approaches to applying CNL to Semantic Annotation, which we call **CLANN - Controlled Language ANNotator** versions I and II, both of which are based on the Controlled Language IE software - CLIE (see Chapter 4). Both CLANN I and CLANN II permit non-expert users to semi-automatically and simultaneously author and annotate meeting minutes and/or status reports using controlled natural language. The motivation to develop two CLANN annotators was for exploratory purposes in order to assess which approach(or a combination of the two) would be more user friendly. Both CLANN annotators are essentially pathfinders to a final CNL annotator - CLANN III (See Chapter 7 for an initial description of CLANN III).

CLANN I is more automatic and aims to sacrifice expressiveness (with respect to controlling, manipulating and creating metadata) over naturalness while in contrast **CLANN II** prioritises expressiveness,(as in control over metadata manipulation), over naturalness. Uncovering the correct balance between expressiveness and naturalness is related to the *habitability* problem [Watt, 1968]. A Natural Language Interface(NLI) is considered habitable if users can express everything needed to complete a task using language they would expect the system to understand. The second aspect of the habitability problem, an aspect sometimes overlooked within the CNL community itself, is that of Chomsky's distinction between competence vs. performance [Chomsky, 1965]. Human linguistic competence can be described as a set of strict rules of a language's grammar(in this case English grammar) while performance consists of the uses we make of competence. In simpler terms, *how information is written using the grammar* is a measure of competence and *what information can be written using the grammar* is a measure of performance. We argue that the design of CNLs is often driven by competence while the second aspect of habitability states that an NLI should also attempt to account for both.

Although the majority of work within the field of CNL for knowledge creation has focused on ontology authoring, as far as we are aware, no other work has sought to

apply CNL to semantic annotation. This chapter makes **three** original contributions and addresses the research questions **SA1** and **SA2** in Chapter 1.

1. An novel approach to manual annotation, using CNL, called Controlled Annotation, where we move away from traditional a-posteriori annotation by merging both authoring an annotations steps together(**SA1**).
2. Two Controlled Language ANNotator - CLANN prototypes, each varying in expressiveness and usability(**SA1**) .
3. Statistical evidence that for certain scenarios, controlled annotation can be more user friendly than standard manual semantic annotation approach(**SA2**) .

Building on previous methodology [Funk et al., 2007], we undertook an evaluation, comparing both types of CLANN annotator with each other. Furthermore we investigated issues of usability and habitability for each tool’s respective CNL. We also included OntoMat [Handschuh et al., 2002], a standard manual semantic annotation tool. Our quantitative results demonstrate that controlled annotation can for certain use cases offer an attractive alternative to non experts over a standard manual annotation tool.

The remainder of this chapter is organised as follows: Section 6.2 discusses our use case and target domain and the design and implementation of both types of CLANN annotators and their corresponding CNLs. Section 6.3 presents our evaluation and discusses our quantitative findings. Section 6.4 discusses the current literature. Finally 6.5 Section offers conclusions, and as a prelude to Chapter 7, discusses further work involving controlled annotation.

6.2 CLANN: Design and Implementation

6.2.1 Use Case and Application Domain

The reader should note that controlled annotation cannot necessarily offer a panacea for manual semantic annotation as a whole since it is unrealistic to expect users to annotate

every textual resource using CNL. However there are certain use-cases in which CNL can offer an attractive alternative as a means for manual semantic annotation, particularly in contexts where controlled vocabulary or terminology is implicit such as health care patient records, business vocabulary and reporting. Our domain use case focuses on project administration tasks such as taking minutes during a project team meeting and writing weekly status reports. Very often such note taking tasks can be repetitive and boring. In our scenario the user is a member of a research group which is part of an integrated EU research project. We chose this domain because: (1) we observed the size of meeting minutes and status reports was quite limited and in addition sentences within such artefacts tended to be short, repetitious and more importantly tended to follow a “subject, predicate, object pattern”, making them good candidates from controlled annotation, and (2) we were able to construct our own small corpus of real-world meeting minutes and status reports generated over the three year period from the Nepomuk¹⁰⁷ project (271 reports).

One scenario could employ the use of pre-defined templates, whereby the user *simultaneously authors and annotates* his/her meeting minutes or status reports in CNL, using a semantic note taking tool - SemNotes¹⁰⁸, which is an application available for Nepomuk-KDE¹⁰⁹ - the KDE instance of the Social Semantic Desktop. The metadata would be available for immediate use after creation for querying and aggregation. The scenario is not limited to the KDE Desktop or the Semantic Desktop. Other scenarios could involve integrating controlled annotation into a Semantic Wiki.

Both CLANN tools are bootstrapped via the Nepomuk Core Ontologies¹¹⁰. More specifically our domain is modelled using a meeting minutes/status report ontology

¹⁰⁷<http://www.semanticdesktop.org>, Retrieved 2008-05-22

¹⁰⁸<http://smile.deri.ie/projects/semn>, Retrieved 2008-05-22

¹⁰⁹<http://nepomuk.kde.org/>, Retrieved 2008-05-22

¹¹⁰<http://www.semanticdesktop.org/ontologies/>, Retrieved 2008-05-22

MEMO¹¹¹, which references the users Personal Information Model Ontology(PIMO) ¹¹². The MEMO ontology was initially engineered for purposes of building both CLANN prototypes and was designed as a proof of concept. Since then we have completely redesigned and reengineered our domain ontology in accordance with proper ontology engineering methodologies, specifically the METHONTOLOGY [Fernandez-Lopez et al., 1997] methodology

We applied Word Smith Tools 5.0¹¹³ to our corpus in order to identify common linguistic patterns and vocabulary which could be applied to both CLANN annotators but more specifically CLANN I. We applied the results of our corpus analysis to the design of CLANN I in conjunction with linguistic introspection - a common requirements analysis step within language engineering. In addition, recall that CLANN I seeks to address some aspects of linguistic performance. Hence a further justification for corpus analysis. This will be elaborated where appropriate in detail within the next sections.

6.2.2 CLANN I and II Overview

As mentioned earlier this work focuses on two systems Controlled Annotation CLANN (Controlled Languages ANNotator) I and II. Both CLANN annotators are implemented in GATE¹¹⁴ and build on the existing advantages of the CLIE software and input controlled language, CLOnE, (see Chapter 4) and share the common features below

1. Both annotators require only one interpreter or runtime environment, the Java Runtime Environment.
2. As far as possible, CLANN I and CLANN II snippets are grammatically lax; in particular it does not matter whether the input is singular or plural (or even in grammatical agreement).

¹¹¹<http://ontologies.smile.deri.ie/2009/02/27/docs/>, Retrieved 2008-05-22

¹¹²<http://www.semanticdesktop.org/ontologies/2007/11/01/pimo/>, Retrieved 2008-05-22

¹¹³<http://www.lexically.net/wordsmith/version5/index.html>, Retrieved 2008-05-22

¹¹⁴General Architecture for Text Engineering, See <http://gate.ac.uk/>, Retrieved 2008-05-22

3. Both types of CLANN are relatively easy to learn by following examples and a small style guide, without having to study elaborate expressions of formal syntax.
4. Both CLANN annotators merge the authoring and annotation steps together.
5. Both annotators share a common template for meeting minutes.
6. Finally both CLANN annotators share a common ontology API similar to CLOnE.

In our scenario each CLANN annotator is anchored to existing semi-structured data such as a `AgendaTitle`, `Scribe` or `ActionItem` based on predefined meeting minutes or status report templates described (See Figure 6.2). The templates were constructed based on linguistic introspection and corpus analysis, which was conducted over the previously mentioned corpus of status reports and meeting minutes derived from the Nepomuk project. We choose to use a template for the CLANN prototypes for practical reasons, so the template could be easily applied to any note-taking environment or for instance emulated in a semantic wiki. For instance the application of extracting semantic metadata from wiki templates in Wikipedia was researched in [Auer and Lehmann, 2007]. Furthermore, it would be inefficient to apply CLANN for creation of this template rather the focus here is the usage of CLANN for authoring and annotating natural language sentences contained within in the template (See Figure 6.2).

6.2.3 GATE interface for CLANN I and CLANN II

At startup, CLANN I or CLANN II are loaded, including the initial textual data, into GATE. The user will have a window like the one shown in Figure 6.3 to work with. He/she can click on the buttons or tabs across the top to bring up the following panes.

Messages This pane explains in detail what a CLANN annotator has just done and includes error messages. The user can distinguish the error messages by the word *WARNING*, and probably ignore the *INFO* messages.

Meeting Date:<date>
Project Name:<project name>
Attendees:<attendee1>(,<attendee2>)+
Chair:<chair>
Scribe:<scribe>

Agenda Items:
Agenda Title:<title>
(Comment:<comment>.)+

RoundTable:
(Comment:<comment>.)+

Figure 6.2: Meeting Minutes Template for CLANN I and II

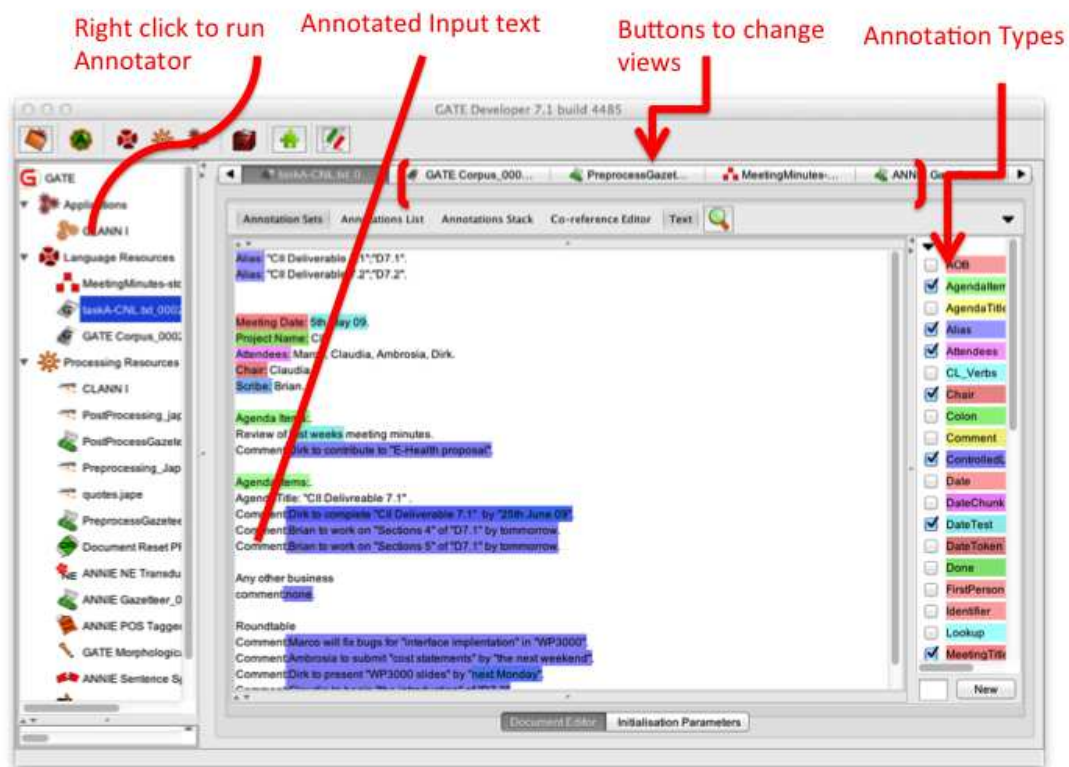


Figure 6.3: Overview of CLANN I and II in GATE

Text input In this pane (shown in Figure 6.3 the user can edit statements in the in either the CLANN I or CLANN II syntax, explained below. To clear any input, the user must select all the text with the mouse and press the backspace key. It is recommend to edit individual parts. This can be achieved as one would edit text normally using the arrow and backspace keys and the mouse.

Annotation This pane (shown in Figure 6.3 on the right hand side) shows you the semantic annotations created by either CLANN I or CLANN II. Example semantic annotations of type `Mention` can be viewed in both Figure 6.6 and Figure 6.9 respectively. See also Appendix B for details on how to inspect an ontology in

GATE .

To execute an annotator on the current input text, the user must right click either on the word *CLANN I* or *CLANN I* near the top of the left-hand pane and click *Run* in the menu that appears.

6.2.3.1 CLANN I: Design and Implementation:

The annotator architecture contains a standard GATE pipeline(see Figure 6.4) which includes the following language processing resources: The GATE English tokenizer, the Hepple POS tagger, a morphological analyser, a gazetteer list component for recognising useful key-phrases, such as structured elements from the templates and reserved CNL phrases. Any sentences for example, preceded by a **Comment:** element are considered candidates for controlled language parsing. Any remaining tokens from the CNL sentence which are not recognised as reserved CNL key-phrases are used as names to generate links to ontological objects. This is followed by a standard Named Entity(NE) transducer in order to recognise useful NEs, a preprocessing JAPE¹¹⁵ finite state transducer(FST) for identifying quoted strings, chunking Noun Phrases(NPs) and additional preprocessing. A second gazetteer list lookup is applied to identify trigger phrases associated with NEs which intersect with quoted and unquoted NP annotation spans. Additional feature values are then added to the NP chunks to indicate the appropriate class to link an NP chunk as an instance to. The last FST parses the CNL from the text and generates the metadata. The current tool is bootstrapped via the Nepomuk Core Ontologies and currently the application populates a meeting minutes/status report ontology MEMO which references a user's Personal Information Model Ontology(PIMO), via the GATE Ontology API. Each meeting minute note follows a pre-defined template(See Figure 6.2). The template is parsed initially to extract the inherent metadata about the meeting.

Each valid sentence in CLANN I matches exactly one syntactic rule and as mentioned

¹¹⁵Java Annotations Pattern Engine

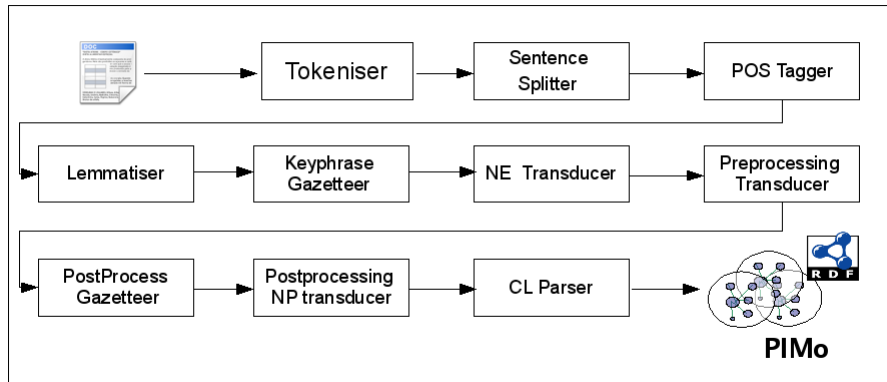


Figure 6.4: CLANN I pipeline

earlier consists of reserved key-phrases (verb phrases, fixed expressions and punctuation marks) as well as *chunks* (which similar to noun phrases are used to name instances). Similar to CLOnE, the language has *quoted chunks*, a series of words which are enclosed in quotes (Eg. "the PhD proposal"). Quoted chunks permit the capture of multi-word expressions as instances. They also permit the use of reserved words that would otherwise be detected by the reserved word gazetteer lookup.

An example syntactic rule is contained below:

- $\langle \text{NP} \rangle \langle \text{VP} \rangle (\langle \text{PREP} \rangle ? \langle \text{NP} \rangle) +$

where $\langle \text{NP} \rangle$ corresponds to *chunk* or *quoted chunk* and $\langle \text{VP} \rangle$ corresponds to reserved verb phrases (infinitive) and paraphrases derived from corpus analysis. Furthermore $\langle \text{PREP} \rangle$ corresponds to any preposition annotated using the POS tagger. Finally $(\langle \text{PREP} \rangle ? \langle \text{NP} \rangle) +$ matches one or more prepositional adjuncts i.e. "for the EU" or "in Work Package 3000". Hence the above rule would match the following sentences.

Comment: *Marco to visit "University of Karlsruhe".*

Comment: *Dirk to complete paper by "Sunday 21st June" for "International Semantic Web Conference".*

The above rule extracts the instances as arguments. The reader should note that

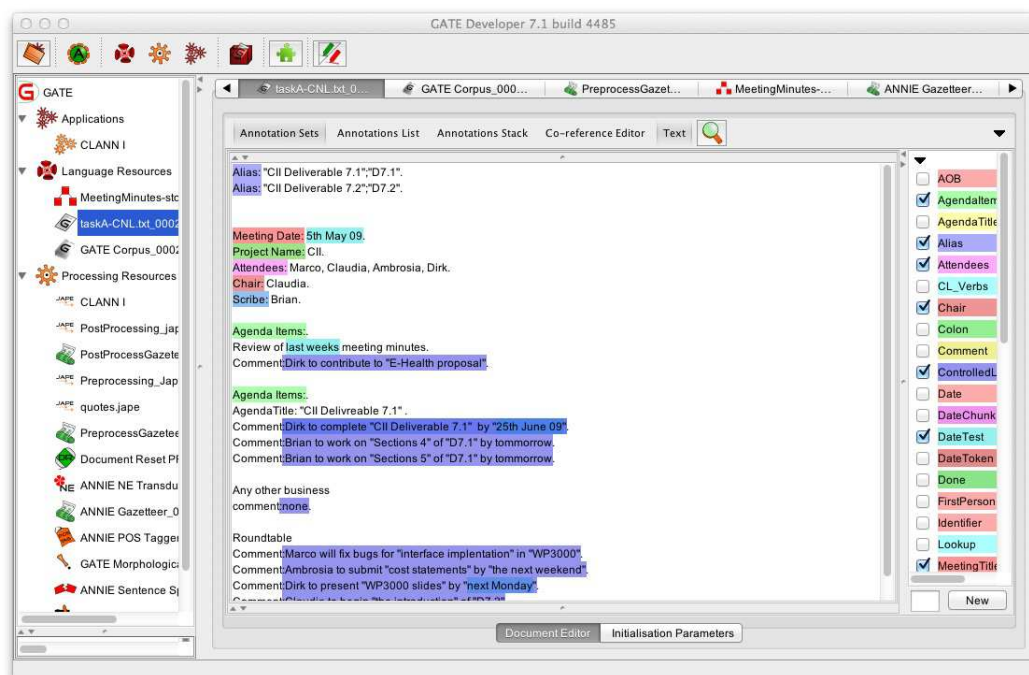


Figure 6.5: CLANN I visualised in GATE

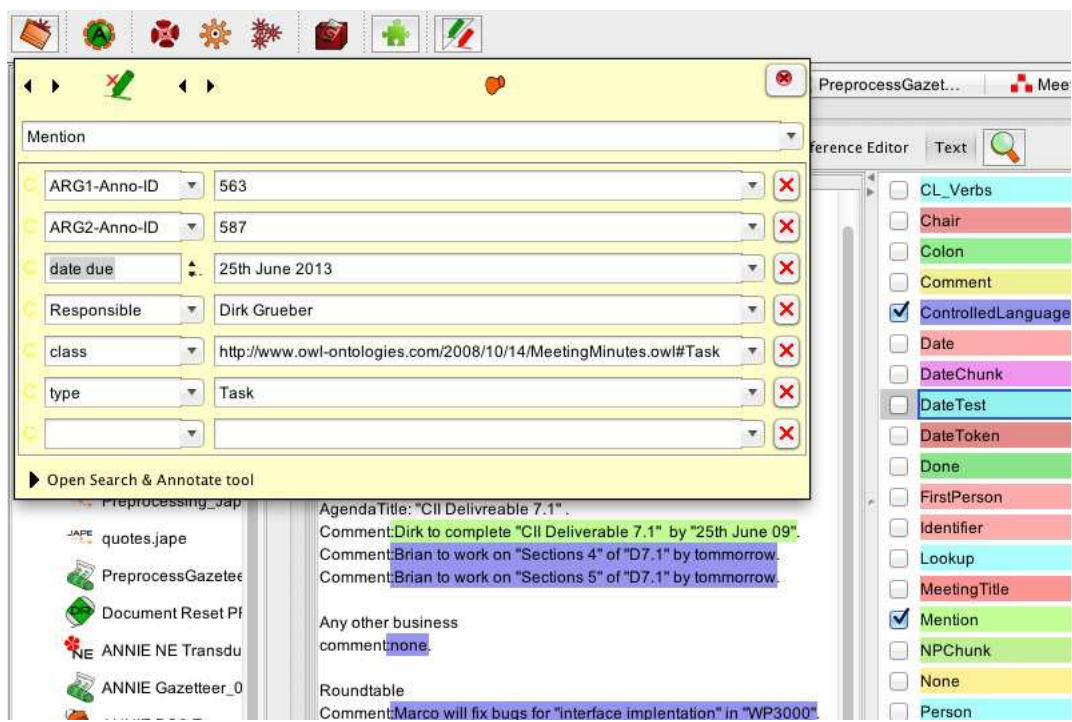


Figure 6.6: An example of a semantic annotation of type Task in CLANN I

prior to this stage that standard NE transducer and post-processing NP transducer (see Figure 6.4) will have collected additional information about each chunk. So *Marco* and *Dirk* are associated to a *Person*, while ‘‘Sunday 21st June’’ will be recognised as a *Date*. Using similar techniques ‘‘University of Karlsruhe’’ would be recognised as an *Organisation* and ‘‘International Semantic Web Conference’’ would be recognised as a *Conference*. The verb phrases *to visit* and *to complete* are then used to identify the relevant properties to link the instances recognised.

CLANN I also creates annotations of type *Mention* where each annotation contains a list of key/value features such as *class* and the corresponding URI for the resource in the ontology - <http://www.owl-ontologies.com/2008/10/14/MeetingMinutes.owl> \#Task (See Figure 6.6). The annotation is anchored to the chunk of text:

Comment: *Dirk to complete “CII Deliverable II” by “25th June 09”.*

Additional information such as *ARG1-Anno-ID=563* and *ARG1-Anno-ID=587* is also associated to the *Mention* annotation. Both feature values correspond to the unique numeric identifiers of the annotations, which mention *Dirk*, who is of type *Person* and ‘‘25th June 09’’, which is of type *Date*. Hence our *Mention* annotation takes two annotation identifiers as arguments, the person responsible for the *Task* and the completion date of the task. Note that, both annotations are within the same span as *Mention*. For additional details on the the interface and usage of CLANN I, we refer the reader to Appendix C.

Other features of CLANN I include simple co-reference using the *Alias:* rule, which allows the user to express the same instance in varied forms. It also enables the usage of a *shorthand* by the user when taking minutes.

CLANN I also incorporates simple elements of language performance in order to make the controlled language more habitable. See the introductory section for a discussion on the term habitability. In order to engineer elements of linguistic performance into CLANN I, we applied corpus analysis to generate lexicalisations for common properties within the MEMO ontology. These were then added to the the gazetteer list component within the

CLANN I pipeline as CNL reserved phrases. We applied mutual information statistics to word frequencies within our corpus to assess the strength of collocation relationships within text. We used the output to generate lists of common trigger phrases which could be aligned to properties within the MEMO ontology. For example:

Comment: *Dirk to complete paper by “Sunday 21st June” for “International Semantic Web Conference”*. can be paraphrased as:

Comment: *Dirk to finish up work on paper by “Sunday 21st June” for “International Semantic Web Conference”*. or

Comment: *Dirk to wrap up paper by “Sunday 21st June” for “International Semantic Web Conference”*. where **to finish up work on** and **to wrap up** are lexicalisations of the property *toComplete*. On initial inspection, the reader may be inclined to view such paraphrases as grammatically incorrect or stylistically inelegant, but recall that CLANN I is language performance driven and seeks to ease the user experience by incorporating such lexicalisations within the controlled language thus making it more habitable. In our evaluation section we will show how this design decision impacts favourably on user satisfaction. For additional examples of the CLANN I language and grammar, we refer the reader to Table 6.1.

6.2.3.2 CLANN II: Design and Implementation

The CLANN II architecture (see Figure 6.7 and Figure 6.8) is similar in design to CLANN I in that it shares the same language processing resources for tokenisation, sentence splitting, POS tagging and morphological analysis. CLANN II uses an identical template as CLANN I, however the **Comment:** element is non-existent and furthermore sentences themselves are not written in controlled language. In CLANN II the user can write any sentence without restriction under the heading of an **Agenda Item**. What differs in CLANN II is that the user can use snippets of controlled language to associate metadata to a particular piece of text. Snippets of CNL are identified within square brackets using [...]. The CLANN II CNL snippets themselves are similar to the CLOnE

Table 6.1: Excerpt of CLANN I grammar with examples

Sentence Pattern	Example	Parsed pattern
<NP><VP><PP>+	Ambrosio to submit "her PhD Proposal" during "the next week".	(Ambrosia <NP>) (to submit <VP>) ((her PhD Proposal <NP><PP>) (during (the next week <NP><PP>).
	Dirk to work on "the E-Health Proposal" with Ambrosia	(Dirk <NP>) (to work <VP>) (on(the E-Health Proposal<NP><PP>) (with (Ambrosia <NP><PP>).
ALIAS:<TEXT>;<ALIAS>	Alias:<"CII Deliverable 6.7">;<"D6.7">	Creates "D6.7" as an alias for "CII De- liverable 6.7".

language. A preprocessing finite state transducer(FST) similar to CLANN I is applied to extract values associated with template elements. In addition, text associated to the CNL snippets is also parsed at this stage. The final stage in the pipeline consists of a JAPE transducer which pulls the instances and properties to parse triples, ignoring the unassociated text. CLANN II shares the same API with respect to ontology manipulation as CLANN I and consults the ontology in similar manner. Similar to CLOnE and CLANN I, the language in CLANN II has *quoted chunks*, a series of words which are enclosed in quotes ("..."). This allows the user to associate metadata to more than one word.

Example syntactic rules are shown below: <TEXT>‘[’IS A CLASSNAME‘]’ where [IS A CLASSNAME] corresponds to a snippet of CNL. Hence:
Dirk[is a Person] to complete paper by "Sunday 21st June"[is a Date] for "International Semantic Web Conference"[is a Conference].

The rule below allows the user to simply embed a sentence in CNL in order to create relation metadata:

['CHUNK PROPERTY CHUNK']

This approach also allows users to handle adjuncts with much greater ease, such as associating the Date instance 'Sunday 21st June' with paper i.e.

["to complete" same as toComplete].

Dirk[is a Person] to complete paper[is a Document] by "Sunday 21st June"[is a Date] for "International Semantic Web Conference"[is a Conference].

[Dirk "to complete" paper].

[Paper hasEndDate "Sunday 21st June"].

Note, that when creating instances of properties, the controlled language will recognise pre-existing mentions i.e. paper and 'Sunday 21st June'. In order to use a property in the CLANN II CNL, the user must either use the appropriate label for the property on inspection of the ontology (in this case *to complete* is a part of the ontology) or alternatively they can use the alias preprocessing command to create a more natural substitute for the property.

CLANN II also creates annotations of type **Mention**, where each annotation contains a list of key/value features such as **class** and the corresponding URI for the resource in the ontology - <http://www.owl-ontologies.com/2008/10/14/MeetingMinutes.owl#Conference>. The annotation is anchored to the chunk of text "International Semantic Web Conference" (See Figure 6.9). The correct instance feature value for ISWC2009 will also be associated to the **Mention** annotation. For additional details on the the interface and usage of CLANN II, we refer the reader to Appendix C.

Another major difference between CLANN II and CLANN I is that the user can also create and manipulate classes, subclasses and class properties. Suppose the user is unsatisfied with the association of **paper** to **Document** and would prefer to associate the text to instance of a non existent class **ConferencePaper**. CLANN II permits the creation of new classes on an ad hoc basis using the following rules:

['<CHUNK> IS A SUBCLASS OF <CLASSNAME>']

resulting in the following:

Table 6.2: Excerpt of CLANN II grammar with examples

Sentence Pattern	Example
<TEXT>['IS A <CLASS>'].	Dirk[is a Person] Creates an object of the class Person with label <i>Dirk</i> . Note the the label is taken from the document content.
<TEXT>['IS A SUBCLASS OF <CLASS>'].	Proposal [is a subclass of Document] or [Proposal is a subclass of Document] Creates a new class with label <i>Proposal</i> as a subclass of the class <i>Document</i> .
<TEXT>['<PROPERTY> <OBJECT>'].	Dirk[to complete "PhD Proposal"] or [Dirk to complete "PhD Proposal"] Creates a triple which links the instances of <i>Dirk</i> and <i>PhD Proposal</i> with the property <i>toComplete</i> .

["Conference Paper" is a subclass of Document]

Dirk[is a Person] to complete paper[is a "Conference Paper"] by "Sunday 21st June"[is a Date] for "International Semantic Web Conference"[is a Conference].

[Dirk to complete paper].

We refer the reader to Table 6.2 for further examples of the CLANN II language.

Both CLANN I and CLANN II differ in the following ways:

- The CLANN II language is portable, while CLANN I must be re-targeted for a new

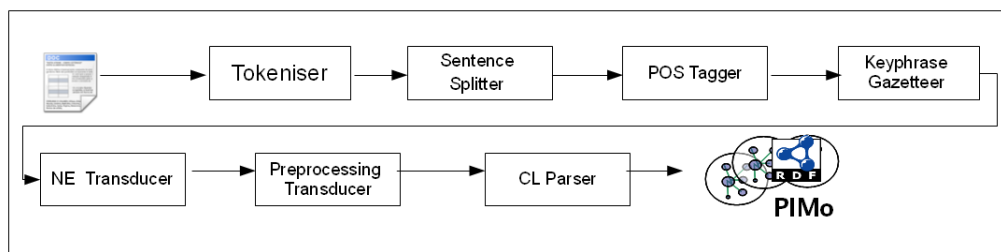


Figure 6.7: CLANN II pipeline

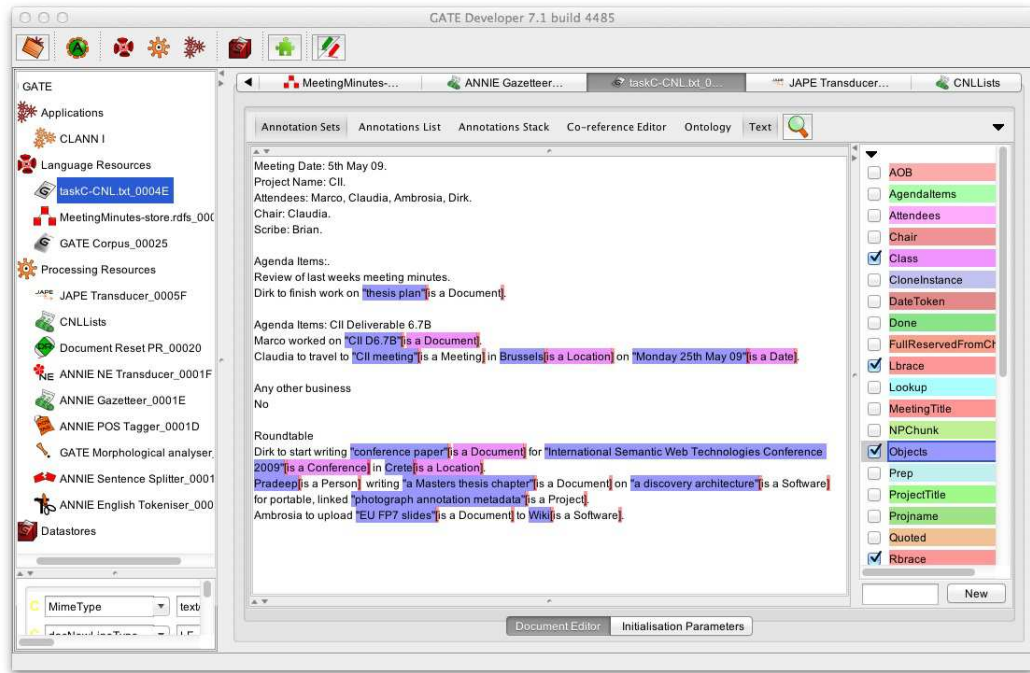


Figure 6.8: CLANN II visualised in GATE

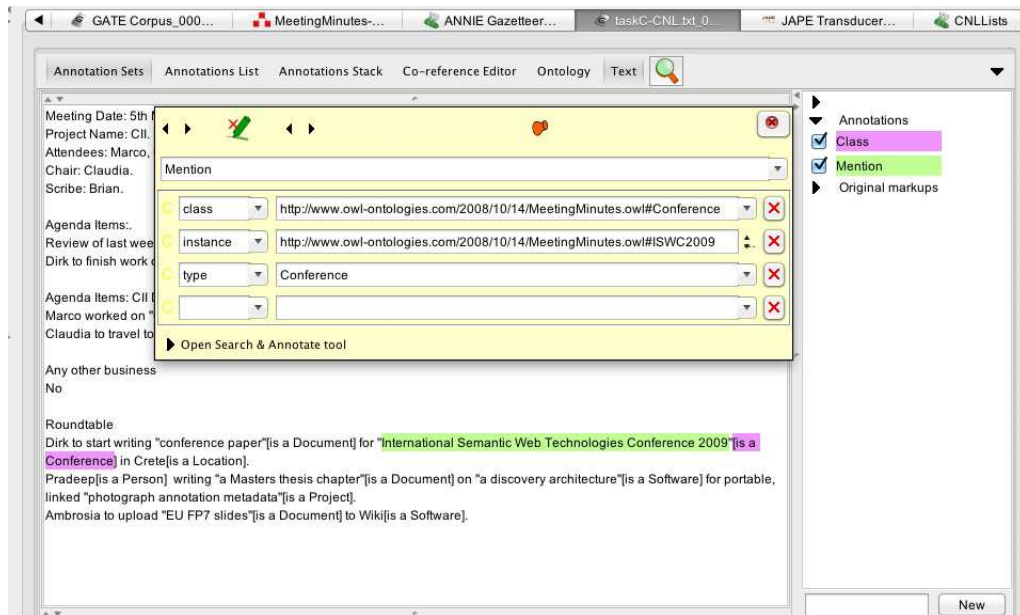


Figure 6.9: An example of a semantic annotation of type Conference in CLANN II

domain. This is an inherent disadvantage of attempting to cater for some linguistic performance in CLANN I.

- CLANN I incorporates some linguistic performance while CLANN II does not. Linguistic Performance is the result of conducting basic simple corpus analysis(mentioned earlier).
- CLANN I cannot create terminological component(TBox) statements. This option is available to CLANN II users. This raises interesting research questions: Should a proper annotation tool permit users to alter the ontology at the class level etc? Would annotators then invariably corrupt the ontology?
- CLANN II is more in line with the traditional spirit of semantic annotation. The CNL snippets act as the “glue” between free text and the knowledge base. This is similar to approach taken in some semantic wikis notably Semantic Media Wiki¹¹⁶ but the we believe that the CLANN II language is more natural and human readable.
- In theory, CLANN II can be applied to a legacy text which is not the case for CLANN I.
- The user must be somewhat familiar with the ontology to use CLANN II, while the ontology is hidden in CLANN I.

Other points in relation to CLANN vs CNL:

- The reader may feel that CLANN I is simply applied CNL for fact creation, however the subtle but crucial distinction here is that CLANN I retains provenance data regarding an extracted instance or relation which point back to original span of text and document source it from where it was extracted. Recall that we retain this data so it can be discovered at a later stage by a semantic agent on the Web or indexed into a semantic search engine. Provenance is arguably irrelevant for

¹¹⁶http://semantic-mediawiki.org/wiki/Semantic_MediaWiki

traditional CNL as the task differs and the role of the content is to interface with the ontology for authoring and population while with respect to CLANN, while modify the ontology, unlike traditional CNLs we also modify the text by creating annotations. Recall, the benefit of retrieving (via semantic search) the portion of text surrounding an annotated CNL statement is that the user can retrieve the surrounding *context*. This leads to improved understanding of search results by focusing the users attention on the anchor text captured by an annotation rather than then returning an entire document.

- The meeting minute is a human-readable account of a meeting and not merely CNL input for instance creation. Hence the annotated document when published on the Web should be implicitly understood by the casual user or should read naturally while as mentioned earlier in Section 6.1 some traditional CNL knowledge statements may seem unnatural or may even require some introduction to logic.

6.3 Evaluation

6.3.1 Research Questions

The aim of our evaluation is to answer the following research questions:

1. **SA1:** Can Controlled Annotation effectively substitute for a standard manual semantic annotation tool in certain scenarios?
2. **SA2:** Is Controlled Annotation more user friendly than standard manual semantic annotation tool in certain scenarios?

6.3.2 Methodology

Our methodology is based on the criteria used to evaluate CLOnE (See Chapter 4) which has previously proven reliable. The methodology involves a *repeated-measures, task-based* evaluation: each subject carries out a similar list of tasks on all tools being compared.

We undertook an evaluation, comparing CLANN I and CLANN II with each other. We also included OntoMaT [Handschuh et al., 2002], a standard manual semantic annotation tool, resulting in a three way comparison of tools. Furthermore we investigated issues of expressivity and naturalness between CLANN I and II in a post hoc manner. The evaluation material and results are available for inspection in Appendix C. The evaluation methodology involved the following across all three (CLANN I, CLANN II and OntoMaT) tools :

- A pre-test questionnaire of 6 questions asking each subject to test their degree of knowledge with respect to semantic annotation tools, ontologies, and controlled natural languages. It was scored by assigning each answer a value from 0 to 2.
- A short document introducing ontologies and semantic annotation (partly inspired by Protégé’s *Ontology 101* documentation [Noy and McGuinness, 2001] and [Handschuh, 2005] for semantic annotation respectively). Subjects were provided with reference guides and examples for both CLANN annotators and OntoMat. The reader should note that both CLANN annotators were not integrated into a specialised interface, rather the texts containing their respective CNLs were loaded into GATE as documents. Our research goals are concerned with exploring the user reaction to using a controlled annotation languages while authoring as well measuring the comparison between our approaches and the standard manual annotation paradigm.
- A post-test questionnaire for each annotation tool, based on the *System Usability Scale* (SUS), which produces a score of 0–100 to compare with previous results [Brooke, 1996].
- A comparative questionnaire similar to the one used in [Funk et al., 2007] was applied to measure each user’s preference for each of the tools over the other (i.e. CLANN I vs OntoMat and CLANN II vs OntoMat and CLANN I vs CLANN II) They were scored similarly to SUS so that for instance 0 would indicate a total preference for OntoMat, 100 would indicate a total preference for CLANN

I, and 50 would result from marking all the questions *neutral*. Subjects were also given the opportunity to make comments and suggestions.

- Three equivalent annotation tasks in the form of a meeting minutes note, each consisting of the following two subtasks:
 - create where necessary and associate instances within the text to classes,
 - create where necessary and associate instance properties within the text to instances.

For all three task lists, the same MEMO ontology was used. The same ontology was loaded into OntoMat for all three tasks. Again for both CLANN annotators the same ontology was used for all three tasks. See Appendix C for an example of all tasks. Note that the task text files contain no CNL language elements from either CLANN I or CLANN II. Templates elements were left partially incomplete across all three tasks.

6.3.3 OntoMat

OntoMat-Annotizer is an interactive webpage annotation tool [Handschuh et al., 2002]. It supports the user with the task of creating and maintaining ontology-based OWL-markups i.e. creation of OWL-instances, attributes and relationships. It includes an ontology browser for the exploration of the ontology and instances and a HTML browser that will display the annotated parts of the text. The intended user is the individual annotator. OntoMat allows the annotator to highlight relevant parts of a web page and create new instances via drag and drop interactions. It is representative of a conventional manual semantic annotation tool.

6.3.4 Sample quality

We recruited 18 volunteers. The sample size ($n = 18$) satisfies the requirements for reliable SUS evaluations [Tullis and Stetson, 2004]. We recruited subjects with both industrial

(**I**) and Academic (**A**) backgrounds. See (in Table 6.3) for details.

Table 6.3: Groups of subjects by source and tool order

Tool order	Background		Total
	Academia	Industry	
1-2-Ont	2	1	3
2-Ont-1	2	1	3
Ont-1-2	3	0	3
Ont-2-1	2	1	3
2-1-Ont	2	1	3
1-Ont-2	0	3	3
Total	11	7	18

In addition we attempted to control bias by selecting volunteers who were either:

- Research Assistants/Programmers/Post-Doctoral Researchers with an industrial background either returning (or new) to Academic Research respectively(**I**),
- Postgraduate Students who were new to the Semantic Web and unfamiliar with Ontology Engineering or Semantic Annotation(**A**),
- Researchers with no background in Natural Language Processing or Ontology Engineering(**A**) or
- Industrial Collaborators (**I**).

In all cases, we ensured that participants had limited or no knowledge of GATE or OntoMat. First, subjects were asked to complete the pre-test questionnaire, then they were permitted time to read the initial Ontology and semantic annotation guide, as well the reference guide for tool in question, and lastly they were asked to carry out each of the three annotation tasks with a different tool. For instance, a sixth of users(three users) carried out task document A with CLANN I, then task document B with CLANN

II and finally task document C with OntoMat; the second group of three users undertook task document A with OntoMat, then task document B with CLANN I and finally task document C with CLANN II and so on. The process of varying the tool order resulted in six permutations overall (See Table 6.3). Each user's time for each task document was recorded. After each task the user completed the SUS questionnaire for the specific tool used, and finally a comparative questionnaire depending on the preceding tool. Comments and feedback were also recorded on the questionnaire forms.

6.3.5 Hypothesis statements

With respect CLANN I and CLANN II and OntoMat SUS scores, we state the our first hypothesis pair as:

5.1H₀: It is predicated that mean usability scores (SUS) for either CLANN I, CLANN II and OntoMat will not differ significantly.

5.1H₁: It is predicated that at least one mean usability score (SUS) for either CLANN I, CLANN II and OntoMat will differ from the other.

In addition, with respect CLANN I and CLANN II and OntoMat SUS times we have the second hypothesis pair:

5.2H₀: It is predicated that mean task times for either CLANN I, CLANN II and OntoMat will not differ significantly.

5.2H₁: It is predicated that at least one mean task time for either CLANN I, CLANN II and OntoMat will differ from the other.

6.3.6 Results and Data Analysis

Table 6.4 and Table 6.8 summarises the main measures obtained from our evaluation. We used SPSS¹¹⁷ to generate all our statistical results.

In particular the mean CLANN I SUS score is above the baseline of 65–70% while

¹¹⁷SPSS 15.0, <http://www.spss.com>, Retrieved 2013-05-22

Table 6.4: Summary of the questionnaire scores

Measure	min	mean	median	max	confidence interval
Pre-test scores	0	4.28	4.5	10	2.7 - 5.8
CLANN 1 SUS rating	22.5	69.2	71.3	90	61.8 - 76.5
CLANN 2 SUS rating	5	55.6	61.3	85	46.6 - 64.5
Ontomat SUS rating	0	32.1	25	75	22.1 - 42.1
CLANN 1 Vs Ontomat Preference	42.5	67.5	65	100	60.1 - 74.9
CLANN 2 Vs Ontomat Preference	37.5	62.2	58.8	92.5	54.6 - 69.8
CLANN 1 Vs CLANN 2 Preference	30	54	56.3	80	47.8 - 60.3

Table 6.5: Summary statistics for tool times

Tool	min	mean	median	max	confidence interval
CLANN I	2	9.33	8	25	6.4 - 12.3
CLANN II	3	13.17	14	22	10 - 16.3
Ontomat	15	24.28	25	25	23.2 - 25.4

the mean SUS score for OntoMat is well below the baseline. In the CLANN I/OntoMat Preference scores, based on the comparative questionnaires, we note that the scores also favour on average CLANN I over OntoMat. Furthermore with respect to CLANN II the mean SUS score is relatively neutral. In addition, although the CLANN II/OntoMat preference score is high, it is not above the SUS baseline. Interestingly the CLANN I/-CLANN II preference is neutral, indicating that users were undecided with regard to the preference of one CLANN tool over the other. Finally the mean Pre-test score is relatively neutral, indicating no particular bias with respect to prior knowledge. Confidence intervals are also displayed in Table 6.4.¹¹⁸

We conducted a one-way repeated measures ANOVA was conducted to compare SUS scores and times across each tool. With respect to:

5.1H₀: It is predicated that mean usability scores (SUS) for either CLANN I, CLANN II and OntoMat will not differ significantly,

we can reject the null hypothesis as the compared groups differed significantly, $F(2,34) = 21.605$, $p < 0.5$, multivariate partial eta squared =.55.

Furthermore, with respect to:

5.2H₁: It is predicated that at least one mean task time for either CLANN I, CLANN II and OntoMat will differ from the other,

we can reject the null hypothesis as the compared groups differed significantly, $F(2,34) = 45.163$, $p < 0.5$, multivariate partial eta squared =.72.

We note that, when engaging in tasks with OntoMat, 14 out of 18 participants requested to withdraw from their assigned annotation tasks due to waning interest. Hence, we choose to penalise the participants by assigning them the maximum time of 25 minutes.

For post-hoc data analysis, we also generated Pearson's coefficients [John L. Phillips, 1996]. Table 6.6 displays the most important coefficients. We note the most important results:

- There is a moderate correlation between the CLANN I SUS score and the CLANN

¹¹⁸A data sample's *95% confidence interval* is a range 95% likely to contain the mean of the whole population that the sample represents [John L. Phillips, 1996].

Table 6.6: Correlation coefficients

Measure	Measure	Pearson's	Correlation
CLANN 1 time	Ontomat time	-0.12	none
CLANN 2 time	Ontomat time	-0.19	none
CLANN 1 time	CLANN 2 time	0.61	strong +
CLANN 1 SUS score	CLANN 1 Vs Ontomat	-0.02	none
CLANN 1 SUS score	CLANN 1 Vs CLANN 2	0.49	moderate +
CLANN 2 SUS score	CLANN 2 Vs Ontomat	0.25	none
CLANN 2 SUS score	CLANN 1 Vs CLANN 2	-0.03	none
Ontomat SUS score	CLANN 1 Vs Ontomat	-0.43	weak -
Ontomat SUS score	CLANN 2 Vs Ontomat	-0.48	moderate -
CLANN 1 Vs Ontomat	CLANN 1 Vs CLANN 2	0.5	none
CLANN 2 Vs Ontomat	CLANN 1 Vs CLANN 2	-0.1	none
CLANN 1 Vs Ontomat	CLANN 2 Vs Ontomat	0.65	strong +

I/CLANN II preference score, with an associated probability of 5%, however the value of the correlation itself is not particularly significant.

- There is also a moderate correlation between the CLANN II SUS score and the OntoMat SUS score, with an associated probability of 5%, however the value of the correlation itself is again not particularly significant.
- There is a relatively strong positive correlation between CLANN I task times and CLANN II task times and furthermore they are not significantly related.
- However there is a strong positive correlation between CLANN II/OntoMat preference scores and CLANN II/OntoMat preference scores and furthermore they are significantly related, $r=+.65$, $n= 18$, $p<.01$, two tails. (Note, alpha was actually adjusted to .004 from .05 as per the Bonferroni correction¹¹⁹ to cater multiple cor-

¹¹⁹See http://en.wikipedia.org/wiki/Bonferroni_correction, Retrieved 2013-05-22

relations) however p was still below this value).

Hence, we can infer that users with little or no experience or knowledge of controlled languages, semantic annotation or ontologies who favour CLANN I over a standard manual semantic annotation tool will also tend to prefer CLANN II to a standard manual semantic annotation tool.

A comparison of SUS scores against user background is shown in Table 6.7. Interestingly users with an industrial background scored CLANN I higher than academics by an additional 5 points. Furthermore, industry type users scored CLANN II higher than academics by an additional 8 points. With respect to the CLANN I/OntoMat preference scores, industrial users scored an extra 8 points higher than academics for CLANN I. Furthermore, users of an academic background scored 3 points higher than industry types in the CLANN II/OntoMat comparative questionnaire. Finally, the most interesting observation is the CLANN I/CLANN II preference score, whereby industrial users scored their questionnaires roughly 13 points higher than academic users. More importantly, the relatively neutral CLANN I/CLANN II preference score of 48.9 is inconsistent with the high mean SUS score of 67.3 provided by academics for CLANN I. This implies that while academics liked using CLANN I, when asked to score preference of CLANN I and CLANN II, they were undecided. This was not the case for industrial users.

Finally, a comparison of tasks times against backgrounds is available in Table 6.8. Academic users were in general slightly faster than industrial users with respect to both CLANN tools. In general users were 3-4 minutes faster using CLANN I, depending on their background, and in general OntoMat was the most time consuming tool averaging 23-25 minutes regardless of the user's background. This is due to the fact that 14 participants out of 18 requested to withdraw from their assigned annotation tasks due to waning interest. Consequently, we penalised the subjects by assigning them the maximum time of 25 minutes. We also note that the 14 participants, who withdrew while using OntoMat, did not complete their respective annotation tasks fully. However, we ensured that they created sufficient metadata to complete the core tasks of creating or linking instances and

Table 6.7: Comparison of SUS scores against backgrounds

Measure	Background	mean	median	standard deviation
Pre-test scores	Academia	3.3	2	3.3
	Industry	5.7	5	2.9
CLANN 1 SUS rating	Academia	67.3	70	18.9
	Industry	72.1	75	10.2
CLANN 2 SUS rating	Academia	52.5	50	22.7
	Industry	60.4	62.5	12.4
Ontomat SUS rating	Academia	28.4	22.5	20.4
	Industry	37.9	35	23.8
CLANN 1 Vs Ontomat Preference	Academia	64.3	62.5	14.8
	Industry	72.5	70	17.9
CLANN 2 Vs Ontomat Preference	Academia	63.4	60	16.4
	Industry	60.4	57.5	17.7
CLANN 1 Vs CLANN 2 Preference	Academia	48.9	52.5	10.7
	Industry	62.1	65	14.2

instance properties. Of those 14 subjects that withdrew, 9 were academics and 5 were from an industrial background.

6.3.7 User Feedback

The test users also provided several suggestions/comments about both CLANN annotators:

- Many subjects requested a feature-rich graphical editor to assist the annotation process i.e. grammar correction, syntax highlighting and auto-completion. This, they argued would improve the speed of annotation and hence the usability of the tools.

Table 6.8: Comparison of task times against backgrounds

Measure	Background	mean	median	standard deviation
CLANN 1 time	Academia	9	8	5.2
	Industry	9.9	4	8.5
CLANN 2 time	Academia	12.5	12	7.7
	Industry	14.3	14	5.7
Ontomat time	Academia	24.5	25	0.6
	Industry	23.4	25	3.7

- Most users requested features specific to the CLANN I editor include automatic template generation, auto-suggestion of verb phrases and instance labels and smart examples assisting the user.
- Most subjects requested features specific to the CLANN II editor included: automatic or assisted generation of snippets, machine learning capabilities to automatically recognise instances and an ontology visualisation layer.

6.3.8 Discussion

Our user evaluation consistently indicated that our subjects found CLANN I significantly more usable than OntoMat for annotation tasks. In addition, we observed that participants took significantly more time completing annotation tasks using OntoMat in comparison to CLANN I or CLANN II. Hence our observations address our research questions (See SA1 and SA2 in Chapter 1) that for certain scenarios, CNLs can effectively substitute for manual semantic annotation and are also considered more user friendly. In addition post-hoc analysis revealed novice users who preferred using CLANN I over OntoMat also tended to prefer using CLANN II over OntoMat. Our data also revealed that this observation was statistically significant. Although, CLANN I SUS scores were higher than CLANN II SUS scores, the CLANN I/CLANN II preference scores were scored as

neutral and/or undecided and were inconsistent with the CLANN I SUS score. Hence it is unclear which CLANN annotator was preferred by our sample users. However, interestingly, we noticed that industrial users in general had high usability scores for CLANN I consistent with CLANN I/CLANN II preference scores. The reader should note that the ontology is completely hidden in CLANN I, requiring no interaction in order to create metadata. It is possible that industrial users appreciated CLANN I's tendency towards habitability because of its corpus driven design and subsequent language performance features. In contrast, although academics gave high usability scores for CLANN I, their CLANN 1/CLANN 2 preference scores were inconsistent with this, indicating that they were undecided with respect to which CLANN tool they preferred. Again recall that CLANN II is more expressive and powerful with regard to metadata creation and manipulation and is less habitable. The tool also requires inspection of the ontology. It is possible that academic users may have had a conflict with respect to deciding between a more formally expressive CNL and more a habitable CNL. Hence industrial users may favour habitability over expressiveness regarding metadata manipulation, while academics would perhaps rather sacrifice habitability for control over metadata and the ontology. We speculate that this may be also the case across industrial and academic groups within the true population of novice users, however our sample size is too small to make any statistical inferences about the population with regard to two subgroups.

6.3.9 Limitations of the evaluation

We compared OntoMat with both CNL annotators, because OntoMat is a standard tool for manual semantic annotation. Furthermore OntoMat permits a user to easily import an external ontology. In addition, OntoMat was capable of emulating the structural annotation associated with the meeting minutes template used by both CLANN annotators. Another possible candidate was Semantic Media Wiki. However, firstly there are known issues with respect to importing external ontologies. Secondly and more importantly, Semantic Media Wiki associates metadata to wiki links and not text directly and

was consequently not directly comparable to controlled annotation¹²⁰. GATE was also a possible choice, but the Ontology Annotation Tool(OAT) in GATE is not as explicit as OntoMat with respect to annotating relational metadata¹²¹ and in addition as CLANN is implemented in GATE, comparing it to OAT would have biased the evaluation.

With respect to the SUS questionnaire, we acknowledge that its high level nature makes the questionnaire quite general, however it is widely used, technologically agnostic and flexible, making it very suitable for general usability assessments. Although the majority of the participants were native speakers, we did not explicitly test for English proficiency among our sample users (as is the case in other work [Dimitrova et al., 2008]). This is due to the fact that knowledge capture task is less complex than typical CNL tasks in that our users were not required to author axiomatic or intensional knowledge in CNL.

Efforts were made to exclude individuals with knowledge of ontologies and semantic annotation. One user had been exposed to GATE before, however our pretest scores were quite neutral. We argue that this was sufficient as a pilot study.

6.3.10 Annotation Metrics

It is important to understand that by comparing information extraction metrics across each tool, it results in an unfair comparison with respect to Ontomat. This is because CLANN I, by its restricted nature, is deterministic and would always have 100% precision and recall. Hence the focus of our evaluation was on usability. The participants were given the meeting minutes in free text for all three tools. The sentences were examples taken from the aforementioned corpus we had gathered.

For both CLANN Tools in GATE:

- Where GATE indicated parsing errors in the message viewer, users were asked to rewrite the statements.

¹²⁰At the time evaluation, the SMWWriter API had not been developed. See <http://semantic-mediawiki.org/wiki/Help:SMWWriter>, Retrieved 2013-05-22

¹²¹This has improved in GATE 6.0

Table 6.9: Summary of the Precision and Recall scores for each tool

Tool	Precision	Recall	F-measure
CLANN I	1	1	1
CLANN II	.9	1	.94
Ontomat	1	.53	.69

- We checked that the ontology was correctly populated after each user, which was the case.

In the case of CLANN II:

- Since users were given some freedom with respect to CLANN II, we noticed that they sometimes linked to a super class rather than to the class instructed.
- For instance one user linked a chunk of text to the *Document* superclass instead of the class *Deliverable*. This is not incorrect rather inexact.
- The above mentioned “error” occurred with 40% of users.

Finally, with respect to Ontomat:

- 14 out 18 participants requested to stop their task, usually after 5-10 minutes.
- We ensured that for each user who withdrew, they completed at least 7/18 annotations, correctly.
- The other remaining 4 participants completed there annotations correctly.

Table 6.9 summaries the standard precision and recall metrics for each tool.

6.4 Related work

A plethora of tools exist for the manual or (semi-)automatic semantic annotation of free text. For a thorough survey of manual and (semi-)automatic semantic annotation

tools and platforms, we refer the reader to Chapter 3, Section 3.2. In addition, the idea of blurring the lines between authoring and annotation, has its origins in the CREAM (CREATING Metadata) framework for semantic annotation [Handschuh and Staab, 2002]. However, the implementation is simplistic and implies dragging an RDF Label for a given concept from the ontology viewer and essentially pasting the label into the document. To our knowledge, however, very little research exists involving the application of CNL to semantic annotation. For a thorough review of CNL in the context of ontology authoring, we refer the reader to Chapter 3, Section 3.1.

The most closely related technologies to controlled annotation (specifically the CLANN II syntax) are semantic wikis, which have become a somewhat popular way of adding semantics to user generated wiki pages. The term semantic wiki often implies either ontology authoring or the semantic annotation of wiki content. A traditional wiki creates links between pages without defining the kind of linkage between pages. Semantic Media Wiki [Krötzsch et al., 2006] allows a user to define the links semantically, thereby adding meaning to links between pages. Each concept or an instance has a page in Semantic Media Wiki (SMW), and each outgoing link from this page is annotated with well-defined properties as links. However this kind of approach differs to the kind of semantic annotation that we aim for. The Semantic Media Wiki model forces the users to use the wiki pages for content creation and to create a new page for each instance¹²². Moreover, the relational meta-data represented in a Semantic Media Wiki always has the corresponding page as its subject, thereby restricting the creation and description of other relevant entities.

With respect to user evaluation, [Krötzsch et al., 2007] describe observations regarding SMW usage. They state first and foremost that the ‘majority of users will neglect annotation as it does not bear immediate benefit’. This is understandable given any annotation context, whereby the benefits of annotation are not recognised until the semantic search

¹²²At the time of evaluation, the SMWWriter API had not been developed <http://semantic-mediawiki.org/wiki/Help:SMWWriter>, Retrieved 2013-05-22

stage. In addition, they argue that “without conclusive studies on the usage of wikis in general, any prediction on the effect of introducing semantics in the (wikipedia) environment lacks justifications”. In [Krötzsch et al., 2007], the authors base their wiki usage experiments on `ontoworld.org`, which is itself maintained by the authors. The site’s function is to collect information about semantic web researchers, events and projects. The authors record 930 registered users, the majority of which have contributed little to the total recorded 37,880 edits. The semantic knowledge base of `ontoworld.org`, at the time, counted 17,562 property annotations for 808 property edits. The majority of the properties have a page in the wiki, while 50% are of type `Page` and are used to annotate hyper-links. With respect to the usage of properties, they noted that 5% of the properties accounted for over 74% of the annotations, whereas the least used 80% of the properties accounted for less than 9% of all semantic statements. The authors state that these results have very similar power-law distributions to those of Wikipedia’s categories [Krötzsch et al., 2007]. In conclusion, they argue that SMW features are at least equivalent to Media Wiki functionality, however as the authors themselves state “one cannot conclude whether or not the requested (annotation) functions are actually considered useful for a given purpose” and that additional research is needed to obtain definitive results. While the research is very important in that it records observations with respect to SMW usage over a large user populations, one cannot conclude any specific user satisfaction rating with respect to the users and further more the user group is arguably extremely biased to that of the semantic web community.

Recent work by Pfisterer et al [Pfisterer et al., 2008] reports better results, but it is in the context of the interface extensions to SMW by AIFB¹²³ in collaboration with Ontoprise GmbH¹²⁴. The interface enhancements include: (1) a fact-box which summarises all facts, linked to a given article, (2) a semantic query interface with strong auto-completion

¹²³Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB) <http://www.aifb.kit.edu>, Retrieved 2013-05-22

¹²⁴<http://www.ontoprise.de/>, Retrieved 2011-05-22

features, (3) an ontology browsing interface and (4) a Semantic Tool-Bar, which seeks to ease the semantic annotation process, but a classical a posteriori fashion and not at the editing/authoring stage. The Semantic Tool-Bar, which is enabled by software derived from the HALO¹²⁵ project, allows users to add/change annotations, whereby the changes are written directly to the wiki source text. They conducted two evaluations, whereby the shared scenario is the creation of a scientific Semantic Wikipedia. They recruited seven test subject matter experts consisting of two experts in physics, three in chemistry and two in biology. The experts had little or no familiarity with semantic wikis and had never edited a Wiki article before the evaluation. They participated in the design and development of the enhanced SMW over a period of seven months. With respect to both evaluations, they are more aligned to usability testing rather than being able to make and statistically significant inferences about the general target user population regarding the usability of their Wiki. A SUS[Brooke, 1996] questionnaire was administered to the group, once before and once after user feedback had been integrated into the enhanced Wiki. The user satisfaction was low prior to the enhancements and high, upon re-administration of the questionnaire. The sample size, at seven, was too small to make any statistically significant claims. More importantly, as the authors note, there are flaws in the evaluation in that a portion of first questionnaire group intersected with the second questionnaire group. This portion had been exposed to the enhanced SMW for a few months, so the high user satisfaction observed is inaccurate.

With respect to the second of the two aforementioned user evaluations, forty-two students from an introductory human-computer interaction class served as the sample user population. Each subject, after being provided introductory material, were asked to annotate a random wiki page in the enhanced SMW. In addition, after completing this task they were asked to formulate a number of queries to the SMW. This was followed by a SUS usability questionnaire. The resulting user satisfaction score was below the SUS baseline at 54.8%. In addition, on average each student created 4.2 annota-

¹²⁵<http://www.projecthalo.com/>, Retrieved 2011-05-22

tions and only 50.3% were fully correct. Errors were caused primarily by unrecognised characters or date formats, which are rectifiable. The speed of the systems reaction accounted for a large amount of negative feedback. However, performance limitations in speed were undoubtedly caused by over 20 users editing the SMW at the same time. This would invariably have had an impact on user satisfaction scores. Nevertheless, no statistical tests are performed on the SUS results and no inferences are made about the general target population. Despite the weak empirical results, the work presented in [Pfisterer et al., 2008] is very important in that it represents a shift towards proper user evaluation and user centred design within the semantic wiki community. The authors themselves acknowledge that there is still a need to provide more “concrete examples” with respect to application of user centred design to semantic wikis.

Other flavours of semantic wikis include IkeWiki¹²⁶[Schaffert, 2006] and KiWi¹²⁷. One could argue that Semantic Wikis are only usable by the Semantic Web community and retain a significant formal barrier to a casual user or even an IT professional in the industry. ACEWiki[Kuhn, 2008] attempts to circumvent this using the CNL ACE in combination with a predictive editor as an interface to a Semantic Wiki. Other work involves integrating the Rabbit CNL (as well as support for ACE) into Semantic Media Wiki The purpose of which is user friendly collaborative ontology authoring using multiple CNLs and template based language generation capabilities [Bao et al., 2009]. However, in both cases, the task here is collaborative ontology authoring and not (controlled) annotation, which seeks to provide wiki content with a semantic backbone.

6.5 Conclusion and Future Work

The research contributions of this chapter can be summarised as follows:

1. An novel approach to manual annotation, using CNL, called Controlled Annotation, where we move away from traditional a-posteriori annotation by merging both

¹²⁶<http://ikewiki.salzburgresearch.at/>, Retrieved 2011-05-22

¹²⁷<http://www.kiwi-project.eu/>, Retrieved 2011-05-22

authoring an annotations steps together(**SA1**).

2. Two Controlled Language ANNotator - CLANN prototypes, each varying in expressiveness and usability(**SA1**) .
3. Statistical evidence that for certain scenarios, controlled annotation can be more user friendly than standard manual semantic annotation approach(**SA2**) .

One could argue that our notion of merging authoring and annotation is not novel. Semantic Media Wiki also merges both authoring and annotation steps. However we believe that an annotation syntax (particularly in the context of CLANN II) should be as close to natural language as possible. So one could view the comparison as an annotation syntax problem. We argue that, for semantic wikis, the use of a formal syntax to link to facts to wiki pages represents a significant barrier to non-semantic users. This is supported by the weak results of user studies reported in the section on related work. Furthermore, in the context of the enhanced Semantic Media Wiki presented in [Krötzsch et al., 2007], the “ annotation as authoring” approach or standard semantic annotation syntax of the Wiki, is essentially abandoned for a Semantic Tool-Bar, which takes a traditional drag and drop a posteriori approach similar to a standard manual annotation tool. The purpose of this is to shield users from the formal annotation syntax of the Wiki. Despite this, the user satisfaction results are not favourable. In summary, there are still open questions with regard to the usability of Semantic Media Wiki, particularly in the context of creating annotation either at the authoring stage or following content creation.

Although the JAPE parsing process is quite robust, the parsing error messaging provided in GATE is insufficient. Clearly, it is evident that a proper interface is required to improve the usability of both CLANN annotators as they are quite prototypical. A possible CLANN III prototype would use link grammar¹²⁸. GATE is ideal for rapid prototyping in this context but the JAPE language has disadvantages inherent to finite-state parsing. Link grammar has more recognition power (particularly due to its context free

¹²⁸<http://www.link.cs.cmu.edu/link/>, Retrieved 2011-05-22

parsing capabilities allowing us to model the reification of triples properly) and furthermore it would provide CLANN with free predictive parsing/editing capabilities to improve usability. These topics along with other future work will be discussed in the following Chapter.

Part IV

Conclusion and Future Work

7 Conclusions and Future Work

This thesis began with the statement that both Ontology Authoring and Semantic Annotation are two of the core challenges for building the Semantic Web. We have shown how Controlled Natural Languages (CNL)s can assist in easing the knowledge acquisition bottleneck, which is inherent to both of the above challenges. We have presented a user friendly means for ontology authoring using a lightweight CNL called CLOnE. We have also presented an enhanced version of the tool, which augments the user friendliness of CLOnE using shallow Natural Language Generation (NLG). The addition of shallow NLG to CLOnE, produces a RoundTrip Ontology Authoring (ROA) environment. Finally, we have provided prototypes for user friendly semantic annotation based on CNL, called Controlled Language Annotator - CLANN, which intersects the fields of CNL with Semantic Annotation. The resulting output is Controlled Annotation, which is a type of latent annotation, whereby both authoring and annotation steps are merged into one as opposed to classic a posteriori annotation.

This concluding chapter is divided into five sections: (1) we revisit the research contributions of this thesis, including (2) open questions, (3) ongoing work with respect to evolving Controlled Annotation and (4) additional future work.

7.1 Research Contributions

The contributions of this thesis can be categorised as follows:

- **Round Trip Ontology Authoring** is the addition of a shallow NLG generator to the CLOnE implementation. **Controlled Language for Ontology Editing**

- **CLOnE** is a compact CNL which enables non-experts to solve basic ontology editing tasks. CLOnE was implemented in the GATE framework by the Sheffield NLP group and is based on GATEs linguistic analysis tools for Information Extraction(IE). CLOnE is a subset of natural language, which can be unambiguously parsed and converted into an ontology. While CLOnE can be unambiguously parsed in order to author an ontology. The reverse of this process NLG, verbalises a given ontology in CLOnE CNL. Combining both CLOnE and the text generator, results in a round trip ontology authoring environment whereby the user (re)produces the controlled language using the text generator, modify or edit the text as required, and subsequently parse the text back into the ontology using the CLIE environment. The process can be repeated, or round-tripped, until the desired effect is achieved. The text generator is a modern XML based shallow NLG application. It is somewhat a hybrid system, in that it has some deep NLG functionality contained in the lexicalisation process, which is informed by the SimpleNLG¹²⁹ lexicon and morphological realisation functionality.

- While one portion of the contribution of this thesis has focused on the applications of CNL to ontology authoring, the remaining portion is concerned with exploring the application of CNLs to semantic annotation. This thesis has proposed **Controlled Annotation**, which is an approach to manual semantic annotation based on CNL. A definition of Controlled Annotation was provided in Chapter 6 which outlines the intersection between traditional CNLs for ontology authoring and the process of Semantic Annotation. Controlled Annotation is a type of latent annotation, whereby the annotation step is not executed in an a-posteriori fashion, after the document authoring, but is rather merged with the authoring stage. As a means of realising our approach, we provided two implementations of controlled annotation – Controlled ANNotator(CLANN) I and II.

¹²⁹<https://code.google.com/p/simplenlg/>, Retrieved 2013-05-22

- **Evaluation of CNLs for Ontology Authoring and Semantic Annotation**

With respect to the evaluation we have provided empirical evidence that:

- ROA is more user friendly than a standard ontology editor for basic ontology editing tasks.
- ROA is more user friendly than CLOnE for basic ontology editing tasks.
- For certain scenarios, Controlled Annotation can be more user friendly than a standard manual semantic annotation approach.

- **Questions and Design issues with respect to CNLs.** Our quantitative evaluation has shown that a compact CNL can effectively substitute for a standard ontology editing tool, with respect to ontology editing tasks. However, the introduction of more complex knowledge modelling such as rules and axioms would have an impact on the design of the CNL and its subsequent usability. One could argue, as does Smart et al [Smart, shed], that user could become “lost in logic”, in that the resulting complex CNL becomes no more user friendly than the ontology editor it is trying to replace. A major question is whether a CNL is appropriate for the task? Although, in the context of ontology authoring, CNLs like CLOnE and ACE offer an attractive alternative to ontology editors, we argue that a CNL is not a panacea for resolving ambiguity when processing natural language. This is particularly true, with respect to authoring fluid natural language texts *completely* in CNL for fact creation (as in CLANN I), such as technical or clinical documentation. We argue that for these scenarios, there should be a pre-existing use case for a *human orientated* CNL, in other words a restricted vocabulary or syntax for a technical domain either legal, clinical or aeronautics such as ASD Simplified Technical English. Without such a use case, despite it being possible to adapt a human-orientated CNL to a machine processable CNL, there would be little incentive for users to interact with it.

7.2 Open Questions

- Firstly, there is the issue of **scalability** for RoundTrip Ontology Authoring. If we have larger semantic repositories such OWLIM-SE (previously BigOWLIM)¹³⁰, which can handle large ontologies and we generate CLOnE output from this knowledge, the important question is how can we present it to the user in an accessible manner? We would risk information and cognitive overload by producing large amounts of CNL text. Data Visualisation tools would be needed to effectively select portions of the ontology for generation and editing in CLOnE and more importantly to permit ease of interaction.
- Secondly, there are open research questions with respect to **ontology lexicalisation**, which is concerned with integrating ontologies and lexicons or enriching ontologies with a lexical layer [Buitelaar et al., 2009b]. A lexical layer within an ontology can be exploited in order to automatically generate language resources for NLP applications, such as Cross Lingual Ontology Based Information Extraction, Semantic Annotation, Knowledge Based Machine Translation and finally (multilingual) NLG and CNL applications for the Semantic Web. While a substantial amount of work has been done with respect to creating formal models for ontology lexicalisation, notably the *lemon* model [Buitelaar et al., 2009b], more research is needed with respect to tool development for automatically creating language resources and their integration with ontology aware NLP open source frameworks such as GATE. This is necessary for ontology lexicalisation to become a standard engineering step within NLP applications for the Semantic Web.
- Thirdly, there is the question of **integrative approaches to semantic annotation**. One could view controlled annotation naively as a solution to all annotation problems. It may prove easy to create instance data for novice users using controlled annotation or even relation metadata where the text to be annotated is completely

¹³⁰<http://www.ontotext.com/owlim>, Retrieved 2013-05-22

restricted. However, with respect to CLANN II, creating relational metadata using CNL snippets could prove cumbersome. Here, it is worth augmenting the CLANN II approach with traditional manual semantic annotation techniques (drag and drop or highlight) and more importantly semi-automatic annotation, whereby active learning could be applied to detect relation metadata across instance metadata created using controlled annotation, in particular the CLANN II approach.

- Finally, we discussed in Chapter 6, user evaluations with respect to Semantic Media Wiki. Clear empirical evidence is still needed with respect to the user friendliness of semantic wikis. Exploring the usability of controlled annotation in a semantic wiki is also an open issue. Another platform for annotation research is Google Docs¹³¹. Finally, collaborative controlled annotation is also another avenue of unexplored work.

7.3 Ongoing Work with CLANN

As mentioned in Chapter 6, based on the results of our evaluation, future work would involve merging best practices of both CLANN annotators into a hybrid annotator - CLANN III. Although the shallow parsing process used by GATE is quite robust, the parsing error messaging provided in the GATE Developer is not user friendly. Hence, a proper interface is needed to improve the usability of both CLANN annotators. In [Dantuluri et al., 2012] we describe initial efforts to build CLANN III using link grammar [Sleator and Temperley, 1991]. Link grammar can provide us with more natural language recognition power and predictive parsing/editing capabilities to improve usability. The paper [Dantuluri et al., 2012] also outlines initial steps for integrating CLANN III into a semantic wiki.

¹³¹http://en.wikipedia.org/wiki/Google_Docs, Retrieved 2013-05-22

7.4 Future Work

- **Ontology Lexicalisation and Controlled Languages:** Grammatical Framework [Angelov and Ranta, 2009, Ranta, 2004] is an implementation framework which the authors claim can cope with a variety of CNLs as well as boost the development of new ones. The core advantage of GF is its multilingualism in that its primary task is domain specific knowledge based Machine Translation (MT) based on Controlled Natural Language. A substantial amount of linguistic competence and domain expertise is needed to define a concrete syntax for a given source/target language. Consequently the authors developed a collection of GF resource libraries to provide a language engineering solution to this issue. The GF libraries now contain a collection of wide coverage grammars for over 15 natural languages. However, despite such efforts, costly grammar engineering needs to be reduced. Ontology lexicalisation can play role here whereby, emerging ontology lexicon models such as *lemon* can automatically bootstrap the creation of language resources from ontologies for exploitation by NLP systems. Some of our initial efforts with respect to mapping *lemon* lexica to GF application grammars are described in [Davis et al., 2012].
- **Multilingual Round Trip Ontology Authoring:** With respect to Roundtrip Ontology authoring, the current CLOnE language is targeted to English, however as GATE permits the creation of multilingual information extraction pipelines. So language specific resources can be replaced relatively easily [Bontcheva et al., 2003]. Hence, re-targeting the CLOnE language for other European languages, i.e. German or French, is achievable. This would involve replacing the part-of-speech tagger and localising the gazetteer lists. With respect to the text generator component in ROA, as a first step, lexicalising the Ontology using the *lemon* source tools¹³² would ease the development of re-targeting the generator to other European languages. We completed some initial work with respect to this task in [Davis et al., 2011],

¹³²<http://monnetproject.deri.ie/lemonsource/>, Retrieved 2013-05-22

whereby the *lemon* source API was manipulated to automatically generate ontology aware GATE gazetteers. The code was wrapped as a custom GATE plugin - `lemonGazeteerGenerator`, allowing us to load a *lemon* ontology-lexicon for a given ontology into GATE in order to generate hierarchical gazetteer entries.

- **Generating Tailored Reports:** With respect to our work with NLG, we have focused only on ontology verbalisation i.e. verbalising the outcomes of the *knowledge meta process* (via RoundTrip Ontology Authoring) as opposed to generating a summary of the facts created in the knowledge base (via Controlled Annotation), in other words the *knowledge process* [Sure, 2003]. NLG can be applied to create tailored reports from a knowledge base. So in the context of the meeting minutes use case in Chapter 6. This could involve a tailored report for a period of a month based on the tasks completed by a particular user i.e. `All engineering task status completed by Brian Davis for June 2012 with respect to the Nepomuk project`. NLG has the potential to keep documentation of formal knowledge up-to-date in automatic manner. The challenge posed by the Semantic Web for NLG systems is to provide tools and techniques that are extensible and maintainable, especially as most deep NLG systems are developed and extended by specialists [Bontcheva and Davis, 2008]. NLG tools can present reports that are context sensitive and user profile centric. In addition, reports generated by NLG can summarize the knowledge in natural language, whereby the appropriate level granularity can be presented to the user depending on their role and expertise. Within NLG research itself, interesting avenues of research that hold promise and worth exploration include, modern XML generation system [Wilcock, 2000], hybrid NLG systems [Klarner, 2004b, Klarner, 2004a], as well as the increasing application of machine learning approaches for adaptive interaction with users [Bontcheva and Cunningham, 2003]. Finally, reference architectures for NLG such as RAGS are worth revisiting, especially with respect to the role semantic web languages can play across data interchange between components within NLG infrastructures [Mellish, 2010].

- **Round Trip Annotation:** While respect to CLANN II, we explored the notion of annotating free text with embedded snippets of Controlled Language. In [Dantuluri et al., 2012], we explore some of the steps required to embed controlled annotation into a semantic wiki. We would envisage using predictive parsing and auto completion to aid in the creation of controlled annotations while authoring an article. However, if an annotated article must be edited, simple text generation could play a role here in automatically generating out the original CNL for manipulation and re-annotation. Consequently, this feed back loop would result in a type of Round Trip Annotation. Finally, there is the issue of evaluating a semantic wiki with controlled annotation features. While a task based user evaluation comparing such a wiki with a baseline semantic wiki is appropriate, we believe that controlled annotation would not replace classical a posteriori annotation but could in fact be complimentary. Annotating relational metadata is particularly cumbersome, and likewise our CLANN II syntax does not adequately deal with complex relations as metadata. We would envisage a multi-model approach to semantic annotation, whereby mixed manual highlight and click a posteriori annotation and controlled annotation could be used to effectively create annotations at the instance level, leaving the detection of relations to machine learning driven semi-supervised semantic annotation. Correct automatic annotation at the relational level would of course be highly dependent on the correct identification of instances of both subjects and objects for a given relation. Controlled Annotation could play a role here in enforcing correct annotations at the instance level.

7.5 Summary

Both (i) ontology authoring and (ii) semantic annotation are core challenges for building the Semantic Web. With respect to (i), this thesis described and evaluated a user friendly way for non-experts to author ontologies using a combination of controlled language technology and shallow natural language generation techniques. The evaluation results

of our approach have proven favourable in comparison to a standard ontology editing tool in the context of simple ontology editing tasks. In regard to (ii), we have explored the application of controlled language technology to the process of semantic annotation, in particular drawing attention to design issues, the trade off between, usability, knowledge capture and complexity of language modelling. Both tools for controlled annotation were evaluated and compared to a standard manual annotation tool. The outcome of the evaluation was also favourable with respect to both implementations.

The work in this thesis focused primarily on language design issues with respect to the application of CNLs to ontology authoring and semantic annotation. Future research challenges, regarding our CNL work, will require richer tool interactivity. This will result in more complex human computer interaction challenges when augmenting and testing our tools for more scalable collaborative knowledge creation, regardless of whether the task is ontology authoring or semantic annotation.

Bibliography

- [Kou, 2002] (2002). In Gómez-Pérez, A., editor, *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, volume 2473 of *Lecture Notes in Computer Science*.
- [Adida et al., 2008] Adida, B., Birbeck, M., McCarron, S., and Pemberton, S. (2008). RDFa in XHTML: Syntax and Processing — A collection of attributes and processing rules for extending XHTML to support RDF. W3C Recommendation, W3C. <http://www.w3.org/TR/rdfa-syntax/>.
- [Adriaens and Schreors, 1992] Adriaens, G. and Schreors, D. (1992). From cogram to alcogram: toward a controlled english grammar checker. In *Proceedings of the 14th conference on Computational linguistics*, pages 595–601, Morristown, NJ, USA. Association for Computational Linguistics.
- [Adrian et al., 2008] Adrian, B., Neumann, G., Trousov, A., and Popov, B. (2008). Ontology-based information extraction systems (obies 2008).
- [Aguado et al., 1998] Aguado, G., Bañón, A., Bateman, J. A., Bernardos, S., Fernández, M., Gómez-Pérez, A., Nieto, E., Olalla, A., Plaza, R., and Sánchez, A. (1998). Ontogeneration: Reusing domain and linguistic ontologies for spanish text generation. In *Proceedings of the ECAI'98 Workshop on Applications of Ontologies and Problem Solving Methods*, Brighton, U.K. European Conference on Artificial Intelligence.
- [Aho and Corasick, 1975] Aho, A. V. and Corasick, M. J. (1975). Efficient string matching: an aid to bibliographic search. *Commun. ACM*, 18(6):333–340.

- [Andrews et al., 2012] Andrews, P., Zaihrayeu, I., and Pane, J. (2012). A classification of semantic annotation systems. *Semantic Web*, 3(3):223–248.
- [Angelov and Enache, 2010] Angelov, K. and Enache, R. (2010). Typeful ontologies with direct multilingual verbalization. In Rosner, M. and Fuchs, N. E., editors, *CNL*, volume 7175 of *Lecture Notes in Computer Science*, pages 1–20. Springer.
- [Angelov and Ranta, 2009] Angelov, K. and Ranta, A. (2009). Implementing controlled languages in gf. In *CNL*, pages 82–101.
- [Antoniou et al., 2005] Antoniou, G., Franconi, E., and Harmelen, F. V. (2005). Introduction to semantic web ontology languages. In *Reasoning Web, Proceedings of the Summer School, Malta, 2005. Number 3564 in Lecture Notes in Computer Science*. Springer.
- [Appelt and Israel, 1999] Appelt, D. E. and Israel, D. J. (1999). Introduction to information extraction technology: A tutorial prepared for IJCAI99.
- [Appelt and Onyshkevych, 1998] Appelt, D. E. and Onyshkevych, B. (1998). The common pattern specification language. In *Proceedings of a workshop on held at Baltimore, Maryland: October 13-15, 1998, TIPSTER '98*, pages 23–30, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Aristotle, 350] Aristotle (ca. 350). Prior analytics.
- [Auer and Lehmann, 2007] Auer, S. and Lehmann, J. (2007). What have innsbruck and leipzig in common? extracting semantics from wiki content. In *ESWC*, pages 503–517.
- [Bailey, 2006] Bailey, B. (2006). Getting the complete picture with usability testing. Usability updates newsletter, U.S. Department of Health and Human Services.
- [Bao et al., 2009] Bao, J., Smart, P., Braines, D., and Shadbolt, N. (2009). A controlled natural language interface for semantic media wiki using the rabbit language. In *Workshop on Controlled Natural Language (CNL'09)*.

- [Baumgartner et al., 2001] Baumgartner, R., Flesca, S., and Gottlob, G. (2001). Visual web information extraction with lixto. In *Proceedings of the 27th International Conference on Very Large Data Bases, VLDB '01*, pages 119–128, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- [Bechhofer et al., 2004] Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. (2004). Owl web ontology language reference. *W3C Recommendation*, 10.
- [Beckett, 2004] Beckett, D. (2004). RDF/XML Syntax Specification (Revised). W3C Recommendation, W3C. <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>.
- [Beckett, 2007] Beckett, D. (2007). Turtle — Terse RDF Triple Language. <http://www.dajobe.org/2004/01/turtle/>.
- [Berners-Lee, 1993] Berners-Lee, T. (1993). *A Brief History of the Web*. W3C Design Issues.
- [Berners-Lee, 2006a] Berners-Lee, T. (2006a). Linked data. <http://www.w3.org/DesignIssues/LinkedData.html>.
- [Berners-Lee, 2006b] Berners-Lee, T. (2006b). Notation 3. <http://www.w3.org/DesignIssues/Notation3>.
- [Berners-Lee et al., 2001a] Berners-Lee, T., Hendler, J., and Lassila, O. (2001a). The semantic web. *Scientific American*, 284(5):34–43.
- [Berners-Lee et al., 2001b] Berners-Lee, T., Hendler, J., and Lassila, O. (2001b). The Semantic Web. *Scientific American*, 284(5).
- [Bernstein and Kaufmann, 2006] Bernstein, A. and Kaufmann, E. (2006). GINO—a guided input natural language ontology editor. In *5th International Semantic Web Conference (ISWC2006)*.

- [Bernstein et al., 2004] Bernstein, A., Kaufmann, E., Fuchs, N., and von Bonin, J. (2004). Talking to the semantic web: a controlled english query interface for ontologies. In *14th Workshop on Information Technology and Systems*, pages 212–217.
- [Bishop et al., 2011] Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., and Velkov, R. (2011). Owlrim: A family of scalable semantic repositories. *Semantic Web*, 2(1):33–42.
- [Bizer et al., 2007] Bizer, C., Cyganiak, R., and Heath, T. (2007). How to Publish Linked Data on the Web. <http://sites.wiwiw.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/>.
- [Bizer et al., 2009] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22.
- [Bloehdorn et al., 2005] Bloehdorn, S., Petridis, K., Saathoff, C., Simou, N., Tzouvaras, V., Avrithis, Y., Handschuh, S., Kompatsiaris, Y., Staab, S., and Strintzis, M. G. (2005). Semantic annotation of images and videos for multimedia analysis. In Gómez-Pérez, A. and Euzenat, J., editors, *The Semantic Web: Research and Applications: Proceedings of the Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29-June 1, 2005*, volume 3532 of *Lecture Notes in Computer Science*, pages 592–607. Springer.
- [Bontcheva, 2003] Bontcheva, K. (2003). Reuse and challenges in evaluating language generation systems: position paper. In *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable?*, Evalinitatives '03, pages 3–9, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Bontcheva, 2005] Bontcheva, K. (2005). Generating tailored textual summaries from ontologies. *The Semantic Web: Research and Applications*, pages 531–545.

- [Bontcheva and Cunningham, 2003] Bontcheva, K. and Cunningham, H. (2003). The semantic web: A new opportunity and challenge for human language technology. Second Workshop on Human Language Technology for the Semantic Web and Web Services.
- [Bontcheva and Davis, 2008] Bontcheva, K. and Davis, B. (2008). *Semantic Knowledge Management: Integrating Ontology Management, Knowledge Discovery and Human Language Technologies*, chapter Natural Language Generation from Ontologies, pages 20–30. In [Davies et al., 2008], 1st edition.
- [Bontcheva et al., 2008] Bontcheva, K., Davis, B., and andYaoyong Li, A. F. (2008). *Semantic Knowledge Management: Integrating Ontology Management, Knowledge Discovery and Human Language Technologies*, chapter Human Language Technologies, pages 20–30. In [Davies et al., 2008], 1st edition.
- [Bontcheva et al., 2003] Bontcheva, K., Maynard, D., Tablan, V., and Cunningham, H. (2003). Gate: A unicode-based infrastructure supporting multilingual information extraction. In *In Proceedings of Workshop on Information Extraction for Slavonic and other Central and Eastern European Languages (IESL03), Borovets*.
- [Bontcheva et al., 2004] Bontcheva, K., Tablan, V., Maynard, D., and Cunningham, H. (2004). Evolving gate to meet new challenges in language engineering. *Natural Language Engineering*, 10(3-4):349–373.
- [Bontcheva and Wilks, 2004] Bontcheva, K. and Wilks, Y. (2004). Automatic report generation from ontologies: the miakt approach. *Natural Language Processing and Information Systems*, pages 1–19.
- [Borin et al., 2008] Borin, L., Forsberg, M., and Lönngrén, L. (2008). Saldo 1.0 (svenskt associationslexikon version 2). *Språkbanken, University of Gothenburg*.
- [Breslin et al., 2005] Breslin, J. G., Harth, A., Bojars, U., and Decker, S. (2005). Towards Semantically-Interlinked Online Communities. In *Proceedings of the 2nd European Semantic Web Conference (ESWC2005)*.

- [Brickley and Guha, 2004] Brickley, D. and Guha, R. (2004). RDF vocabulary description language 1.0: RDF schema. W3C Recommendation, W3C. <http://www.w3.org/TR/rdf-schema/>.
- [Brickley and Miller, 2005] Brickley, D. and Miller, L. (2005). FOAF vocabulary specification. <http://xmlns.com/foaf/0.1/>.
- [Broekstra and Kampman, 2003] Broekstra, J. and Kampman, A. (2003). The SeRQL Query Language. Technical report, Aduna.
- [Brooke, 1996] Brooke, J. (1996). SUS: a “quick and dirty” usability scale. In Jordan, P., Thomas, B., Weerdmeester, B., and McClelland, A., editors, *Usability Evaluation in Industry*. Taylor and Francis, London.
- [Buitelaar et al., 2009a] Buitelaar, P., Cimiano, P., Haase, P., and Sintek, M. (2009a). Towards linguistically grounded ontologies. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 111–125, Berlin, Heidelberg. Springer-Verlag.
- [Buitelaar et al., 2009b] Buitelaar, P., Cimiano, P., Haase, P., and Sintek, M. (2009b). Towards linguistically grounded ontologies. In Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., and Simperl, E. P. B., editors, *ESWC*, volume 5554 of *Lecture Notes in Computer Science*, pages 111–125. Springer.
- [Buitelaar and Ramaka, 2005] Buitelaar, P. and Ramaka, S. (2005). Unsupervised ontology-based semantic tagging for knowledge markup. *Learning in Web Search (LWS 2005)*, page 26.
- [Buraga et al., 2006] Buraga, S. C., Cojocar, L., and Nichifor, O. C. (2006). Survey on web ontology editing tools. *Transactions on Automatic Control and Computer Science, Romania*, pages 1–6.

- [Bush and Wang, 1945] Bush, V. and Wang, J. (1945). As we may think. *Atlantic Monthly*, 176:101–108.
- [Calder, 1996] Calder, J. (1996). Statistical techniques. In Sapsford, R. and Jupp, V., editors, *Data Collection and Analysis*, chapter 9. Open University.
- [Cardoso and Escórcio, 2007] Cardoso, J. and Escórcio, A. L. N. (2007). Editing tools for ontology construction. *Idea, March*, pages 1–27.
- [Carr et al., 2004] Carr, L., Miles-Board, T., Wills, G., Woukeu, A., and Hall, W. (2004). Towards a knowledge-aware office environment. *Practical Aspects of Knowledge Management*, pages 129–140.
- [Caterpillar Corporation, 1974] Caterpillar Corporation (1974). *Dictionary for Caterpillar Fundamental English*. Caterpillar Corporation.
- [Chomsky, 1956] Chomsky, N. (1956). Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113–124.
- [Chomsky, 1965] Chomsky, N. (1965). *Aspects of the theory of syntax*. MIT Press, Cambridge, Massachusetts.
- [Cimiano et al., 2004] Cimiano, P., Handschuh, S., and Staab, S. (2004). Towards the self-annotating web. In *Proceedings of the 13th international conference on World Wide Web, WWW '04*, pages 462–471, New York, NY, USA. ACM.
- [Ciravegna et al., 2002] Ciravegna, F., Dingli, A., Petrelli, D., and Wilks, Y. (2002). User-system cooperation in document annotation based on information extraction. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 122–137.
- [Ciravegna and Wilks, 2003] Ciravegna, F. and Wilks, Y. (2003). Designing adaptive information extraction for the semantic web in amilcare. *Annotation for the Semantic*

Web, in the Series *Frontiers in Artificial Intelligence and Applications*, IOS Press, Amsterdam.

- [Clark et al., 2005] Clark, P., Harrison, P., Jenkins, T., Thompson, J. A., and Wojcik, R. H. (2005). Acquiring and using world knowledge using a restricted subset of english. In Russell, I. and Markov, Z., editors, *FLAIRS Conference*, pages 506–511. AAAI Press.
- [Connolly and Sluckin, 1971] Connolly, T. G. and Sluckin, W. (1971). *An Introduction to Statistics for the Social Sciences*. Macmillan, third edition.
- [Consortium, 2008] Consortium, L. D. (2008). Ace (automatic content extraction) english annotation guidelines for relations (version 6.2).
- [Corcho et al., 2003] Corcho, O., Fernández-López, M., and Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies: where is their meeting point? *Data Knowl. Eng.*, 46(1):41–64.
- [Cunningham, 1999] Cunningham, H. (1999). Information extraction - a user guide. Technical Report CS-99-07, University of Sheffield, UK.
- [Cunningham, 2002] Cunningham, H. (2002). GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254.
- [Cunningham, 2005] Cunningham, H. (2005). Information extraction, automatic. *Encyclopedia of Language and Linguistics*,.
- [Cunningham et al., 2002] Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2002). 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)*.
- [Cunningham et al., 2011] Cunningham, H., Tablan, V., Roberts, I., Greenwood, M., and Aswani, N. (2011). Information extraction and semantic annotation for multi-paradigm

- information management. *Current Challenges in Patent Information Retrieval*, pages 307–327.
- [Dannélls, 2008] Dannélls, D. (2008). Generating tailored texts for museum exhibits. In *Proceedings of the 6th edition of LREC 2008, Workshop on Language Technology for Cultural Heritage Data (LaTeCH), Marrakech, Morocco.*, pages 17–20.
- [Dannélls et al., 2012] Dannélls, D., Damova, M., Enache, R., and Chechev, M. (2012). Multilingual online generation from semantic web ontologies. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 239–242. ACM.
- [Dantuluri et al., 2012] Dantuluri, P., Davis, B., Ludwick, P., and Handschuh, S. (2012). Engineering a controlled natural language into semantic mediawiki. *Controlled Natural Language*, pages 53–72.
- [Davies et al., 2008] Davies, J., Grobelnik, M., and Mladenic, D., editors (2008). *Semantic Knowledge Management: Integrating Ontology Management, Knowledge Discovery and Human Language Technologies*. Springer Publishing Company, Incorporated, 1st edition.
- [Davis et al., 2011] Davis, B., Badra, F., Buitelaar, P., Wunner, T., and Handschuh, S. (2011). Squeezing lemon with gate. In *Proceedings of the 2nd International Workshop on the Multilingual Semantic Web (MSW 2011), Workshop at ISWC 2011*.
- [Davis et al., 2010] Davis, B., Dantuluri, P., Handschuh, S., and Cunningham, H. (2010). Towards controlled natural language for semantic annotation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 6(4):64–91.
- [Davis et al., 2012] Davis, B., Enache, R., van Grondelle, J., and Pretorius, L. (2012). Multilingual verbalisation of modular ontologies using gf and lemon. In *CNL 2012*, volume LNCS. Springer, Springer.

- [Davis et al., 2008] Davis, B., Iqbal, A. A., Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., and Handschuh, S. (2008). Roundtrip ontology authoring. In Sheth, A. P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T. W., and Thirunarayan, K., editors, *International Semantic Web Conference*, volume 5318 of *Lecture Notes in Computer Science*, pages 50–65. Springer.
- [Decker, 2002] Decker, S. (2002). *Semantic web methods for knowledge management*. PhD thesis, Universität Karlsruhe.
- [Dekel et al., 2004] Dekel, O., Keshet, J., and Singer, Y. (2004). Large margin hierarchical classification. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 27–, New York, NY, USA. ACM.
- [Dill et al., 2003] Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J., et al. (2003). Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th international conference on World Wide Web*, pages 178–186. ACM.
- [Dimitrova et al., 2008] Dimitrova, V., Denaux, R., Hart, G., Dolbear, C., Holt, I., and Cohn, A. (2008). Involving Domain Experts in Authoring OWL Ontologies. In *Proceedings of the 7th International Semantic Web Conference (ISWC 2008), Karlsruhe, Germany*. Springer.
- [Dingli et al., 2003] Dingli, A., Ciravegna, F., and Wilks, Y. (2003). Automatic semantic annotation using unsupervised information extraction and integration. In *Proceedings of SemAnnot 2003 Workshop*.
- [Dymetman et al., 2000] Dymetman, M., Lux, V., and Ranta, A. (2000). Xml and multilingual document authoring: convergent trends. In *Proceedings of the 18th conference on Computational linguistics - Volume 1*, pages 243–249, Morristown, NJ, USA. Association for Computational Linguistics.

- [Engelbart, 1962] Engelbart, D. C. (1962). Augmenting Human Intellect: A Conceptual Framework. Summary Report AFOSR-3223 - Contract AF49(638)-1024, SRI Project 3578, Air Force Office of Scientific Research, Stanford Research Institute, Menlo Park, CA.
- [Engelbart and English, 1968] Engelbart, D. C. and English, W. K. (1968). A research center for augmenting human intellect. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, AFIPS '68 (Fall, part I), pages 395–410, New York, NY, USA. ACM.
- [Engelbrecht et al., 2009] Engelbrecht, P. C., Hart, G., and Dolbear, C. (2009). Talking rabbit: A user evaluation of sentence production. In *CNL*, pages 56–64.
- [España-Bonet et al., 2011] España-Bonet, C., Enach, R., Slaski, A., Ranta, A., Marquez, L., and Gonzalez, M. (2011). Patent translation within the molto project. In *Workshop on Patent Translation, MT Summit XIII*, pages 70–78.
- [Etzioni et al., 2005] Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D., and Yates, A. (2005). Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.
- [Fellbaum, 1998] Fellbaum, C., editor (1998). *WordNet - An Electronic Lexical Database*. MIT Press.
- [Fernandez-Lopez et al., 1997] Fernandez-Lopez, M., Gomez-Perez, A., and Juristo, N. (1997). Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium*, pages 33–40, Stanford, USA.
- [Foster and White, 2004] Foster, M. E. and White, M. (2004). Techniques for text planning with xslt. In *Proceedings of the Workshop on NLP and XML (NLPXML-2004): RDF/RDFS and OWL in Language Technology*, NLPXML '04, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

- [Francopoulo et al., 2006] Francopoulo, G., George, M., Calzolari, N., Monachini, M., Bel, N., Pet, M., Soria, C., et al. (2006). Lmf for multilingual, specialized lexicons. In *International Conference on Language Resources and Evaluation-LREC 2006*.
- [Frege, 1879] Frege, G. (1879). Begriffsschrift, eine der arithmetischen nachgebildete formelsprache des reinen denkens.
- [Fuchs and Schwertel, 2003] Fuchs, N. and Schwertel, U. (2003). Reasoning in attempto controlled english. *Principles and Practice of Semantic Web Reasoning*, pages 174–188.
- [Fuchs and Schwitter, 1996a] Fuchs, N. and Schwitter, R. (1996a). Attempto controlled english (ace). See citeseer.ist.psu.edu/article/fuchs96attempto.html.
- [Fuchs and Schwitter, 1996b] Fuchs, N. and Schwitter, R. (1996b). Attempto Controlled English (ACE). In *CLAW96: Proceedings of the First International Workshop on Controlled Language Applications*, Leuven, Belgium.
- [Fuchs and Schwitter, 2007] Fuchs, N. and Schwitter, R. (2007). Web-annotations for humans and machines. *The Semantic Web: Research and Applications*, pages 458–472.
- [Fuchs et al., 2006] Fuchs, N. E., Kaljurand, K., Kuhn, T., Schneider, G., Royer, L., and Schröder, M. (2006). Attempto Controlled English and the semantic web. Deliverable I2D7, REVERSE Project.
- [Funk et al., 2007] Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., and Handschuh, S. (2007). Clone: Controlled language for ontology editing. In *ISWC/ASWC*, pages 142–155.
- [Gatt and Reiter, 2009] Gatt, A. and Reiter, E. (2009). Simplenlg: a realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation, ENLG '09*, pages 90–93, Stroudsburg, PA, USA. Association for Computational Linguistics.

- [Gazdar and Mellish, 1989] Gazdar, G. and Mellish, C. (1989). *Natural Language Processing in Prolog*. Addison-Wesley, Reading, MA.
- [Genesereth et al., 1992] Genesereth, M., Fikes, R., et al. (1992). Knowledge interchange format-version 3.0: reference manual.
- [Gilardoni et al., 2005] Gilardoni, L., Biasuzzi, C., Ferraro, M., Fonti, R., and Slavazza, P. (2005). Machine learning for the semantic web: Putting the user into the cycle. In *Dagstuhl Seminar*.
- [Goble et al., 2003] Goble, S. B. C., Carr, L., Hall, W., Kampa, S., and De Roure, D. (2003). Cohse: Conceptual open hypermedia service. *Annotation for the semantic Web*, 96:193.
- [Goldberg et al., 1994] Goldberg, E., Driedger, N., and Kittredge, R. (1994). Using natural language to produce weather forecasts. *IEEE Expert*, April '94, pages 45–53.
- [Goldfarb, 1996] Goldfarb, C. F. (1996). *The Roots of SGML - A Personal Recollection*. Retrieved 2007-07-07.
- [Gómez-Pérez et al., 2007] Gómez-Pérez, A., Fernández-López, M., and Corcho, O. (2007). *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. (Advanced Information and Knowledge Processing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- [Gracia et al.,] Gracia, J., Montiel-Ponsoda, E., Cimiano, P., and Asunción. Challenges for the multilingual web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 11(0):63 – 71.
- [Grant and Beckett, 2004] Grant, J. and Beckett, D. (2004). RDF test cases. W3C Recommendation, W3C. <http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/>.
- [Grover et al., 2000] Grover, C., Holt, A., Holt, E., Klein, E., and Moens, M. (2000). Designing a controlled language for interactive model checking.

- [Groza, 2012] Groza, T. (2012). *Advances in semantic authoring and publishing*, volume 13. Akademische Verlagsgesellschaft AKA; IOS Press.
- [Groza et al., 2011] Groza, T., Handschuh, S., and Decker, S. (2011). Capturing rhetoric and argumentation aspects within scientific publications. *Journal on data semantics XV*, pages 1–36.
- [Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.
- [Guarino, 1998] Guarino, N. (1998). *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition.
- [Haase et al., 2008] Haase, P., Lewen, H., Studer, R., Tran, D. T., Erdmann, M., dAquin, M., and Motta, E. (2008). The neon ontology engineering toolkit. *WWW*.
- [Hallett et al., 2006] Hallett, C., Power, R., and Scott, D. (2006). Summarisation and visualisation of e-health data repositories.
- [Hallett et al., 2007] Hallett, C., Scott, D., and Power, R. (2007). Composing questions through conceptual authoring. *Comput. Linguist.*, 33(1):105–133.
- [Handschuh, 2005] Handschuh, S. (2005). *Creating Ontology-based Metadata by Annotation for the Semantic Web*. PhD thesis.
- [Handschuh and Staab, 2002] Handschuh, S. and Staab, S. (2002). Authoring and annotation of web pages in cream. In *WWW*, pages 462–473.
- [Handschuh et al., 2002] Handschuh, S., Staab, S., and Ciravegna, F. (2002). S-cream - semi-automatic creation of metadata. In *Proc. of the European Conference on Knowledge Acquisition and Management - EKAW-2002. Madrid, Spain, October 1-4, 2002*, LNCS. Springer.

- [Handschuh et al., 2003a] Handschuh, S., Staab, S., and Studer, R. (2003a). Leveraging metadata creation for the semantic web with cream. In Günter, A., Kruse, R., and Neumann, B., editors, *KI*, volume 2821 of *Lecture Notes in Computer Science*, pages 19–33. Springer.
- [Handschuh et al., 2003b] Handschuh, S., Volz, R., and Staab, S. (2003b). Annotation for the deep web. *IEEE Intelligent Systems*, 18(5):42–48.
- [Hart et al., 2007] Hart, G., Dolbear, C., and Goodwin, J. (2007). Lege feliciter: Using structured english to represent a topographic hydrology ontology. In *OWLED*.
- [Hart et al., 2008] Hart, G., Johnson, M., and Dolbear, C. (2008). Rabbit: Developing a control natural language for authoring ontologies. In *5th European Semantic Web Conference (ESWC2008)*, pages 348–360.
- [Hayes-Roth et al., 1983] Hayes-Roth, F., Waterman, D. A., and Lenat, D. B. (1983). An overview of expert systems. In Hayes-Roth, F., Waterman, D. A., and Lenat, D. B., editors, *Building Expert Systems*, pages 3–29. Addison-Wesley, London.
- [Hearst, 1992] Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics - Volume 2, COLING '92*, pages 539–545, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Heflin and Hendler, 2001] Heflin, J. and Hendler, J. (2001). A portrait of the semantic web in action. *IEEE Intelligent Systems*, 16(2):54–59.
- [Hielkema et al., 2007a] Hielkema, F., Edwards, P., Mellish, C., and Farrington, J. (2007a). A flexible interface to community-driven metadata, to appear. In *in Proceedings of the e Social Science conference 2007, Ann Arbor, Michigan*.
- [Hielkema et al., 2007b] Hielkema, F., Mellish, C., and Edwards, P. (2007b). Using wysi-wym to create an open-ended interface for the semantic grid. In *Proceedings of the*

- Eleventh European Workshop on Natural Language Generation, ENLG '07*, pages 69–72, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Hielkema et al., 2008] Hielkema, F., Mellish, C., and Edwards, P. (2008). Evaluating an ontology-driven wysiwyw interface. In White, M., Nakatsu, C., and McDonald, D., editors, *INLG*. The Association for Computer Linguistics.
- [Hildebrand et al., 1977] Hildebrand, D. K., Laing, J. D., and Rosenthal, H. (1977). *Analysis of Ordinal Data*. Quantitative Applications in the Social Sciences. Sage.
- [Hobbs et al., 1996] Hobbs, J. R., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M., and Tyson, M. (1996). Fastus: A cascaded finite-state transducer for extracting information from natural-language text. In Roche, E. and Schabes, Y., editors, *Finite State Devices for Natural Language Processing*. MIT Press.
- [Hoeffler, 2004] Hoeffler, S. (2004). The syntax of Attempto Controlled English: An abstract grammar for ACE 4.0. Technical Report ifi-2004.03, Department of Informatics, University of Zurich.
- [Hunter and Schroeter, 2008] Hunter, J. and Schroeter, R. (2008). Co-annotate: A system for tagging relationships between multiple mixed-media objects. *Multimedia, IEEE*, 15(3):42–53.
- [Hwang, 1999] Hwang, C. H. (1999). Incompletely and imprecisely speaking: Using dynamic ontologies for representing and retrieving information. In Franconi, E. and Kifer, M., editors, *KRDB*, volume 21 of *CEUR Workshop Proceedings*, pages 14–20. CEUR-WS.org.
- [Jacobs et al., 1991] Jacobs, P., Krupka, G., and Rau, L. (1991). Lexico-semantic pattern matching as a companion to parsing in text understanding. In *Fourth DARPA Speech and Natural Language Workshop*, volume 337, page 342.

- [John L. Phillips, 1996] John L. Phillips, J. (1996). *How to Think about Statistics*. W. H. Freeman and Company, New York.
- [Joshi, 1987] Joshi, A. K. (1987). The relevance of Tree Adjoining Grammar to generation. In Kempen, G., editor, *Natural Language Generation*. Martinus Nijhoff Publishers, Dordrecht.
- [Kahan et al., 2001] Kahan, J., Koivunen, M.-R., Prud’Hommeaux, E., and Swick, R. R. (2001). Annotea: An open rdf infrastructure for shared web annotations. In *Proceedings of the Tenth International World Wide Web Conference*, pages 623–632, Hong Kong. ACM Press.
- [Kaljurand, 2006] Kaljurand, K. (2006). Writing owl ontologies in ace. Technical report, University of Zurich.
- [Kaljurand, 2008] Kaljurand, K. (2008). ACE View — an ontology and rule editor based on Attempto Controlled English. In *5th OWL Experiences and Directions Workshop (OWLED 2008)*, Karlsruhe, Germany. 12 pages.
- [Kaljurand and Fuchs, 2007] Kaljurand, K. and Fuchs, N. (2007). Verbalizing OWL in Attempto Controlled English. In *Proceedings of OWL: Experiences and Directions (OWLED 2007)*.
- [Kaljurand and Fuchs, 2006] Kaljurand, K. and Fuchs, N. E. (2006). Bidirectional mapping between OWL DL and Attempto Controlled English. In *Fourth Workshop on Principles and Practice of Semantic Web Reasoning*, Budva, Montenegro.
- [Kalyanpur et al., 2006] Kalyanpur, A., Parsia, B., Sirin, E., Grau, B., and Hendler, J. (2006). Swoop: A web ontology editing browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):144–153.

- [Karvounarakis et al., 2002] Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., and Scholl, M. (2002). RQL: A Declarative Query Language for RDF. In *Proceedings of the International World-Wide Web Conference (WWW2002)*.
- [Kashyap et al., 2008a] Kashyap, V., Bussler, C., and Moran, M. (2008a). *The Semantic Web: semantics for data and services on the Web*. Data-centric systems and applications. Springer.
- [Kashyap et al., 2008b] Kashyap, V., Bussler, C., and Moran, M. (2008b). *The Semantic Web: Semantics for Data and Services on the Web*. Springer, Berlin.
- [Kaufmann, 2007] Kaufmann, E. (2007). *Talking to the semantic web : natural language query interfaces for casual end-users*. PhD thesis, Universität Zürich.
- [Klarner, 2004a] Klarner, M. (2004a). Hybrid nlg in a generic dialog system. *Natural Language Generation*, pages 205–211.
- [Klarner, 2004b] Klarner, M. (2004b). Hyperbug: A scalable natural language generation approach. In Porzel, R., editor, *HLT-NAACL 2004 Workshop: 2nd Workshop on Scalable Natural Language Understanding*, pages 65–71, Boston, Massachusetts, USA. Association for Computational Linguistics.
- [Klyne and Carroll, 2004] Klyne, G. and Carroll, J. J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, W3C. <http://www.w3.org/TR/rdf-concepts/>.
- [Knublauch et al., 2004] Knublauch, H., Fergerson, R. W., Noy, N. F., and Musen, M. A. (2004). The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In McIlraith, S. ., Plexousakis, D., and van, Harmelen, r. a. n. k., editors, *The Semantic Web ISWC 2004*, volume 3298 of *Lecture Notes in Computer Science*, chapter 17, pages 229–243. Springer Berlin / Heidelberg, Berlin, Heidelberg.

- [Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, ACL '07*, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Kogut et al., 2001] Kogut, P., Holmes, W., et al. (2001). Aerodaml: Applying information extraction to generate daml annotations from web pages. In *First International Conference on Knowledge Capture (K-CAP 2001). Workshop on Knowledge Markup and Semantic Annotation, Victoria, BC, Canada*.
- [Koivunen, 2005] Koivunen, M. (2005). Annotea and semantic web supported collaboration. In *Invited talk at Workshop on User Aspects of the Semantic Web (UserSWeb), at European Semantic Web Conference (ESWC 2005) Heraklion, Greece*, volume 29.
- [Krötzsch et al., 2006] Krötzsch, M., Vrandečić, D., and Völkel, M. (2006). Semantic MediaWiki. In *The Semantic Web - ISWC 2006*, volume 4273, chapter 68, pages 935–942. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Krötzsch et al., 2007] Krötzsch, M., Vrandečić, D., Völkel, M., Haller, H., and Studer, R. (2007). Semantic Wikipedia. *Journal of Web Semantics*, 5(4):251–261.
- [Kuhn, 2006] Kuhn, T. (2006). Attempto Controlled English as ontology language. In Bry, F. and Schwertel, U., editors, *REWERSE Annual Meeting 2006*.
- [Kuhn, 2008] Kuhn, T. (2008). AceWiki: Collaborative Ontology Management in Controlled Natural Language. In *Proceedings of the 3rd Semantic Wiki Workshop*. CEUR Workshop Proceedings.
- [Kuhn, 2010a] Kuhn, T. (2010a). *Controlled English for Knowledge Representation(to Appear)*. PhD thesis, University of Zurich.

- [Kuhn, 2010b] Kuhn, T. (2010b). An evaluation framework for controlled natural languages. In Fuchs, N. E., editor, *Proceedings of the Workshop on Controlled Natural Language (CNL 2009)*, volume 5972 of *Lecture Notes in Computer Science*, pages 1–20, Berlin / Heidelberg, Germany. Springer.
- [Kuhn et al., 2006] Kuhn, T., Royer, L., Fuchs, N., and Schroeder, M. (2006). Improving text mining with controlled natural language: A case study for protein interactions. In *Data Integration in the Life Sciences*, pages 66–81. Springer.
- [Lassila and Swick, 1999] Lassila, O. and Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, W3C. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>.
- [Li and Bontcheva, 2007] Li, Y. and Bontcheva, K. (2007). Hierarchical, perceptron-like learning for ontology-based information extraction. In *Proceedings of the 16th international conference on World Wide Web, WWW '07*, pages 777–786, New York, NY, USA. ACM.
- [Li et al., 2005] Li, Y., Bontcheva, K., and Cunningham, H. (2005). Using uneven margins svm and perceptron for information extraction. In *Proceedings of the Ninth Conference on Computational Natural Language Learning, CONLL '05*, pages 72–79, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Lopez and Motta, 2004] Lopez, V. and Motta, E. (2004). Ontology driven question answering in AquaLog. In *NLDB 2004 (9th International Conference on Applications of Natural Language to Information Systems)*, Manchester.
- [Lopez et al., 2006] Lopez, V., Motta, E., and Uren, V. (2006). Poweraqua: Fishing the semantic web. In *ESWC*, pages 393–410.
- [Lorch et al., 2003] Lorch, M., Proctor, S., Lepro, R., Kafura, D., and Shah, S. (2003). First experiences using xacml for access control in distributed systems. In *Proceedings*

- of the 2003 ACM workshop on XML security, XMLSEC '03*, pages 25–37, New York, NY, USA. ACM.
- [Malone et al., 2010] Malone, J., Holloway, E., Adamusiak, T., Kapushesky, M., Zheng, J., Kolesnikov, N., Zhukova, A., Brazma, A., and Parkinson, H. (2010). Modeling sample variables with an experimental factor ontology. *Bioinformatics*, 26(8):1112–1118.
- [Mann et al., 1983] Mann, W. C., Matthiessen, Christian M. I. M, a., and University of Southern California, M. d. R. I. S. I. (1983). Nigel : A systemic grammar for text generation / william c. mann and christian m. i. m. matthiessen. [microform].
- [Maybury, 1995] Maybury, M. T. (1995). Research in multimedia and multimodal parsing and generation. *Artif. Intell. Rev.*, 9(2-3):103–127.
- [Maynard, 2003] Maynard, D. (2003). Multi-source and multilingual information extraction. *Expert Update*, 6:11–15.
- [Maynard et al., 2008] Maynard, D., Peters, W., and Li, Y. (2008). Evaluating evaluation metrics for ontology-based applications: Infinite reflection. In *LREC*.
- [Maynard et al., 2005] Maynard, D., Yankova, M., Kourakis, A., and Kokossis, A. (2005). Ontology-based information extraction for market monitoring and technology watch. In *ESWC Workshop End User Aspects of the Semantic Web*, Heraklion, Crete.
- [McDonald and Bolc, 1988] McDonald, D. D. and Bolc, L., editors (1988). *Natural language generation systems*. Springer-Verlag New York, Inc., New York, NY, USA.
- [McDowell and Cafarella, 2006] McDowell, L. and Cafarella, M. (2006). Ontology-driven information extraction with ontosyphon. *The Semantic Web-ISWC 2006*, pages 428–444.
- [McDowell et al., 2003] McDowell, L., Etzioni, O., Gribble, S., Halevy, A., Levy, H., Pentney, W., Verma, D., and Vlasheva, S. (2003). Mangrove: Enticing ordinary people

- onto the semantic web via instant gratification. In Fensel, D., Sycara, K., and Mylopoulos, J., editors, *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 754–770. Springer Berlin Heidelberg.
- [McDowell et al., 2004] McDowell, L., Etzioni, O., and Halevy, A. (2004). Semantic email: theory and applications. *Web Semant.*, 2(2):153–183.
- [McKeown et al., 1990] McKeown, K. R., Elhadad, M., Fukumoto, Y., Lim, J., Lombardi, C., Robin, J., Smadja, F., and Smadja, F. (1990). Natural language generation in comet. In *Current Research in Natural Language Generation*, pages 103–139.
- [Mellish, 2010] Mellish, C. (2010). Using semantic web technology to support nlg. case study: Owl finds rags. In *INLG’10*, pages –1–1.
- [Mellish and Dale, 1998] Mellish, C. and Dale, R. (1998). Evaluation in the context of natural language generation. *Computer Speech and language*, 12(4):349–374.
- [Mellish and Pan, 2008] Mellish, C. and Pan, J. Z. (2008). Natural language directed inference from ontologies. *Artif. Intell.*, 172(10):1285–1315.
- [Mellish et al., 2006] Mellish, C., Scott, D., Cahill, L., Paiva, D., Evans, R., and Reape, M. (2006). A reference architecture for natural language generation systems. *Natural language engineering*, 12(01):1–34.
- [Mendes et al., 2011] Mendes, P., Jakob, M., García-Silva, A., and Bizer, C. (2011). Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8. ACM.
- [Miles and Bechhofer, 2009] Miles, A. and Bechhofer, S. (2009). SKOS Simple Knowledge Organization System Reference. W3C Recommendation, W3C. <http://www.w3.org/TR/skos-reference/>.
- [Mitamura, 1999] Mitamura, T. (1999). Controlled language for multilingual machine translation. In *Proceedings of Machine Translation Summit VII*, pages 46–52.

- [Mizoguchi et al., 1995] Mizoguchi, R., Vanwelkenhuysen, J., and Ikeda, M. (1995). Task ontology for reuse of problem solving knowledge. In *In Proc. of KB&KS '95*, pages 46–59.
- [Namgoong and Kim, 2007] Namgoong, H. and Kim, H. (2007). Ontology-based controlled natural language editor using cfg with lexical dependency. In *ISWC/ASWC*, pages 353–366.
- [Nelson, 1965] Nelson, T. H. (1965). Complex information processing: a file structure for the complex, the changing and the indeterminate. In *Proceedings of the 1965 20th national conference*, ACM '65, pages 84–100, New York, NY, USA. ACM.
- [Niles and Pease, 2001] Niles, I. and Pease, A. (2001). Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001*, FOIS '01, pages 2–9, New York, NY, USA. ACM.
- [Nilsson et al., 2008] Nilsson, M., Powell, A., Johnston, P., and Naeve, A. (2008). Expressing Dublin Core metadata using the Resource Description Framework (RDF)0. DCMI Recommendation, DCMI. <http://dublincore.org/documents/dc-rdf/>.
- [Nordstrom et al., 1990] Nordstrom, B., Petersson, K., and Smith, J. M. (1990). *Programming in Martin-Löf's Type Theory: An Introduction*. Oxford University Press, USA.
- [Norta et al., 2010] Norta, A., Carlson, L., and Yangarber, R. (2010). Utility survey of ontology tools. *Department of Computer Science, Department of Linguistics-University of Helsinki, Finland*, 62:63–65.
- [Noy and McGuinness, 2001] Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory.

- [O'Brien, 2003] O'Brien, S. (2003). Controlling controlled english. an analysis of several controlled language rule sets. *Proceedings of EAMT-CLAW*, 3:105–114.
- [Ogden, 1935] Ogden, C. K. (1935). *The A B C of basic English (in Basic) / by C. K. Ogden ; with an account of the sounds of basic English by A. Lloyd James*. K. Paul, Trench, Trubner, London .:
- [Oh and Rudnicky, 2000] Oh, A. H. and Rudnicky, A. I. (2000). Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems - Volume 3*, ANLP/NAACL-ConvSyst '00, pages 27–32, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Onyshkevych et al., 1993] Onyshkevych, B., Okurowski, M. E., and Carlson, L. (1993). Tasks, domains, and languages. In *Proceedings of the 5th conference on Message understanding*, MUC5 '93, pages 7–17, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Oram, 2001] Oram, P. (2001). Wordnet: An electronic lexical database. christiane fellbaum (ed.). cambridge, ma: Mit press, 1998. pp. 423. *Applied Psycholinguistics*, 22(01):131–134.
- [Pereira and Warren, 1980] Pereira, F. C. N. and Warren, D. H. D. (1980). Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. *Artif. Intell.*, 13(3):231–278.
- [Pereira and Wright, 1991] Pereira, F. C. N. and Wright, R. N. (1991). Finite-state approximation of phrase structure grammars. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, ACL '91, pages 246–255, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Perna and Spector, 2004] Perna, J. and Spector, A. (2004). *Introduction to IBM Systems Journal, Special Issue on Unstructured Information Management*, 43(3):p447–448.

- [Pfisterer et al., 2008] Pfisterer, F., Nitsche, M., Jameson, A., and Barbu, C. (2008). User-Centered Design and Evaluation of Interface Enhancements to the Semantic MediaWik. In *Proceedings of Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges*. CEUR Workshop Proceedings.
- [Popov et al., 2004] Popov, B., Kiryakov, A., Kirilov, A., Manov, D., Ognyanoff, D., and Goranov, M. (2004). Kim - semantic annotation platform. *Journal of Natural Language Engineering*, 10(3-4):375-392.
- [Power, 2012] Power, R. (2012). Owl simplified english: A finite-state language for ontology editing. In *CNL*, pages 44-60.
- [Power et al., 1998a] Power, R., Scott, D., and Evans, R. (1998a). What you see is what you meant: direct knowledge editing with natural language feedback. In *Proceedings of the 13th Biennial European Conference on Artificial Intelligence*, pages 675-681.
- [Power et al., 1998b] Power, R., Scott, D., and Evans, R. (1998b). What you see is what you meant: direct knowledge editings with natural language feedback. In Prade, H., editor, *13th European Conference on Artificial Intelligence (ECAI'98)*, pages 677-681. John Wiley and Sons, Chichester, England.
- [Power et al., 2009] Power, R., Stevens, R., Scott, D., and Rector, A. (2009). Editing owl through generated cnl.
- [Prud'hommeaux and Seaborne, 2008] Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL query language for RDF. W3C Recommendation, W3C. <http://www.w3.org/TR/rdf-sparql-query/>.
- [Quint and Vatton, 1997] Quint, V. and Vatton, I. (1997). An introduction to amaya. *World Wide Web Journal*, 2(2):39-46.
- [Ranta, 2004] Ranta, A. (2004). Grammatical Framework: A Type-Theoretical Grammar Formalism. *Journal of Functional Programming*, 14(02):145-189.

- [Rector et al., 2003] Rector, A., Rogers, J., Taweel, A., Ingram, D., Kalra, D., Milan, J., Singleton, P., Gaizauskas, R., Hepple, M., Scott, D., et al. (2003). Clef: joining up healthcare with clinical and post-genomic research.
- [Reiter, 1994] Reiter, E. (1994). Has a consensus nl generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the Seventh International Workshop on Natural Language Generation*, INLG '94, pages 163–170, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Reiter, 1995] Reiter, E. (1995). Nlg vs. templates. In *In Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 95–106.
- [Reiter and Dale, 2000] Reiter, E. and Dale, R. (2000). *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.
- [Scerri, 2010] Scerri, S. (2010). *Supporting Email-based Collaborative Work across a Social Semantic Space (PhD Thesis)*. NUI Galway.
- [Schaffert, 2006] Schaffert, S. (2006). Ikwiki: A semantic wiki for collaborative knowledge management. In *In 1st International Workshop on Semantic Technologies in Collaborative Applications (STICA06)*.
- [Schroeter et al., 2006] Schroeter, R., Hunter, J., Guerin, J., Khan, I., and Henderson, M. (2006). A synchronous multimedia annotation system for secure collaboratories. In *e-Science and Grid Computing, 2006. e-Science'06. Second IEEE International Conference on*, pages 41–41. IEEE.
- [Schroeter et al., 2003] Schroeter, R., Hunter, J., and Kosovic, D. (2003). Vannotea: A collaborative video indexing, annotation and discussion system for broadband networks. In *Knowledge capture*, pages 1–8. ACM Press (Association for Computing Machinery).

- [Schwitter, 2002] Schwitter, R. (2002). English as a formal specification language. In *Proceedings of the 13th International Workshop on Database and Expert Systems Applications*, DEXA '02, pages 228–232, Washington, DC, USA. IEEE Computer Society.
- [Schwitter, 2007] Schwitter, R. (2007). Controlled natural languages. Technical report, Centre for Language Technology, Macquarie University.
- [Schwitter et al., 2003] Schwitter, R., Ljungberg, A., and Hood, D. (2003). Ecole—a look-ahead editor for a controlled language. *EAMT-CLAW03*, pages 141–150.
- [Schwitter and Tilbrook, 2004] Schwitter, R. and Tilbrook, M. (2004). Controlled natural language meets the semanticweb. In *Proceedings of the Australasian Language Technology Workshop 2004*, pages 55–62, Sydney, Australia.
- [Seaborne, 2004] Seaborne, A. (2004). RDQL – A Query Language for RDF. W3C Member Submission, W3C.
- [Shiffman et al., 2010] Shiffman, R. N., Michel, G., Krauthammer, M., Fuchs, N. E., Kaljurand, K., and Kuhn, T. (2010). Writing clinical practice guidelines in controlled natural language. In Fuchs, N. E., editor, *Proceedings of the Workshop on Controlled Natural Language (CNL 2009)*, volume 5972 of *Lecture Notes in Computer Science*, pages 265–280, Berlin / Heidelberg, Germany. Springer.
- [Sigurd, 1991] Sigurd, B. (1991). Referent grammar in text generation. In Paris, C. L., Swartout, W. R., and Mann, W. C., editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 315–327. Kluwer, Boston.
- [Simon, 2006] Simon, S. (2006). Stats: Steve’s attempt to teach statistics. Technical report, Children’s Mercy Hospitals & Clinics, Kansas City, Missouri.
- [Sleator and Temperley, 1991] Sleator, D. D. and Temperley, D. (1991). Parsing english with a link grammar.
- [Smart, 2008] Smart, P. R. (2008). Controlled natural languages and the semantic web.

- [Smart, shed] Smart, P. R. (2008,(Unpublished)). Controlled natural languages and the semantic web. Technical report, School of Electronics and Computer Science, University of Southampton.
- [Stevens et al., 2011] Stevens, R., Malone, J., Williams, S., Power, R., and Third, A. (2011). Automating generation of textual class definitions from owl to english. *Journal of Biomedical Semantics*, 2(Suppl 2):S5.
- [Stevens and Johnson, 1983] Strevens, P. and Johnson, E. (1983). Seaspeak: A project in applied linguistics, language engineering, and eventually esp for sailors. *The ESP Journal*, 2(2):123 – 129.
- [Studer et al., 1998] Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data Knowledge Engineering*, 25(1–2):161–197.
- [Sure, 2003] Sure, Y. (2003). *Methodology, Tools and Case Studies for Ontology based Knowledge Management*. PhD thesis, Universität Karlsruhe.
- [Tallis, 2003] Tallis, M. (2003). Semantic word processing for content authors. In *Proceedings of the Knowledge Markup & Semantic Annotation Workshop, Florida, USA*.
- [Thompson et al., 2005] Thompson, C. W., Pazandak, P., and Tennant, H. R. (2005). Talk to your semantic web. *IEEE Internet Computing*, 9(6):75–78.
- [Tijerino and Mizoguchi, 1993] Tijerino, Y. A. and Mizoguchi, R. (1993). Multis ii: Enabling end-users to design problem-solving engines via two-level task ontologies. In Aussenac-Gilles, N., Boy, G. A., Gaines, B. R., Ganascia, J.-G., Kodratoff, Y., and Linster, M., editors, *EKAW*, volume 723 of *Lecture Notes in Computer Science*, pages 340–359. Springer.
- [Tullis and Stetson, 2004] Tullis, T. S. and Stetson, J. N. (2004). A comparison of questionnaires for assessing website usability. In *Usability Professionals' Association Conference*, Minneapolis, Minnesota.

- [Turney, 2001] Turney, P. D. (2001). Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the 12th European Conference on Machine Learning, EMCL '01*, pages 491–502, London, UK, UK. Springer-Verlag.
- [Uren et al., 2006] Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., and Ciravegna, F. (2006). Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semant.*, 4(1):14–28.
- [van Heijst et al., 1997] van Heijst, G., Schreiber, A. T., and Wielinga, B. J. (1997). Using explicit ontologies in kbs development. *Int. J. Hum.-Comput. Stud.*, 46(2-3):183–292.
- [Vargas-Vera et al., 2002] Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., and Ciravegna, F. (2002). Mnm: Ontology driven semi-automatic and automatic support for semantic markup. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 213–221.
- [W3C OWL Working Group, 2009] W3C OWL Working Group (2009). OWL 2 Web Ontology Language Document Overview.
- [Wang et al., 2006] Wang, T., Li, Y., Bontcheva, K., Cunningham, H., and Wang, J. (2006). Automatic extraction of hierarchical relations from text. In *Proceedings of the 3rd European conference on The Semantic Web: research and applications, ESWC'06*, pages 215–229, Berlin, Heidelberg. Springer-Verlag.
- [Watt, 1968] Watt, W. C. (1968). Habitability. *American Documentation*, 19:338–351.
- [White, 2004] White, M. (2004). Reining in CCG chart realization. In *Proceedings of the 3rd International Conference on Natural Language Generation (INLG 2004)*.
- [White and Baldrige, 2003] White, M. and Baldrige, J. (2003). Adapting chart realization to CCG. In *Proceedings of 9th European Workshop on Natural Language Generation*, Budapest, Hungary.

- [Wilcock, 2000] Wilcock, G. (2000). Abstract an overview of shallow xml-based natural language generation.
- [Wilcock, 2005] Wilcock, G. (2005). An overview of shallow xml-based natural language generation. In *The Second Baltic Conference on Human Language Technologies, Proceedings, Tallinn, Estonia*, pages 67–78.
- [Wilks, 1990] Wilks, Y. (1990). Where am I coming from: The reversability of analysis and generation in natural language processing. In Pütz, M., editor, *Thirty Years of Linguistic Evolution*. John Benjamins Publishing Company.
- [Williams et al., 2011] Williams, S., Third, A., and Power, R. (2011). Levels of organisation in ontology verbalisation. In *Proceedings of the 13th European Workshop on Natural Language Generation, ENLG '11*, pages 158–163, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [Wimalasuriya and Dou, 2010] Wimalasuriya, D. C. and Dou, D. (2010). Ontology-based information extraction: An introduction and a survey of current approaches. *J. Inf. Sci.*, 36(3):306–323.

A Evaluation documents for CLIE/CLOnE

A.1 Training manual

An *ontology* is a formal representation of knowledge about a domain: it contains information about the objects and types of objects in that domain and the relationships between them. *Formal* here means basically “machine-readable”—the information is stored in a well-defined way so that computer programs can read and analyse it and reason with it. In recent years ontologies have become very important to scientific research because they help with the digital classification and retrieval of human-readable information (such as research papers and other documents).

An ontology consists of classes, instances and properties. The classes and instances are often drawn in a tree as shown in Figure A.1.

A *class* is a description of a set or the name of a type of thing, such as **Person** or **Document**. Classes are arranged in a hierarchy, so that the **Document** class might have several subclasses (subtypes), **Book**, **Journal** and **Article**. In this example **Document** is the “direct superclass” of **Book**, **Journal** and **Article**.

An *instance* or *individual* is one member of a class; for example, **Syntactic Structures**

is an instance of the class `Book`, and `Noam Chomsky` is an instance of the class `Person`. Because of the superclass-subclass relationship, `Syntactic Structures` is also an instance of the class `Document`.

A *property* is a relation between two classes which can be instantiated between instances. We can for example create a property to express the idea that “persons are authors of documents”, and then define an instance of this property to express the idea that “Noam Chomsky is the author of *Syntactic Structures*”. Note that a property has an inherent “direction”; in other words, the two arguments of an property cannot (usually) be interchanged: it is not the case, for example, that documents can be authors of persons.

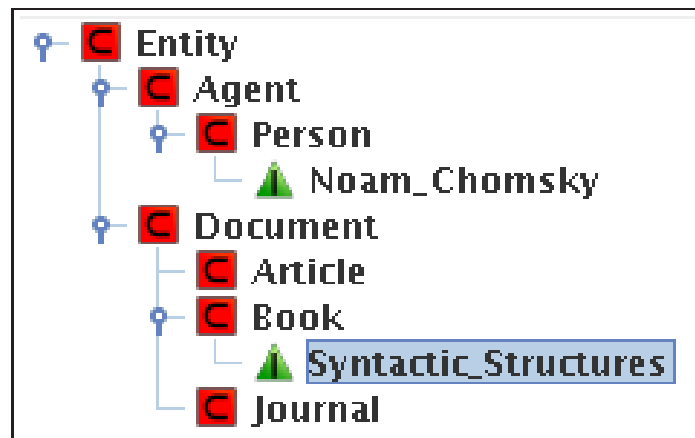


Figure A.1: Graphical depiction of classes and instances

A.1.1 CLIE How-To

The facilitator will start CLIE and load the initial data, so that you will have a window like the one shown in Figure A.2 to work with. You can click on the buttons or tabs across the top to bring up the following panes.

Messages This pane explains in detail what CLIE has just done and includes error messages. You can distinguish the error messages by the word *WARNING*, and probably ignore the *INFO* messages.

Text input In this pane (shown in Figure A.2) you will type statements in the controlled language explained below. To clear the input, select all the text with the mouse and press the backspace key. You can also edit individual parts of the text normally using the arrow and backspace keys and the mouse.

Ontology This pane (shown in Figure A.3) shows you the state of the ontology, represented by a class and instance diagram (top left), a general list of properties (below), and information about the selected class or instance (right). Click on classes or instances to change the information in the right-hand section. Before you begin the tasks, you might wish to look at the initial ontology to become familiar with it.

To run CLIE on your current input text, right click on the word *CLIE* near the top of the left-hand pane and click *Run* in the menu that appears. As you are probably aware, human language does not lend itself to precise computer processing; it contains stylistic variations, ambiguities and other features that make it difficult for computers

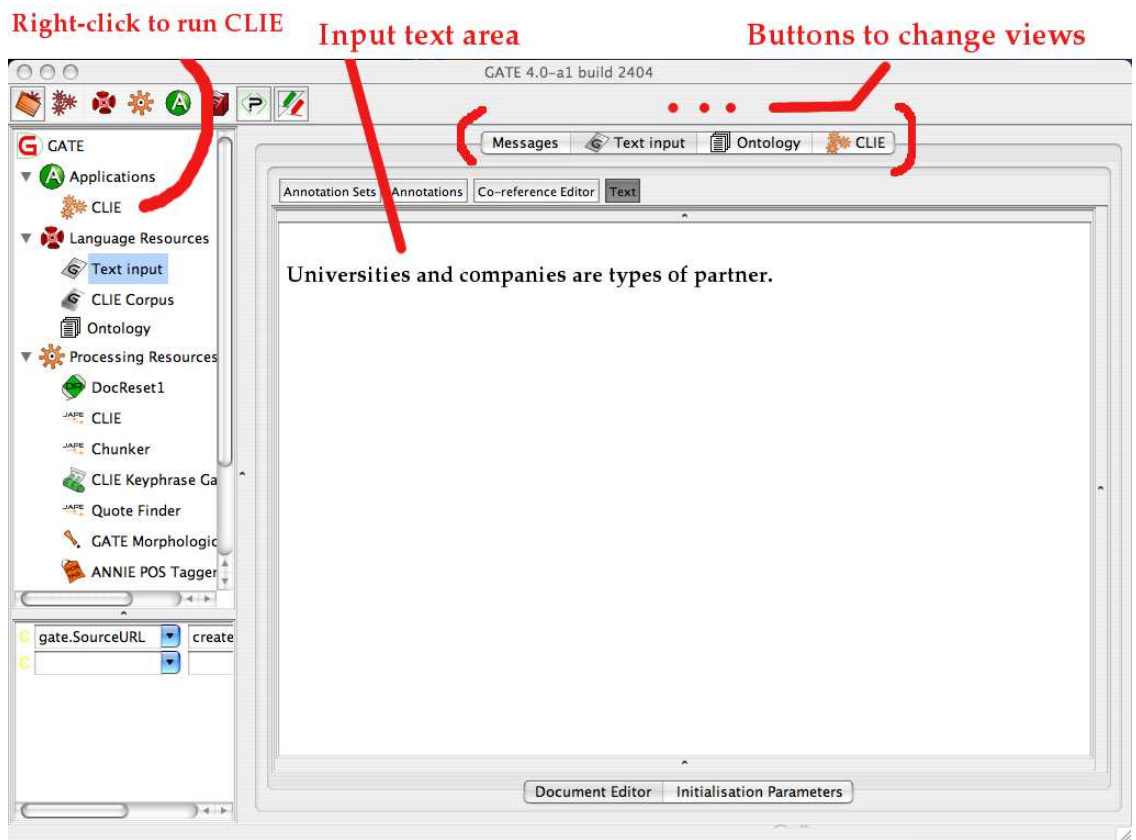


Figure A.2: CLIE Text input

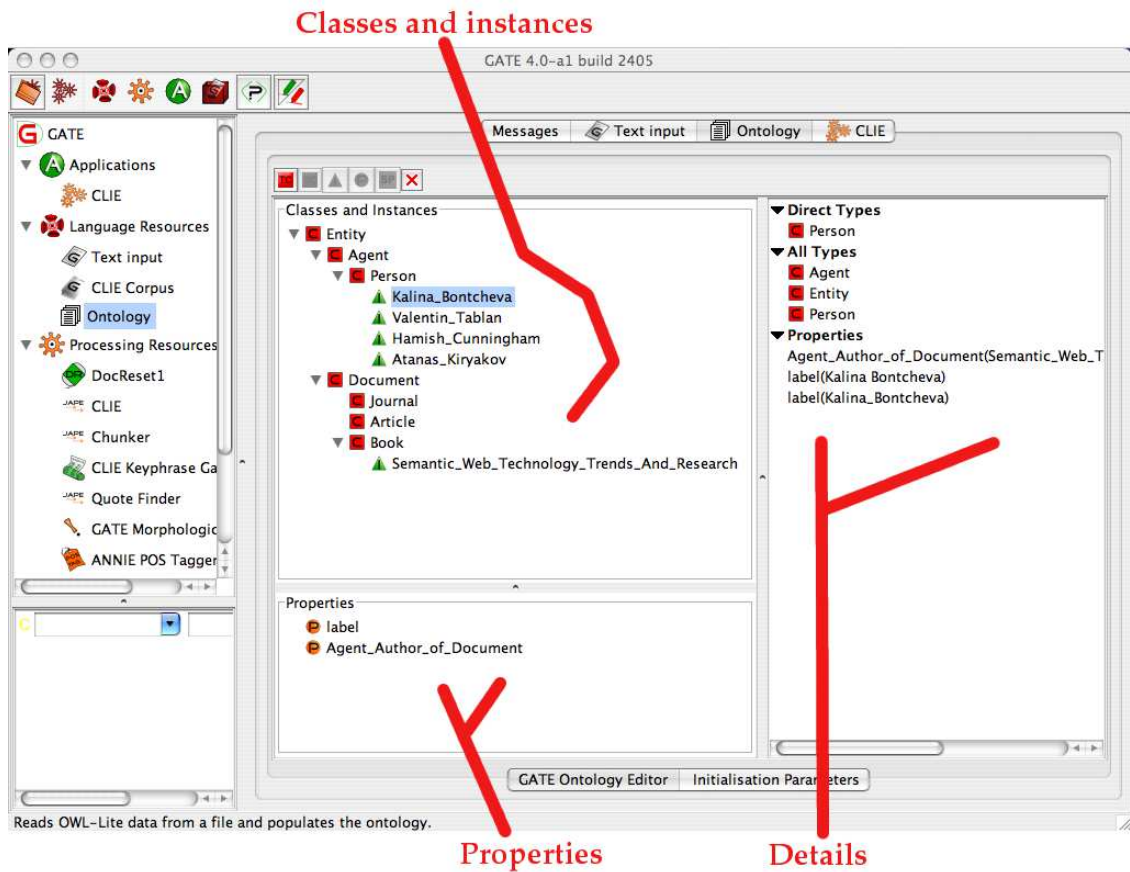


Figure A.3: CLIE Ontology viewer

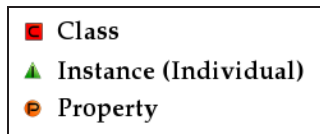


Figure A.4: Symbols used in CLIE

to interpret. One way to let people write instructions and data so that computers can handle the input correctly is to use an artificial language, such as a computer programming language. Another approach is to use a *controlled language*, which is a restricted subset of a natural language such as English. All the sentences in the controlled language are human-readable sentences in English and have a specific meaning for the computer program, but not all English sentences are valid in the controlled language. In one of the programs you will test today, you will type sentences in a controlled language and run a program that interprets them in order to add more classes, instances and properties to a simple ontology representing a digital library (a computerised record of information about documents).

The controlled language used in CLIE, called CLOnE, is designed to be easy to learn from examples. (To avoid giving you the answers to your tasks here, we provide examples about research projects.) The controlled language consists of keywords (including punctuation) and names (of classes, instances and properties). In the following examples, keywords are underlined.

- Manipulating classes in the CLOnE language

1. Universities and companies are types of partner.

The class `Partner` must already exist. Make classes `University` and `Company` direct subclasses of `Partner` and create the first two classes if they do not already exist. (A class can have more than one direct superclass.)

2. Forget that universities are types of partner.

Unlink the subclass-superclass relationship. This statement does not delete any classes.

- Manipulating instances in the CLOnE language

3. 'University of Sheffield' is a university.

Create an instance `University of Sheffield` of the class `University` (which must already exist). Because the name contains a preposition (`of`) it needs to be enclosed in quotation marks—these tell the CLIE program to treat everything between them as one name.

4. 'Smith and Sons' and 'Jones Ltd.' are companies.

Create two instances `Smith and Sons` and `Jones Ltd` of the class `Company` (which must already exist). The names are quoted because the first one contains a keyword (`and`) and the second one contains punctuation (`.`). (Figure A.5 lists all the keywords.)

- Deleting classes and instances in the CLOnE language

5. Forget 'Smith and Sons', Alice Smith and projects.

Delete the instance `Smith and Sons` and `Alice Smith` and the class `Project`.

In this statement, the list can contain a mixture of classes and instances.

- Manipulating properties the CLOnE language

6. Persons are authors of deliverables.

If the classes `Person` and `Deliverable` exist, define a property `Person Author of Deliverable` between them.

7. Forget that persons are authors of deliverables.

Delete the property defined in the last example.

8. Alice Smith and Bob Davis are authors of 'D2.3.4'.

If `Alice Smith` and `Bob Davis` are instances of `Person` and `D2.3.4` is an instance of `Deliverable`, and the property already exists, create two property definitions to indicate that they are authors of it. (`D2.3.4` must be quoted because it contains punctuation (.)).

9. Forget that Bob Davis is author of 'D2.3.4'.

Remove the property definition for one of the authors in the previous example. (This leaves the other author defined.)

- Typing the names of classes and instances

10. Names are normalized using initial upper-case letters and the base forms of words with underscores between them, so that `Deliverables` and `deliverable` both refer to the class `Deliverable`, and `Alice Smith` refers to the instance `Alice_Smith`.

11. Names containing reserved words (see Figure A.5), punctuation, prepositions (such as *of*) and determiners (**the**, **that**, **these**, etc.) must be enclosed in quo-

tation marks ('...'). For example, 'Journal of Cell Biology', 'Smith and Sons' and 'String Theory' will not be interpreted correctly without them.

In order to carry out the tasks we ask you to do, you can alternate between the ontology viewer and the input text box. When you are satisfied that your input text is probably right, click “Run” in the CLIE pane and check the results in the ontology viewer. You can make corrections by undoing mistakes with “Forget...” statements and trying again with new statements.

When you click “Run”, CLIE processes your input text from the top down, so it is important (for example) to define a new class before using it to define instances, subclasses or properties—although you can do all these steps in the same input text.

and	is	text as
are	is a	texts
are a type of	is a type of	texts as
are also called	is also called	textual
are also known as	is also known as	textual as
are called	is an	that can have
are known as	is called	that has
are types of	is known as	that have
can have	number	there are
date	number as	there is
date as	numbers	which are
dates	numbers as	which can have
dates as	numeric	which has
delete all	numeric as	which have
delete everything	string	which is
forget	string as	with value
forget all	strings	works at
forget everything	strings as	.
forget that	studies at	,
has	supervises	
have	text	

Figure A.5: Reserved words and phrases in CLOnE

A.1.2 Protégé How-To

The facilitator will start Protégé and load the initial data, so that you will have a window like the one shown in Figure B.4 to work with. The named buttons across the top have the following roles.

OWLClasses As shown in Figure B.4, this button brings up the *Subclass Explorer*, which shows a hierarchical diagram of the classes in the ontology and which you can use to select a specific class to work with, and the *Class Editor*, used for editing an individual class.

You can right-click on a class in the Subclass Explorer to get a menu which will allow you to create subclasses and individuals (instances) of existing classes, and you can drag and drop classes to move them in the class hierarchy.

Properties This produces the *Property Browser* as shown in Figure B.5, a list of the properties in the ontology.

You can create new properties by selecting the correct kind of property (“Object” or “Datatype”) and clicking the first button after the list heading (“Object properties” or “Datatype properties”). In the *Property Editor* you can select the domain and range of each property.

Individuals This produces the display shown in Figure B.6, including the *Class Browser* (similar to the Subclass Explorer), the *Instance Browser* (which shows a list of the instances of the class currently selected in the Class Browser), and the *Individual*

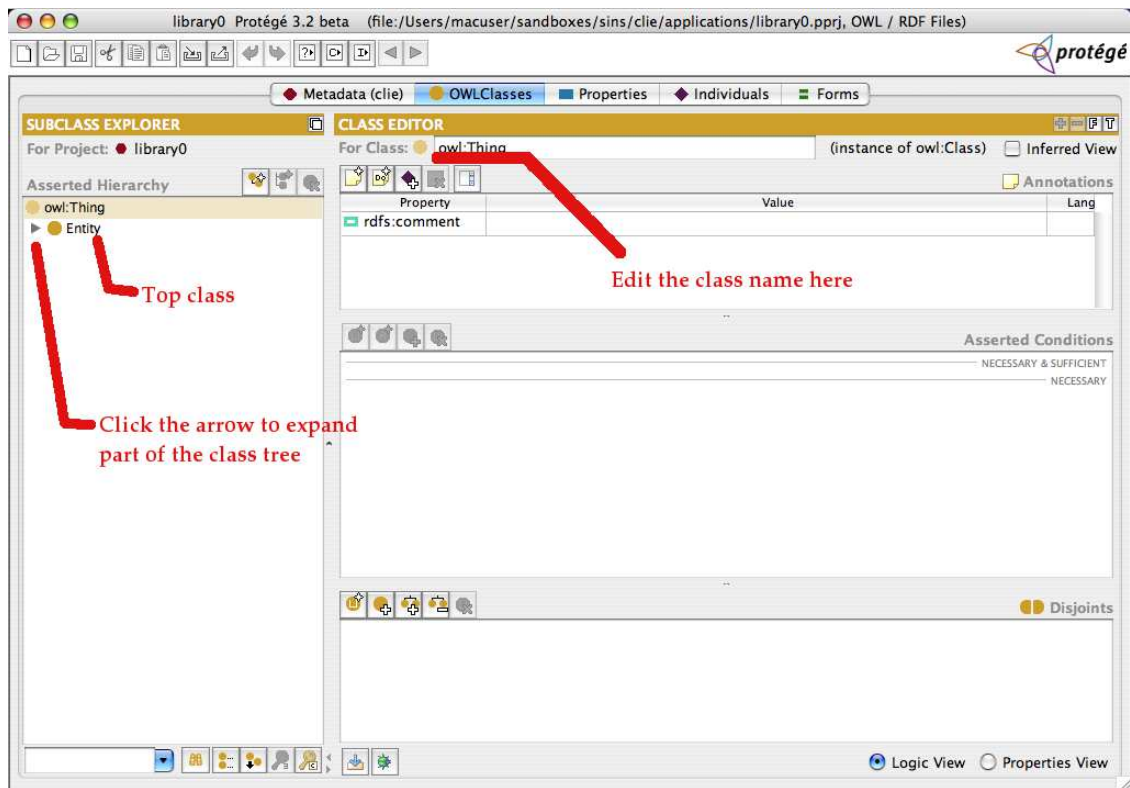


Figure A.6: Protégé's Subclass Explorer and Class Editor

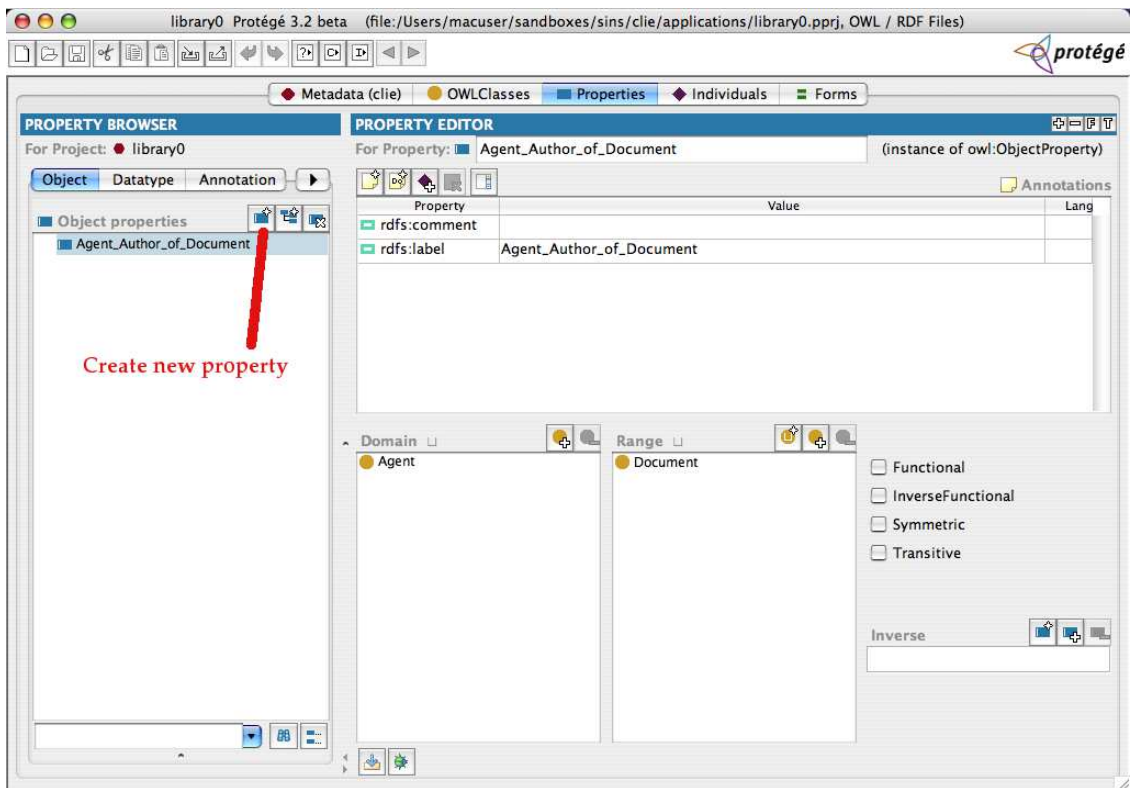


Figure A.7: Protégé's Property Browser and Property Editor

Editor, which lets you edit and fill in the property definitions of the instance selected in the Instance Browser.

The second button with a purple symbol in the Instance Browser provides another means of adding instances to the selected class.

You can ignore the **Metadata** and **Forms** buttons and panes.

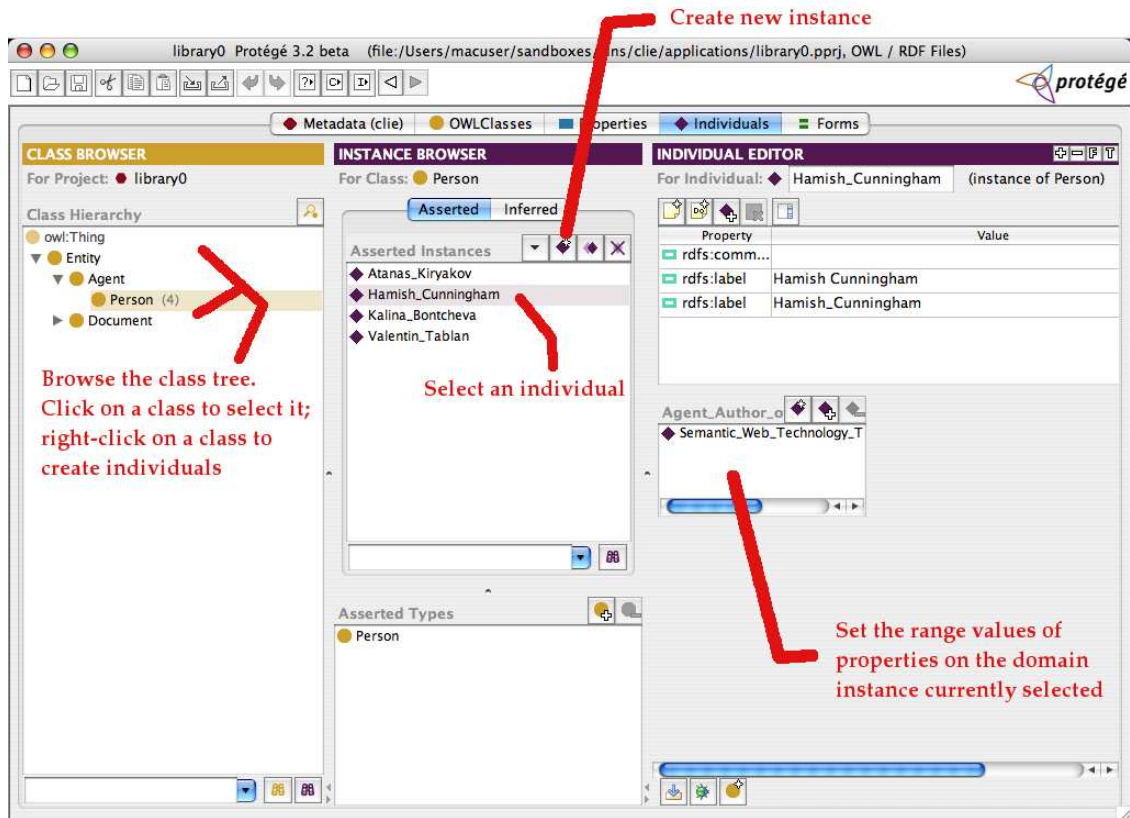


Figure A.8: Protégé’s Instance Browser and Individual Editor

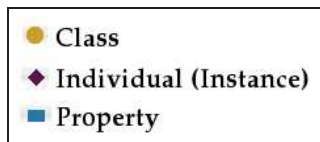


Figure A.9: Symbols used in Protégé

A.2 Test procedure, tasks and questionnaires

First we asked each subject to complete the pre-test questionnaire shown in Figure A.15.

We then gave him either CLIE or Protégé loaded with the initial ontology shown in Figure A.10 and asked him to carry out the groups of related tasks (Task List A) shown in Figure A.11, while we recorded the time taken to complete each group.

We then asked the subject to complete the questionnaire shown in Figure A.16. We then gave the subject the other tool (Protégé or CLIE, respectively) loaded with the ontology in the state that would result from correctly carrying out the tasks listed above, as shown in Figure A.12, and asked him to carry out the additional groups of tasks (Task List B) in Figure A.13 with the second tool.

We then saved the ontology to examine later for correctness; the correct result is illustrated in Figure A.14. We asked the subject to complete the questionnaire shown in Figure A.16 and then separately the one in Figures A.17 and A.18.

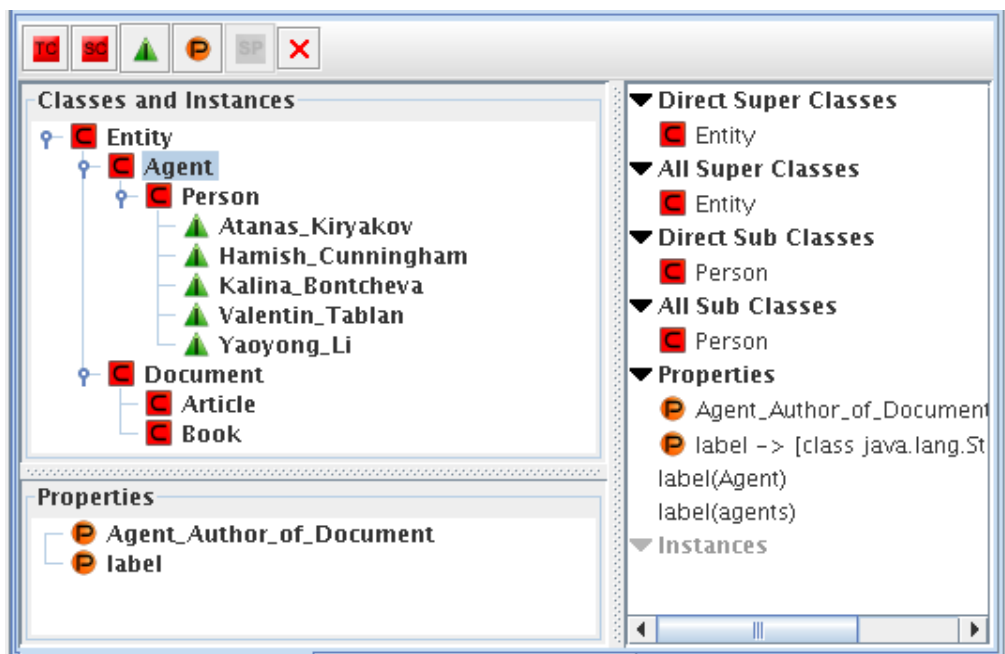


Figure A.10: Initial ontology (viewed in GATE)

- Class tasks
 - Create a subclass *Periodical* of *Document*.
 - Create a subclass *Journal* of *Periodical*.
- Instance tasks
 - Create an instance *Crossing the Chasm* of class *Article*.
 - Create an instance *Journal of Knowledge Management* of class *Journal*.
- Property tasks
 - Create a property that agents are publishers of documents.
 - Define a property that Hamish Cunningham, Kalina Bontcheva and Yaoyong Li are authors of *Crossing the Chasm*.

Figure A.11: Task list A

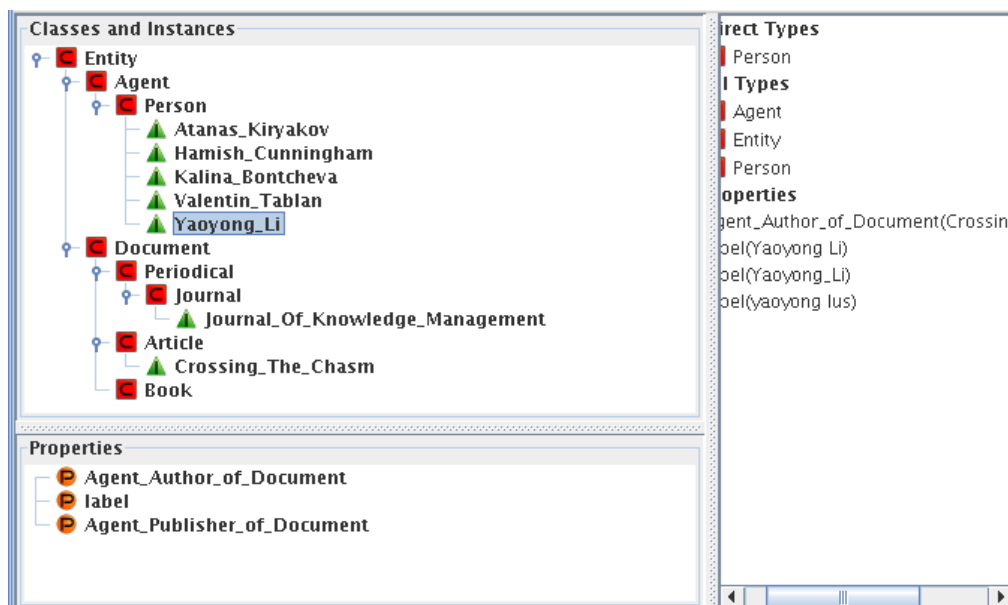


Figure A.12: Intermediate ontology (viewed in GATE)

- Class tasks
 - Create a subclass *Institution* of *Agent*.
 - Create a subclass *Company* of *Institution*.
- Instance tasks
 - Create an instance *Wiley and Sons* of class *Company*.
 - Create an instance *Trends and Research* of class *Book*.
- Property tasks
 - Define a property that *Wiley and Sons* is publisher of *Trends and Research*.
 - Define a property that *Wiley and Sons* is publisher of *Crossing the Chasm*.

Figure A.13: Task list B

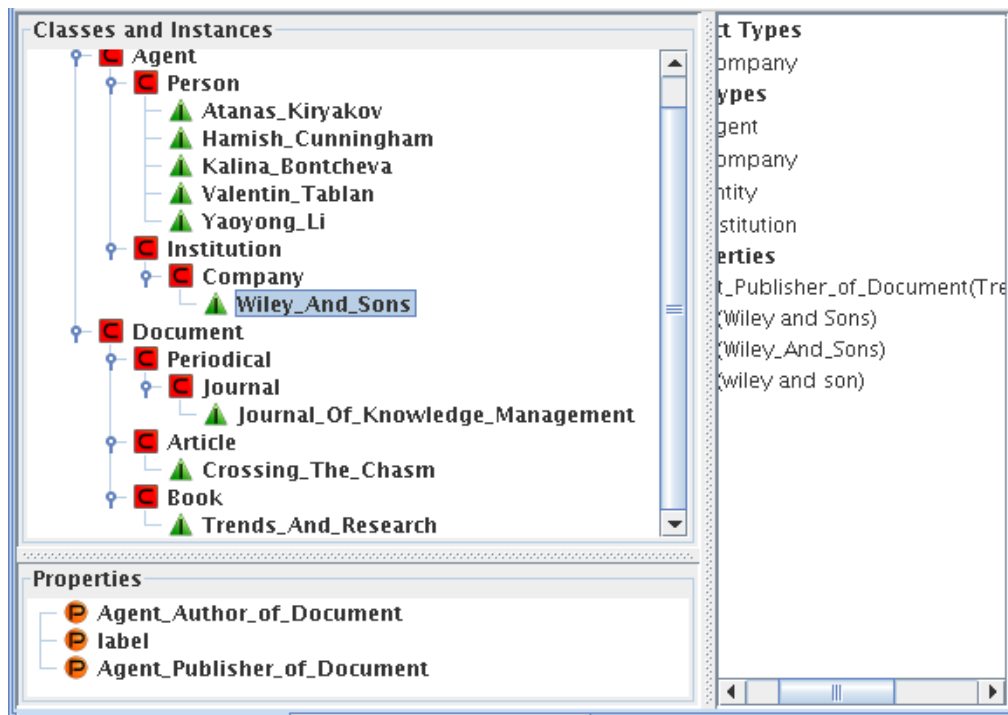


Figure A.14: Final ontology (viewed in GATE)

- | | | | | |
|---|--|--------------------------------|------------------------------------|--------------------------------|
| 1 | I understand the term “Semantic Web”. | No <input type="checkbox"/> | A little <input type="checkbox"/> | Yes <input type="checkbox"/> |
| 2 | I am familiar with Ontologies. | No <input type="checkbox"/> | A little <input type="checkbox"/> | Yes <input type="checkbox"/> |
| 3 | I have worked with Ontologies. | Never <input type="checkbox"/> | Sometimes <input type="checkbox"/> | Often <input type="checkbox"/> |
| 4 | I have built or designed a Ontologies. | Never <input type="checkbox"/> | Sometimes <input type="checkbox"/> | Often <input type="checkbox"/> |
| 5 | I understand the term “Controlled Language”. | No <input type="checkbox"/> | A little <input type="checkbox"/> | Yes <input type="checkbox"/> |
| 6 | I have used a Controlled Language. | Never <input type="checkbox"/> | Sometimes <input type="checkbox"/> | Often <input type="checkbox"/> |

Figure A.15: Pre-test questionnaire

		Strongly	Disagree	Neutral	Agree	Strongly
		disagree				agree
1	I think that I would like to use the system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	I found the system unnecessarily complex	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	I think that I would need the support of a technical person to be able to use this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	I found the various rules in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7	I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8	I found the system very cumbersome to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9	I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
10	I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure A.16: Post-test questionnaire for each system

		CLIE			CLIE			Protégé			Protégé		
1	I found one system's documentation easier to understand.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
2	I particularly disliked using one system.	disliked	<input type="checkbox"/>	disliked	<input type="checkbox"/>	neutral	<input type="checkbox"/>	disliked	<input type="checkbox"/>	disliked	<input type="checkbox"/>		
		strongly								strongly			
3	I found one system easier to use.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
4	One system was harder to learn.	much	<input type="checkbox"/>	harder	<input type="checkbox"/>	neutral	<input type="checkbox"/>	harder	<input type="checkbox"/>	much	<input type="checkbox"/>		
		harder								harder			
5	I would prefer to use one system again.	strongly	<input type="checkbox"/>	prefer	<input type="checkbox"/>	neutral	<input type="checkbox"/>	prefer	<input type="checkbox"/>	strongly	<input type="checkbox"/>		
		prefer								prefer			
6	I found one system more complicated.	much more	<input type="checkbox"/>	more	<input type="checkbox"/>	neutral	<input type="checkbox"/>	more	<input type="checkbox"/>	much more	<input type="checkbox"/>		
		complex		complex				complex		complex			
7	I found it easier to control classes and subclasses in one system.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
8	New properties were easier to create with one system.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
9	It was difficult to create instances in one system.	much	<input type="checkbox"/>	harder	<input type="checkbox"/>	neutral	<input type="checkbox"/>	harder	<input type="checkbox"/>	much	<input type="checkbox"/>		
		harder								harder			
10	It was awkward to fill (or define) properties in one system.	very	<input type="checkbox"/>	awkward	<input type="checkbox"/>	neutral	<input type="checkbox"/>	awkward	<input type="checkbox"/>	very	<input type="checkbox"/>		
		awkward								awkward			

Figure A.17: Post-test questionnaire comparing the tools

Do you have any comments on any of the systems?

Do you have any specific problems to report?

Do you have any suggestions for improving CLIE?

Figure A.18: Post-test questionnaire comparing the tools

B Evaluation documents for Round Trip Ontology

Authoring ROA

B.1 Training manual

An *ontology* is a formal representation of knowledge about a domain: it contains information about the objects and types of objects in that domain and the relationships between them. *Formal* here means basically “machine-readable”—the information is stored in a well-defined way so that computer programs can read and analyse it and reason with it. In recent years ontologies have become very important to scientific research because they help with the digital classification and retrieval of human-readable information (such as research papers and other documents).

An ontology consists of classes, instances and properties. The classes and instances are often drawn in a tree as shown in Figure C.1.

A *class* is a description of a set or the name of a type of thing, such as **Researcher** or **Conference**. Classes are arranged in a hierarchy, so that the **Researcher** class might have several subclasses (subtypes), **Staff**, **Professor** and **Senior Researcher**. In this example **Researcher** is the “direct superclass” of **Staff**, **Student** and **Senior Researcher**.

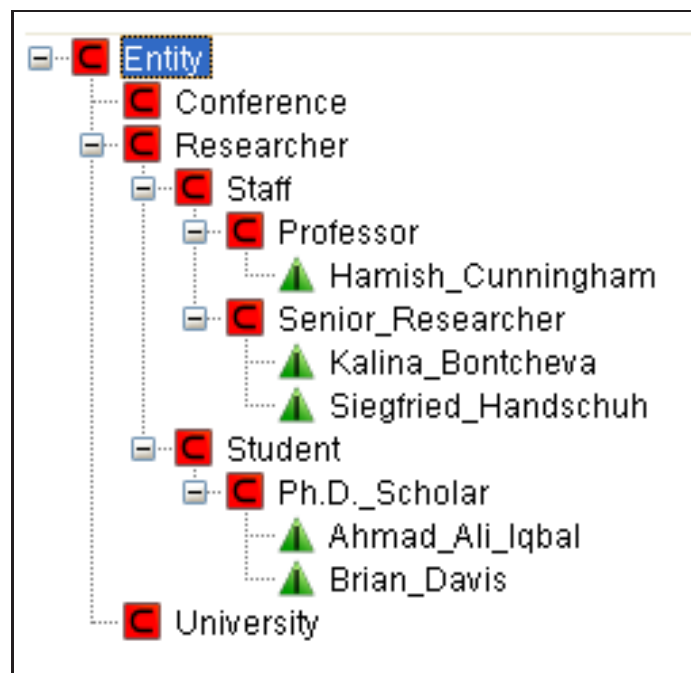


Figure B.1: Graphical depiction of classes and instances

An *instance* or *individual* is one member of a class; for example, **Brian Davis** is an instance of the class **Student**, and **Siegfried Handschuh** is an instance of the class **Senior Researcher**. Because of the superclass-subclass relationship, **Brian Davis** is also an instance of the class **Researcher**.

A *property* is a relation between two classes which can be instantiated between instances. We can for example create a property to express the idea that “Senior Researcher supervise Students”, and then define an instance of this property to express the idea that “Siegfried Handschuh supervises *Brian Davis*”. Note that a property has an inherent “direction”; in other words, the two arguments of an property cannot (usually) be interchanged: it is not the case, for example, that Students can supervise Senior Researchers.

B.1.1 Round Trip Ontology Authoring (ROA) - How-To

The facilitator will start ROA and load the initial textual data, so that you will have a window like the one shown in Figure B.2 to work with. You can click on the buttons or tabs across the top to bring up the following panes.

Messages This pane explains in detail what ROA has just done and includes error messages. You can distinguish the error messages by the word *WARNING*, and probably ignore the *INFO* messages.

Text input In this pane (shown in Figure B.2) you will edit statements in the controlled language explained below. To clear any input, select all the text with the mouse and press the backspace key. We recommend editing individual parts. This can be achieved as you would edit text normally using the arrow and backspace keys and the mouse.

Ontology This pane (shown in Figure B.3) shows you the state of the ontology, represented by a class and instance diagram (top left), a general list of properties (below), and information about the selected class or instance (right). Click on classes or instances to change the information in the right-hand section. Before you begin the tasks, you might wish to look at the initial ontology to become familiar with it.

To enable RoundTrip Ontology Authoring on your current input text, right click on the word *CLIE* near the top of the left-hand pane and click *Run* in the menu that appears.

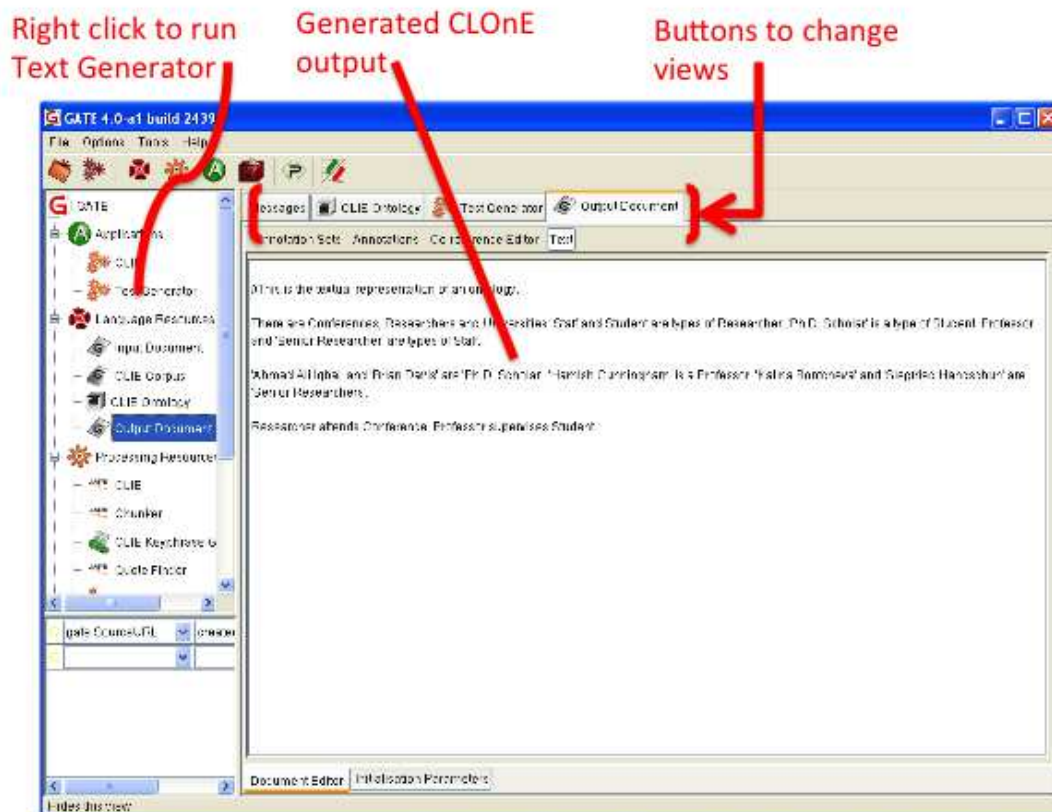


Figure B.2: CLOnE Text input

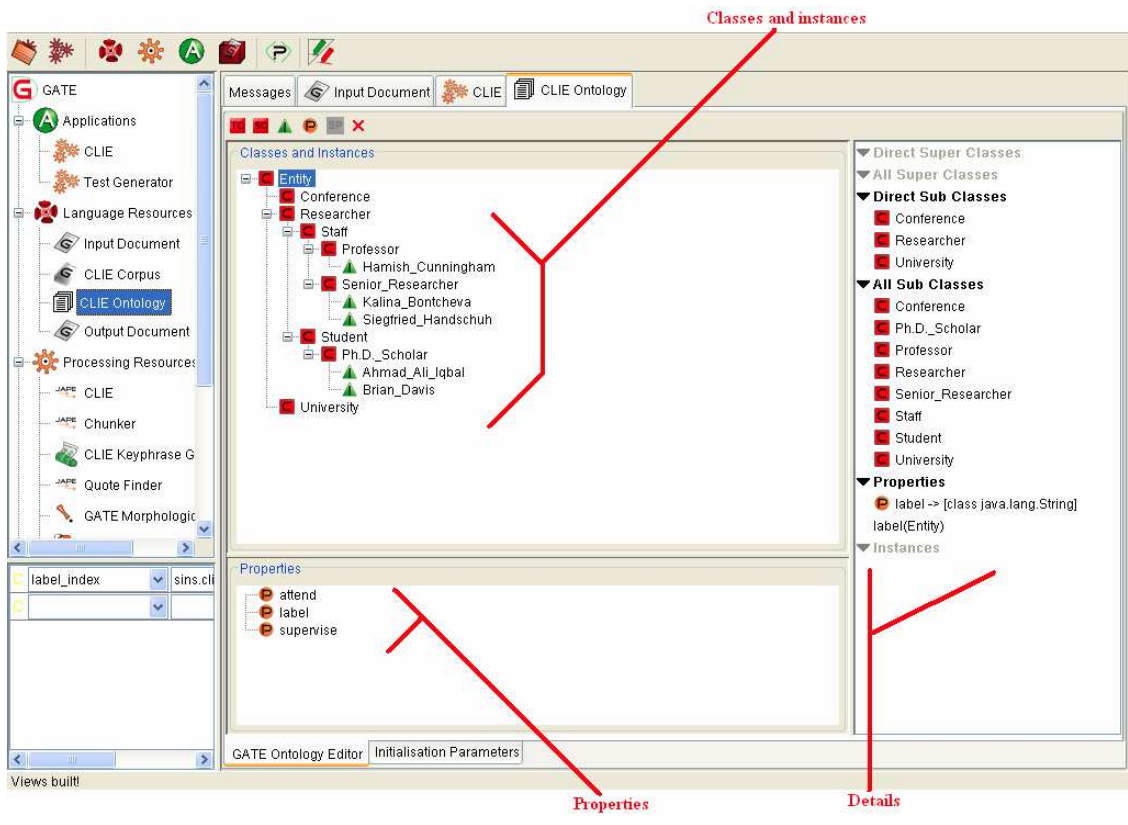


Figure B.3: ROA Ontology viewer

As you are probably aware, human language does not lend itself to precise computer processing; it contains stylistic variations, ambiguities and other features that make it difficult for computers to interpret. One way to let people write instructions and data so that computers can handle the input correctly is to use an artificial language, such as a computer programming language. Another approach is to use a *controlled language*, which is a restricted subset of a natural language such as English. All the sentences in the controlled language are human-readable sentences in English and have a specific meaning for the computer program, but not all English sentences are valid in the controlled language. In one of the programs you will test today, you will type sentences in a controlled language and run a program that interprets them in order to add more classes, instances and properties to a simple ontology representing a knowledge about academic institutions (a computerised record of information about universities).

The controlled language, CLOnE used in the software called CLIE is designed to be easy to learn from examples. (In previous work to avoid giving a participant the answers to his/her tasks here, we provided examples about research projects.). However in this evaluation we are investigating the effect of using Natural Language Text generated from the Ontology using the CLIE Text generator as a method of easing the learning experience of the user unfamiliar with CLOnE. In short, *the participant is expected to edit the text generated in the CLOnE or the purposes of updating the Ontology.*

CLOnE consists of keywords (including punctuation) and names (of classes, instances and properties). In the following examples, keywords are underlined. The reader should

reference the CLOnE text outputted by the CLIE text generator B.2 as the the basis for the examples below.

- Manipulating classes

1. There are projects and partners.

Create two new classes, Project and Partner, directly under the top class Entity.

2. There are Researchers, Universities and Conferences.

Students and Staff are types of Researchers.

Professors and Senior Researchers are types of Staff.

Make classes PostDoc and Research Assistant direct subclasses of Staff.

Edit the above line as such: Professors,Senior Researchers,PostDocs and Research Assistants are types of Staff.

- Manipulating instances

3. 'Ph.D. Scholar' is a type of Student.

Create an instance M.Sc. Scholar of the class Student Because the name contains a punctuation(.) it needs to be enclosed in quotation marks—these tell the CLIE program to treat everything between them as one name. Edit the text as follows: 'Ph.D. Scholar' and 'M.Sc. Scholar' are types of

Student

4. Ahmad Ali Iqbal and Brian Davis are 'PhD Scholars'.

Create two new instances Simon Scerri and Knud Moeller of the class 'PhD Scholar' (which must already exist). The class name is quoted because it contains contains a punctuation (.). Edit the text as follows:

5. Simon Scerri and Knud are 'PhD Scholars'.

- Deleting classes and instances

6. Forget that Ahmad Ali Iqbal and Brian Davis are 'PhD Scholars'. Delete the instances Ahmad Ali Iqbal and Brian Davis Copy and paste the line Ahmad Ali Iqbal and Brian Davis and insert Forget that before it.

- Manipulating properties

7. Researchers attend conferences .

If the classes Researcher and Conferences exist, define a property Researcher submits to Conference between them. Simply replace attend with submit to.

8. Professor supervises students.

If Hamish Cunningham and Ahmad Ali Iqbal are instances of Professor and Student and the property supervises already exists, create a new prop-

erty definition to indicate that `Hamish Cunningham Supervises Ahmad Ali Iqbal`. Simply replace `Professor` with `Hamish Cunningham` and `students` with `Ahmad Ali Iqbal`.

- Typing the names of classes and instances
 9. Names are normalised using initial upper-case letters and the base forms of words with underscores between them, so that `Deliverables` and `deliverable` both refer to the class `Deliverable`, and `Alice Smith` refers to the instance `Alice_Smith`.
 10. Names containing reserved words (see Figure A.5), punctuation, prepositions (such as *of*) and determiners (**the**, **that**, **these**, etc.) must be enclosed in quotation marks (`'...'`). For example, `'Journal of Cell Biology'`, `'Smith and Sons'` and `'String Theory'` will not be interpreted correctly without them.

In order to carry out the tasks we ask you to do, you can alternate between the ontology viewer and the input text box. When you are satisfied that your input text is probably right, click “Run” in the CLIE pane and check the results in the ontology viewer. You can make corrections by undoing mistakes with “Forget...” statements and trying again with new statements.

When you click “Run”, CLIE processes your input text from the top down, so it is important (for example) to define a new class before using it to define instances, subclasses

or properties—although you can do all these steps in the same input text.

B.1.2 Protégé How-To

The facilitator will start Protégé and load the initial data, so that you will have a window like the one shown in Figure B.4 to work with. The named buttons across the top have the following roles.

OWLClasses As shown in Figure B.4, this button brings up the *Subclass Explorer*, which shows a hierarchical diagram of the classes in the ontology and which you can use to select a specific class to work with, and the *Class Editor*, used for editing an individual class.

You can right-click on a class in the Subclass Explorer to get a menu which will allow you to create subclasses and individuals (instances) of existing classes, and you can drag and drop classes to move them in the class hierarchy.

Properties This produces the *Property Browser* as shown in Figure B.5, a list of the properties in the ontology.

You can create new properties by selecting the correct kind of property (“Object” or “Datatype”) and clicking the first button after the list heading (“Object properties” or “Datatype properties”). In the *Property Editor* you can select the domain and range of each property.

Individuals This produces the display shown in Figure B.6, including the *Class Browser* (similar to the Subclass Explorer), the *Instance Browser* (which shows a list of the instances of the class currently selected in the Class Browser), and the *Individual*

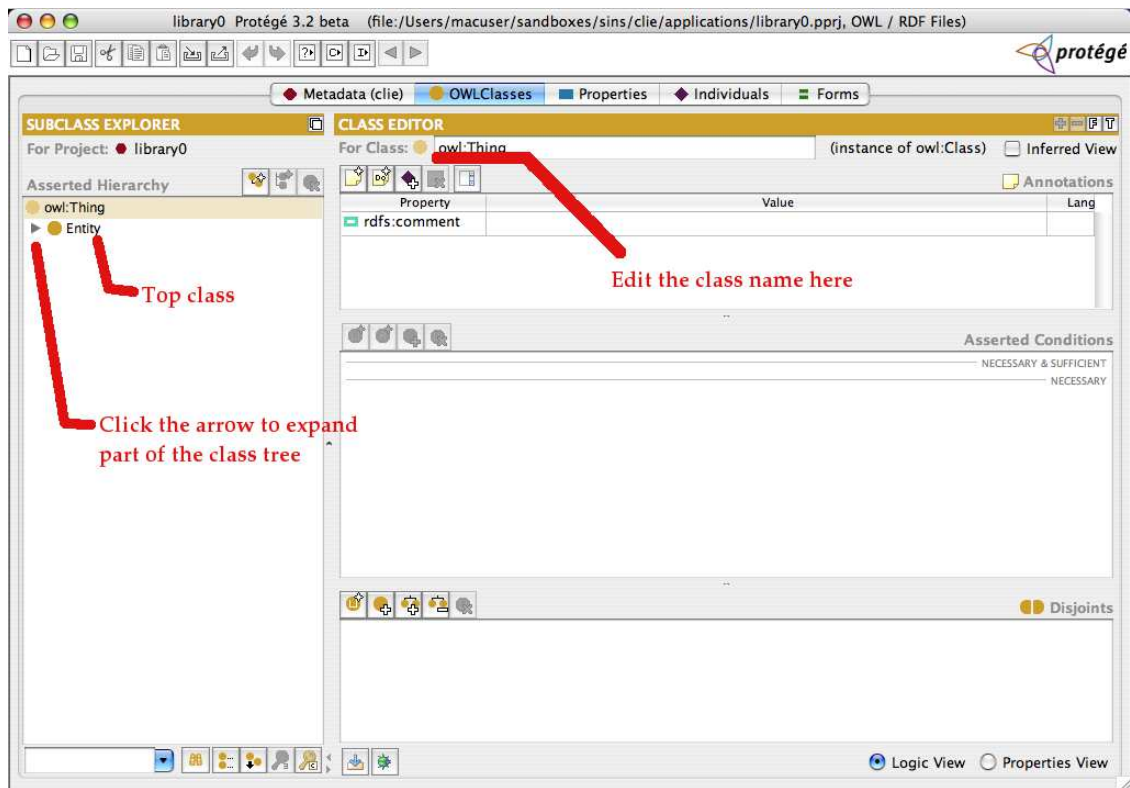


Figure B.4: Protégé's Subclass Explorer and Class Editor

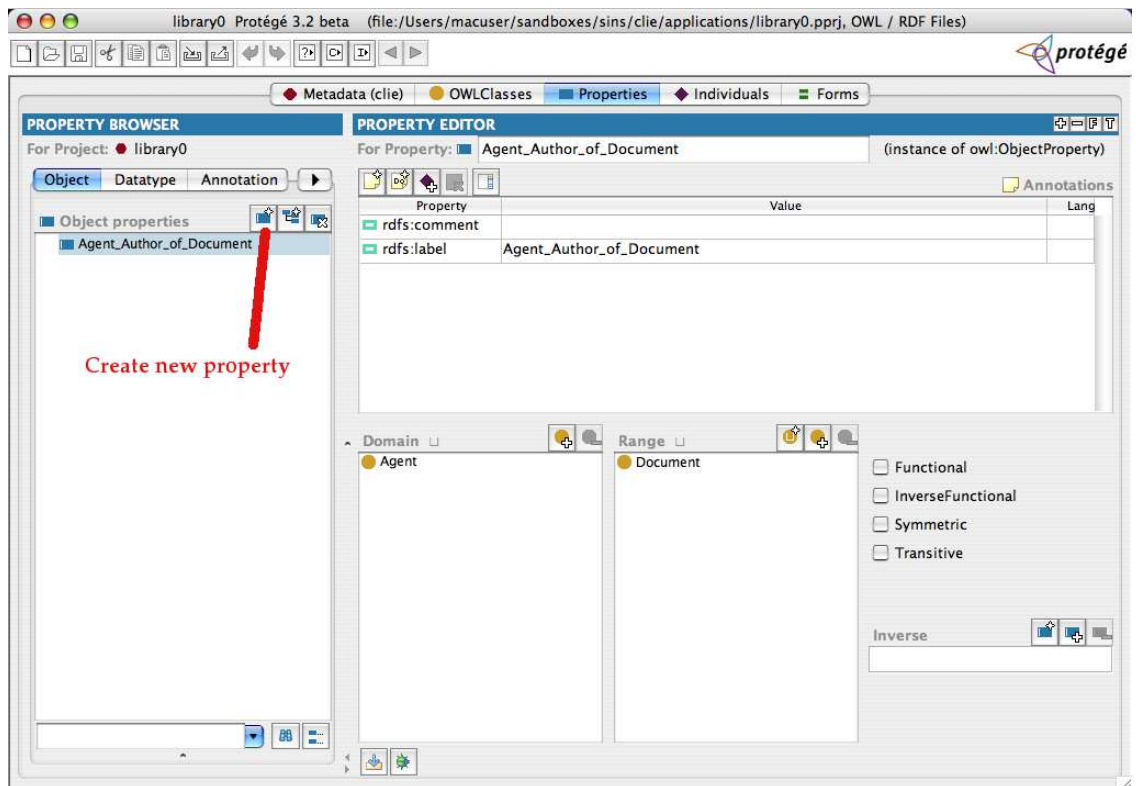


Figure B.5: Protégé's Property Browser and Property Editor

Editor, which lets you edit and fill in the property definitions of the instance selected in the Instance Browser.

The second button with a purple symbol in the Instance Browser provides another means of adding instances to the selected class.

You can ignore the **Metadata** and **Forms** buttons and panes.

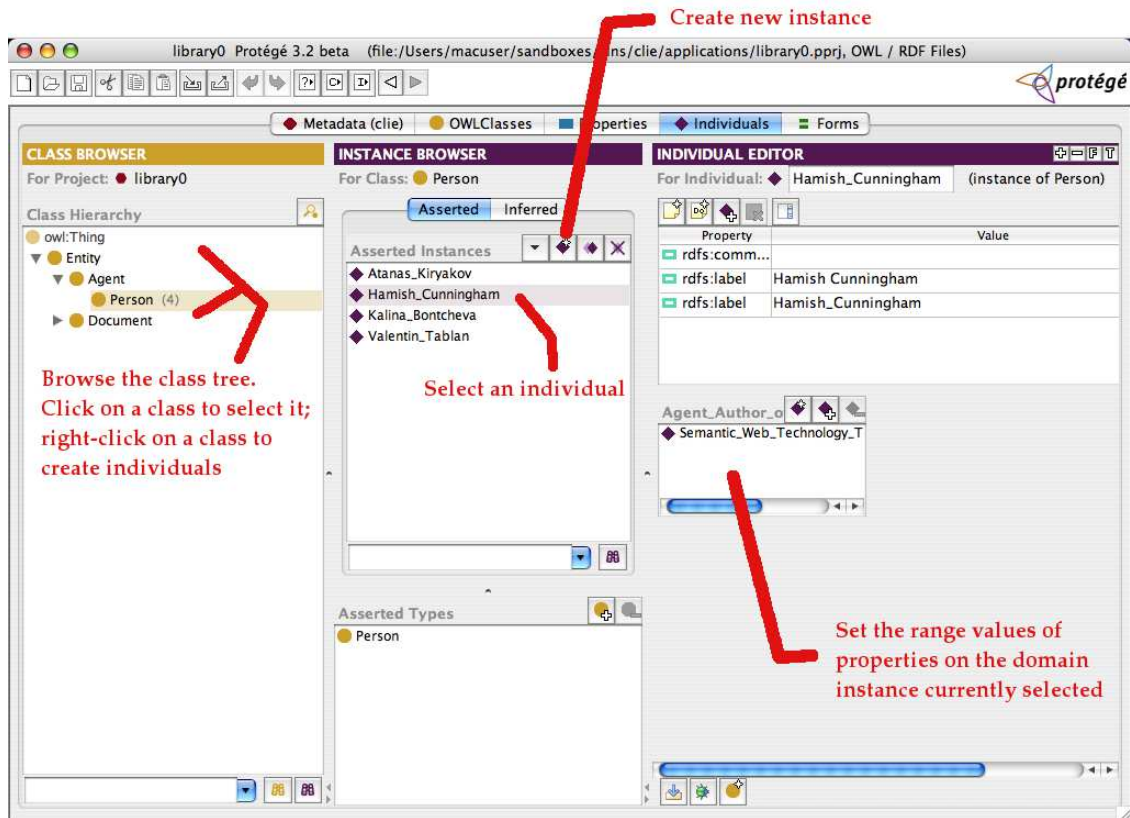


Figure B.6: Protégé's Instance Browser and Individual Editor

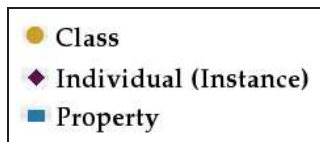


Figure B.7: Symbols used in Protégé

B.2 Test procedure, tasks and questionnaires

First we asked each subject to complete the pre-test questionnaire shown in Figure B.13.

We then gave him either CLOnE outputted from the CLIE Text generator (collectively know as ROA) or loaded with the initial ontology shown in Figure B.3 and asked him to carry out the groups of related tasks (Task List A) shown in Figure B.9, while we recorded the time taken to complete each group.

We then asked the subject to complete the questionnaire shown in Figure B.14. We then gave the subject the other tool (Protégé or ROA, respectively) loaded with the ontology in the state that would result from correctly carrying out the tasks listed above, as shown in Figure B.8, and asked him

her to carry out the additional groups of tasks (Task List B) in Figure B.11 with the second tool.

We then saved the ontology to examine later for correctness; the correct result is illustrated in Figure B.12. We asked the subject to complete the questionnaire shown in Figure B.14 and then separately the one in Figures B.15 and B.16.

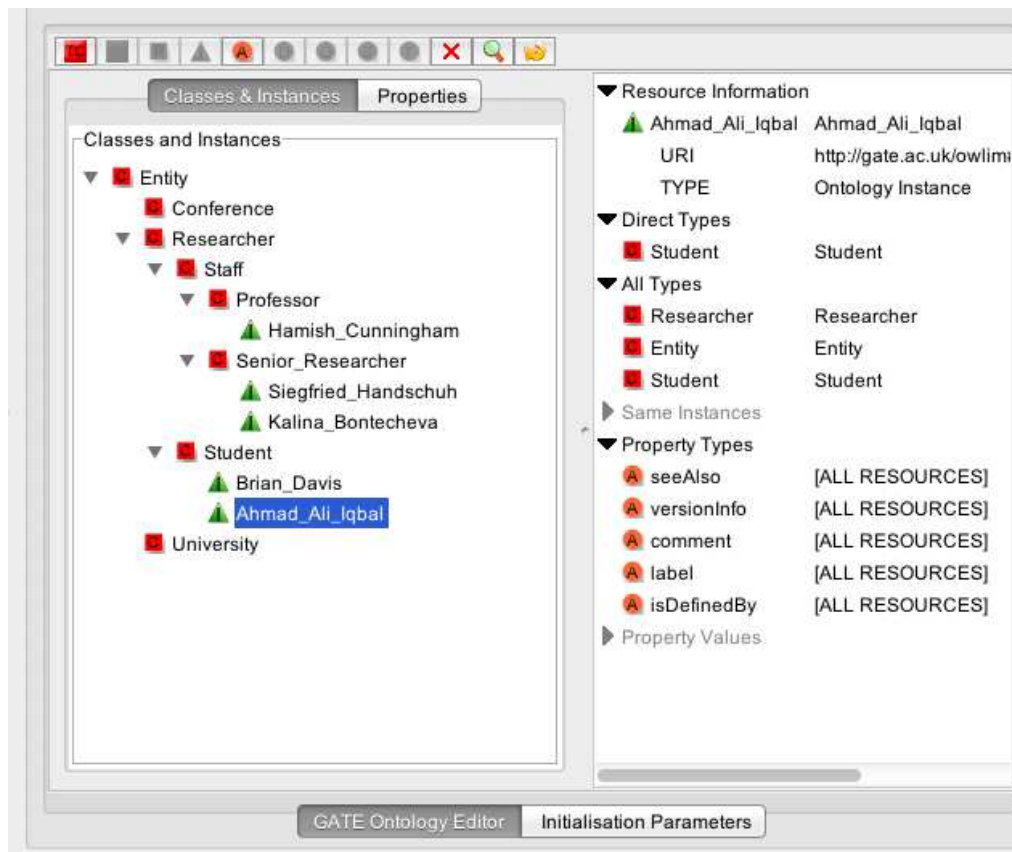


Figure B.8: Initial ontology (viewed in GATE)

- Class tasks
 - Create a subclass *Institute* of *University*.
 - Create a subclass *Workshop* of *Conference*.
- Instance tasks
 - Create an instance *International Semantic Web Conference* of class *Conference*.
 - Create an instance *DERI* of class *Institute*.
- Property tasks
 - Create a property that Senior Researchers supervise students.
 - Define a property that *Siegfried Handschuh* supervises *Brian Davis*.

Figure B.9: Task list A

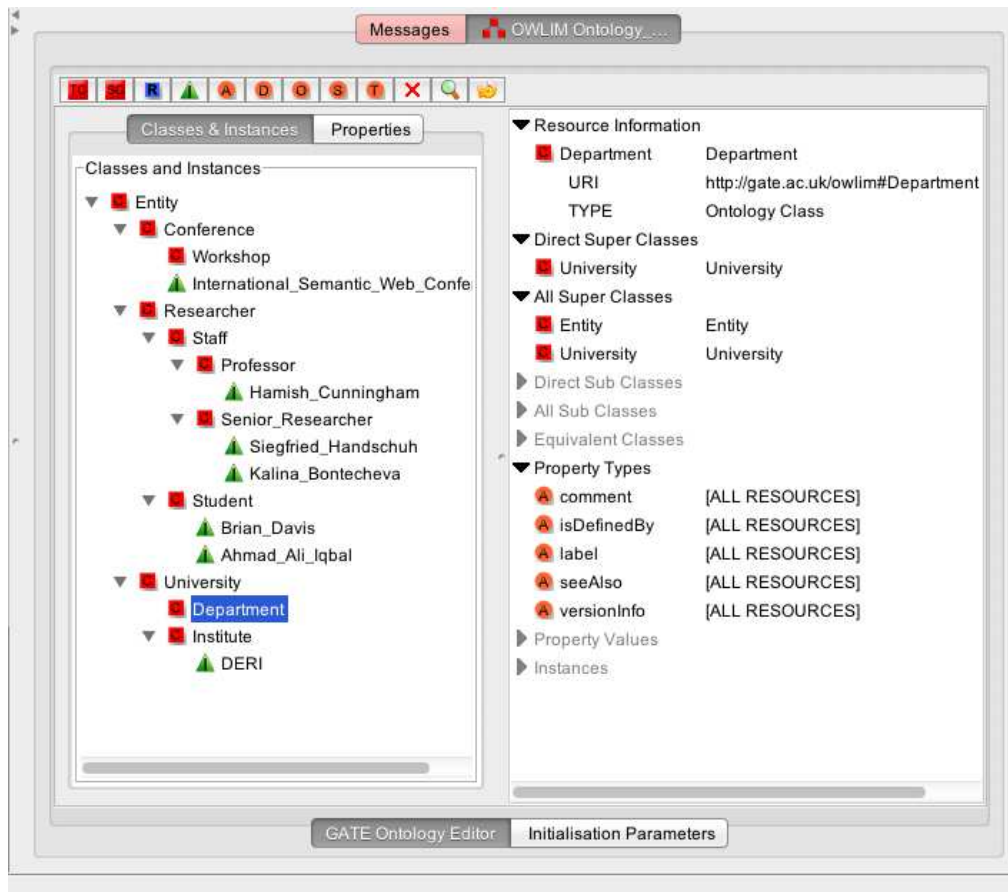


Figure B.10: Intermediate ontology (viewed in GATE)

- Class tasks
 - Create a subclass *Department* of *University*.
 - Create a subclass *Research Group* of *Department*.
- Instance tasks
 - Create an instance *SMILE* of class *Research Group*.
 - Create an instance *Stefan Decker* of class *Professor*.
- Property tasks
 - Define a property that *Senior Researcher* works for *Professor*.
 - Define a property that *Siegfried Handschuh* works for *Stefan Decker*.

Figure B.11: Task list B

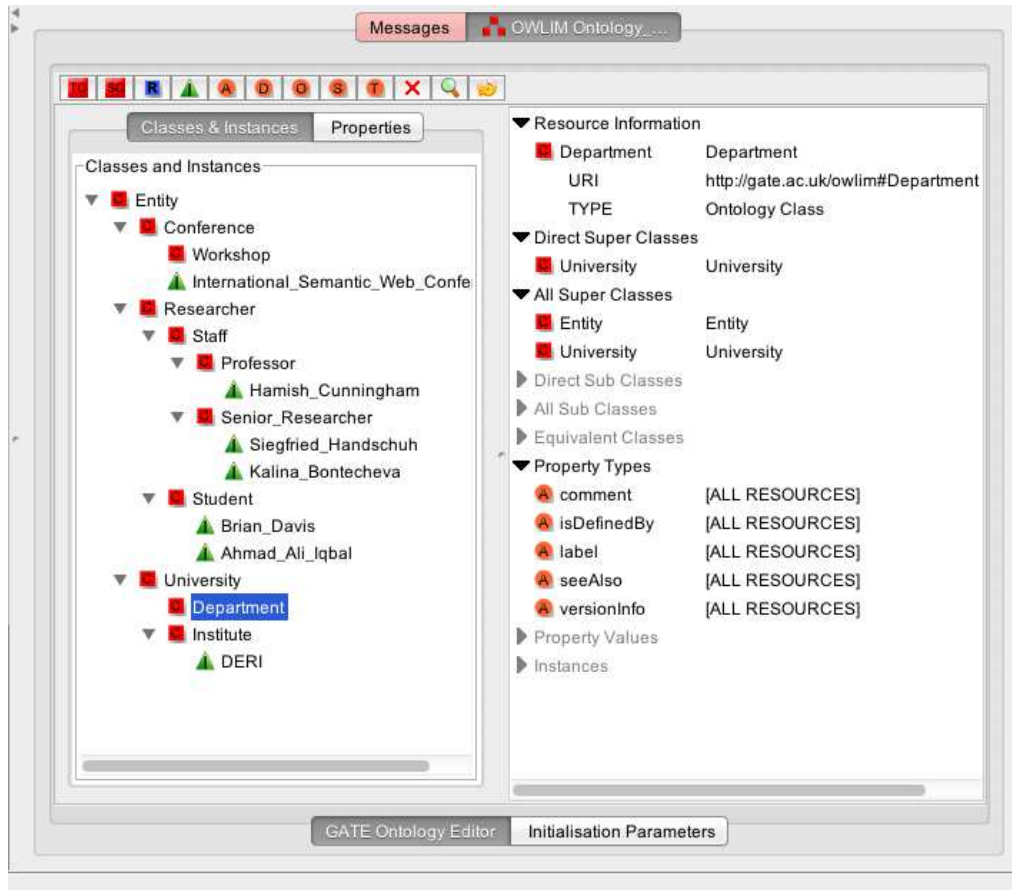


Figure B.12: Final ontology (viewed in GATE)

- | | | | | |
|---|--|--------------------------------|------------------------------------|--------------------------------|
| 1 | I understand the term “Semantic Web”. | No <input type="checkbox"/> | A little <input type="checkbox"/> | Yes <input type="checkbox"/> |
| 2 | I am familiar with Ontologies. | No <input type="checkbox"/> | A little <input type="checkbox"/> | Yes <input type="checkbox"/> |
| 3 | I have worked with Ontologies. | Never <input type="checkbox"/> | Sometimes <input type="checkbox"/> | Often <input type="checkbox"/> |
| 4 | I have built or designed a Ontologies. | Never <input type="checkbox"/> | Sometimes <input type="checkbox"/> | Often <input type="checkbox"/> |
| 5 | I understand the term “Controlled Language”. | No <input type="checkbox"/> | A little <input type="checkbox"/> | Yes <input type="checkbox"/> |
| 6 | I have used a Controlled Language. | Never <input type="checkbox"/> | Sometimes <input type="checkbox"/> | Often <input type="checkbox"/> |

Figure B.13: Pre-test questionnaire

		Strongly	Disagree	Neutral	Agree	Strongly
		disagree				agree
1	I think that I would like to use the system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	I found the system unnecessarily complex	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	I think that I would need the support of a technical person to be able to use this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	I found the various rules in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7	I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8	I found the system very cumbersome to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9	I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
10	I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure B.14: Post-test questionnaire for each system

		ROA			ROA			Protégé			Protégé		
1	I found one system's documentation easier to understand.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
2	I particularly disliked using one system.	disliked	<input type="checkbox"/>	disliked	<input type="checkbox"/>	neutral	<input type="checkbox"/>	disliked	<input type="checkbox"/>	disliked	<input type="checkbox"/>		
		strongly								strongly			
3	I found one system easier to use.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
4	One system was harder to learn.	much	<input type="checkbox"/>	harder	<input type="checkbox"/>	neutral	<input type="checkbox"/>	harder	<input type="checkbox"/>	much	<input type="checkbox"/>		
		harder								harder			
5	I would prefer to use one system again.	strongly	<input type="checkbox"/>	prefer	<input type="checkbox"/>	neutral	<input type="checkbox"/>	prefer	<input type="checkbox"/>	strongly	<input type="checkbox"/>		
		prefer								prefer			
6	I found one system more complicated.	much more	<input type="checkbox"/>	more	<input type="checkbox"/>	neutral	<input type="checkbox"/>	more	<input type="checkbox"/>	much more	<input type="checkbox"/>		
		complex		complex				complex		complex			
7	I found it easier to control classes and subclasses in one system.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
8	New properties were easier to create with one system.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
9	It was difficult to create instances in one system.	much	<input type="checkbox"/>	harder	<input type="checkbox"/>	neutral	<input type="checkbox"/>	harder	<input type="checkbox"/>	much	<input type="checkbox"/>		
		harder								harder			
10	It was awkward to fill (or define) properties in one system.	very	<input type="checkbox"/>	awkward	<input type="checkbox"/>	neutral	<input type="checkbox"/>	awkward	<input type="checkbox"/>	very	<input type="checkbox"/>		
		awkward								awkward			

Figure B.15: Post-test questionnaire comparing the tools

Do you have any comments on any of the systems?

Do you have any specific problems to report?

Do you have any suggestions for improving ROA?

Figure B.16: Post-test questionnaire comparing the tools

C Evaluation documents for Controlled Annotation

C.1 Training manual

C.1.1 Ontologies

An *ontology* is a formal representation of knowledge about a domain: it contains information about the objects and types of objects in that domain and the relationships between them. *Formal* here means basically “machine-readable”—the information is stored in a well-defined way so that computer programs can read and analyse it and reason with it. In recent years ontologies have become very important to scientific research because they help with the digital classification and retrieval of human-readable information (such as research papers and other documents).

An ontology consists of classes, instances and properties. The classes and instances are often drawn in a tree as shown in Figure C.1.

C.1.2 Semantic Annotation

Semantic Annotation, in the context of this evaluation consists of attaching a set of instances and or relational metadata to a textual document. We distinguish between

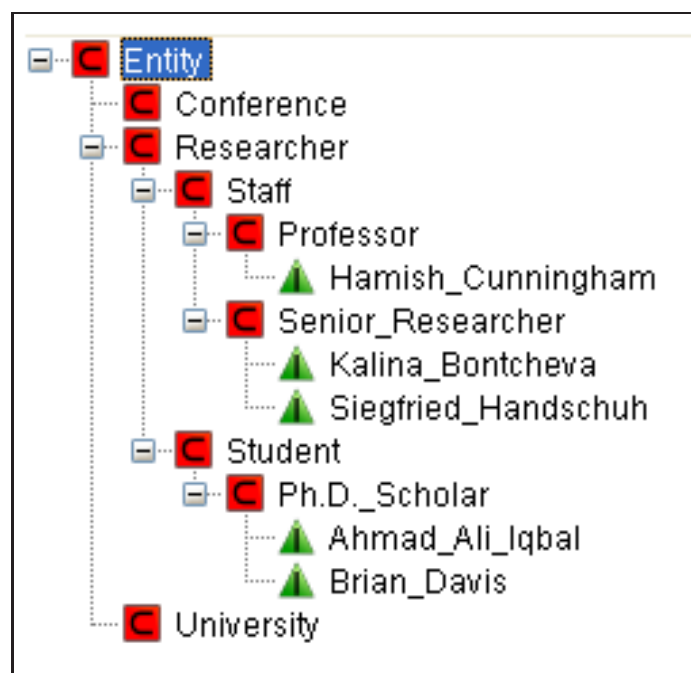


Figure C.1: Graphical depiction of classes and instances

instantiations of OWL/RDF(S) classes, instantiated properties from one class instance to a datatype instance (also called attribute instance) and instantiated properties from one class instance to another class instance, a relationship instance.

C.1.2.1 Terms:

- **Metadata:** Metadata is data about data. In our context the annotations are metadata about documents.
- **Relational:** Metadata: We use this term relational metadata to denote the annotations that contain relationship instances.

The term annotation can often denote a “private or shared note” or “comment”. This alternative meaning of annotation may be emulated in our approach by modelling these notes with attribute instances i.e. An attribute instance, of type ‘hasComment’.

C.1.3 Controlled Natural Language

As mentioned at the beginning of this evaluation, the goal of this evaluation is to measure the user friendliness of controlled natural languages as interface technologies for semantic annotation in comparison to standard annotation tools. As you are probably aware, human language does not lend itself to precise computer processing; it contains stylistic variations, ambiguities and other features that make it difficult for computers to interpret. One way to let people write instructions and data so that computers can handle the input correctly is to use an artificial language, such as a computer programming language.

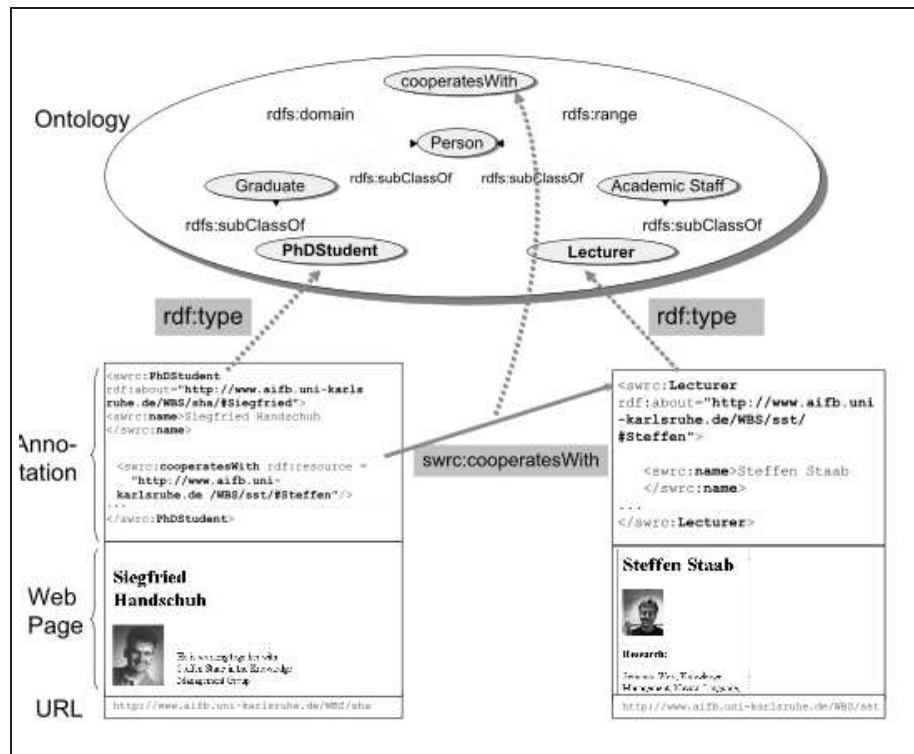


Figure C.2: Our use of the terms ontology, annotation and relational metadata

Another approach is to use a *controlled natural language*, which is a restricted subset of a natural language such as English. All the sentences in the controlled language are human-readable sentences in English and have a specific meaning for the computer program, but not all English sentences are valid in the controlled language. In two of the programs you will test today, you will type sentences in a controlled language and run a program that interprets them in order to link the text to and if needed create, classes, instances and properties in an ontology representing knowledge about a fictitious EU research project (a computerised record of information about team meetings concerning the project).

C.1.4 (CLANN) Type I and II - How-To

The facilitator will load either the CLANN I or CLANN II pipeline into GATE and load the initial textual data, so that you will have a window like the one shown in Figure C.3 to work with. You can click on the buttons or tabs across the top to bring up the following panes.

Messages This pane explains in detail what a CLANN annotator has just done and includes error messages. You can distinguish the error messages by the word *WARNING*, and probably ignore the *INFO* messages.

Text input In this pane (shown in Figure C.3) you will edit statements in the in either the CLANN I or CLANN II syntax, explained below. To clear any input, select all the text with the mouse and press the backspace key. We recommend editing individual parts. This can be achieved as you would edit text normally using the arrow and backspace keys and the mouse.

Annotation This pane (shown in Figure C.3 on the right hand side) shows you the semantic annotations created by either CLANN I or CLANN II. Before you begin the tasks, you might wish to look at the initial ontology to become familiar with it. See Appendix B for further details on how to inspect an ontology in GATE .

To execute an annotator on your current input text, right click either on the word *CLANN I* or *CLANN I* near the top of the left-hand pane and click *Run* in the menu that appears.

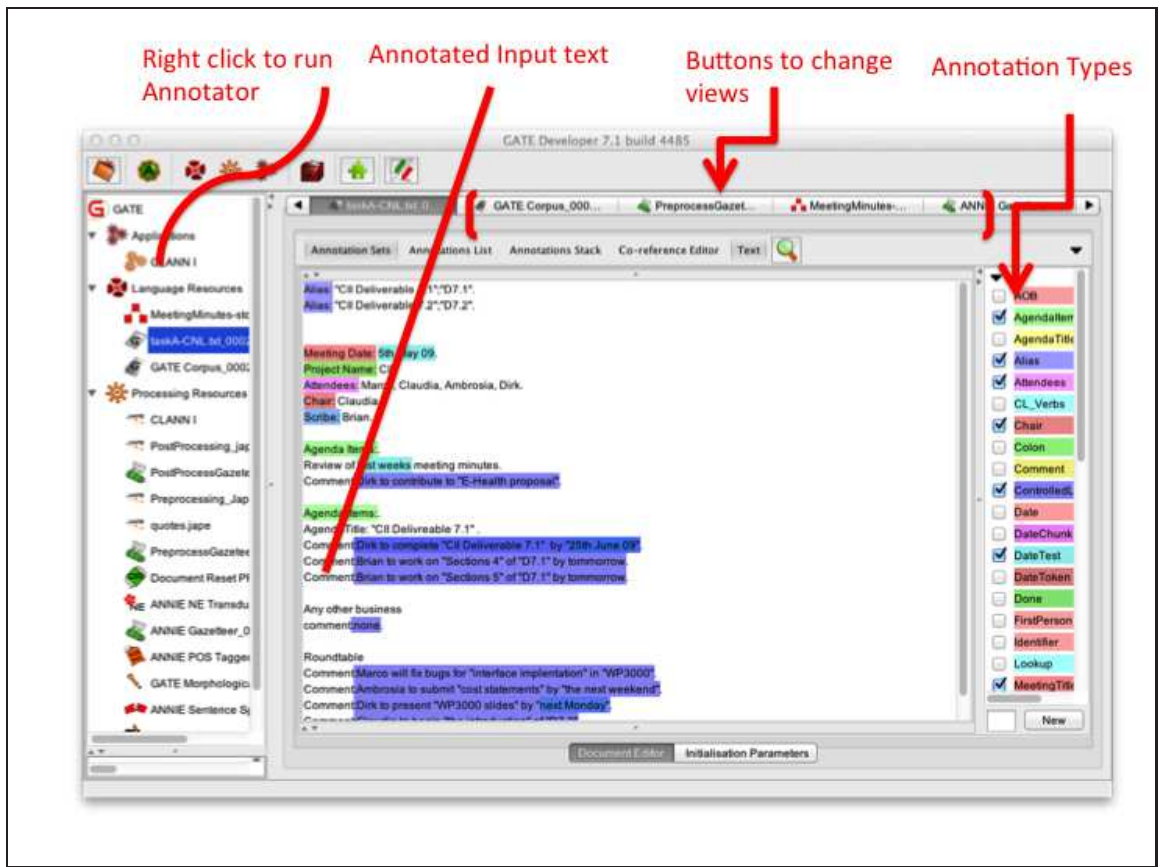


Figure C.3: Overview of CLANN I and II in GATE

C.1.5 Scenario

You are a member of a Research group in SAP Research Karlsruhe, Germany and have been asked to record the minutes of the latest team meeting. The other team members are: **Claudia, Dirk, Ambrosia and Marco**. The research project they are working on is called **CID**. Today you are the scribe and must record the meeting minutes for the CID project meeting with the other SAP team members using prototype tools for Controlled Natural Language ANNotator (CLANN), developed for the CID project.

C.1.6 Controlled Language ANNotator (CLANN) Type I

Two types of Controlled Natural Language ANNotator (CLANN) tools have been built for this evaluation. The following is guide for CLANN I:

Meeting Date: 5th May 09.

Project Name: CII.

Attendees: Marco, Claudia, Sven.

Chair: Claudia.

Scribe: Brian

Agenda Items:.

Review of last weeks meeting minutes.

Sven still needs to give input to Semantilix proposal

CII Deliverable

Marko to finally complete CII Deliverable 2.2 by 30th June 09

Claudia will start to write the introduction and the technical overview of D2.2

Sven to complete Sections 1 and 2 by end of the week.

Any other business

Roundtable

Marco will do bug fixing for middleware implementation in WP3000.

Sven to hand up a detailed report by the next weekend.

Figure C.4: Initial Sample meeting minutes in free or uncontrolled text

C.1.6.1 Instructions for writing in CLANN I:

1. Remove any adverbs or adjectives/modifiers from the input text:

Sven still needs to give input to Semantilix proposal

Exception: This is not necessary if the noun phrase is a multiword expression:

Sven to demonstrate a working prototype for the project review.

Why? Multiword Noun phrases must be surrounded by double or single quotes:

Sven to hand up ''a detailed report'' ''by the next weekend''.

2. Dates or Date expressions must be surrounded by single or double quotes:

''by the next weekend''.

''30th June 09''.

3. Persons or Single noun phrases do not need to be surrounded by double or single quotes:

Sven needs to give input to ''Semantilix proposal''

4. Verb Phrase (VP) expressions from the the key-phrase list (See C.8) can only be used. Rewrite the verb phrase if necessary. Hence:

Sven to hand up ''a detailed report'' ''by the next weekend''.

Changes to

Sven to submit ''a detailed report'' ''by the next weekend''.

5. Separate sentences with complex adjuncts containing conjunctions:

Claudia to write the introduction and the technical overview of D2.2.

Changes to:

Claudia to write the introduction of ''D2.2.''

Claudia to write the technical overview of ''D2.2''.

6. Make references to other nouns or entities explicit!

Sven to complete Sections 1 and 2 by end of the week.

Changes to:

Sven to complete ''Sections 1'' of D1.2 by ''end of the week''.

Sven to complete ''Sections 2'' of D1.2 by end of the week.

7. Again multiword units must be surrounded by single or double quotes.

8. Prepositions such as of and by can be left alone.

9. Nouns contains reserved phrases see Figure C.8 or punctuation must be surrounded by single or double quotes. The"." is reserved for end of sentence boundary only so D1.2 must be written as ''D1.2'' or 'D1.2'. Hence the final rewritten CNL form is:

Sven to complete ''Sections 1'' of ''D1.2'' by ''end of the week''.

Sven to complete ''Sections 1'' of ''D1.2'' by ''end of the week''.

10. All sentences must end with a ''.'

11. If you want a sentence to be considered for CNL parsing it must be preceded by the key-phrase `''Comment:''`:

```

Alias:<QuotedPhrase><QuotedPhrase>

Meeting Date: <Date> .

Project Name: <NAME> .

Attendees: (<Person>,+).

Chair: <Person>.

Agenda Items:

''Review of last weeks meeting minutes.''

(Comments: <CNL_Phrase>.)+ | ''comment none''.

Agenda Items.

AgendaTitle:<Title>.

(Comments: <CNL_Phrase>.)+ | ''comment none''.

AgendaTitle: any other business .

(Comments: <CNL_Phrase>.)+ | ''comment none''.

Roundtable.

(Comments: <CNL_Phrase>.)+ | ''comment none''.

| = disjunction, OR

(...) + = 1 or more

''...'' = Reserved Words in template.

```

Figure C.5: Meeting Minutes Template for CLANN I and II in BNF

Alias:CII Deliverable 5.2;D5.2,
T1500 Ipad;working prototype, '
'project review; 'EU Project Review Kaiserslautern 08 November 2009.
Meeting Date: 5th May 09.
Project Name: CII.
Attendees: Marco, Claudia, Sven.
Chair: Claudia.
Scribe: Brian
AgendaItems:.
Review of last weeks meeting minutes.
Sven needs to give input to 'Semantlix proposal'.
AgendaItems:.
AgendaTitle:CII Deliverable
Comment:Marco to complete 'CII Deliverable 5.2' by '30th June 00'.
Comment:Claudia to write the introduction of 'D5.2'.
Comment:Claudia to write the 'technical overview' of 'D5.2'.
Comment:Sven to complete 'Sections 1' of 'D5.2' by 'end of the week'.
Comment:Sven to complete 'Sections 2' of 'D5.2' by 'end of the week'.
Roundtable.
Comment:Marco to do bug fixing for 'middleware implementation' in WP3000.
Sven to complete a 'detailed report' of WP3000 by next weekend.
Sven to demonstrate a 'working prototype' for the 'project review'.

C.1.6.2 A note on Alias

Using the alias template, a user can describe synonyms or aliases for the same entity, hence it can act as a short hand for the user. Thus, if the user writes:

Alias: `''CII Deliverable 5.2'';D5.2.`

Any subsequent mention of D5.2 will tell the controlled natural language parser to substitute "D5.2" with "CII Deliverable 5.2". This is a form of predefined nominal coreference.

to complete	to give input to	will do
to submit	to contribute	working
to finish	to provide	to write
to provide	to provide technical	will write
needs to give input	input to	to write
still needs to give	to organise	to travel to
input	to organize	to commit
to give input	will work	to upload

Figure C.7: Sample of reserved words and phrases in CLANN I

Agenda Items:	comment:	Roundtable:
AgendaItems:	Comment:	Roundtable
AgendaItems	Comments:	roundtable:
AgendaTitle:	comments:	roundtable
Agenda Title	chair:	Scribe:
any other business	Chair	scribe:
Any other business	Date:	.
Attendees:	date:	''
attendees:	Project Name:	,

Figure C.8: Reserved template phrases and punctuation in CLANN 2

C.2 Controlled Language ANNotator Type II (CLANN II)

Two types of CNL annotators have been built for this evaluation. The following is guide for annotator Type II (called CLANN II):

C.2.1 Instructions for using CLANN II

Previously, you may have had to rewrite free or uncontrolled language into controlled language using the CLANN I Annotator. Type II is less "automatic" and gives the user more freedom to express his/her model, but also requires more manual effort. In CLANN II, the user inserts snippets of controlled language into free text. Snippets are of two varieties, the Predicate Snippets and Sentence Snippets. More, specifically, you insert a Predicate snippet of controlled language directly adjacent to the right of the object you wish to annotate. A sentence snippet can be inserts anywhere inside the text:

Predicate Snippet: <SUBJECT> [<VERB> <OBJECT>]

Eg: DERI [IS A INSTITUTE] enables networked knowledge.

Sentence Snippet: [<SUBJECT> <VERB> <OBJECT>]

Eg: [DERI "LOCATED IN" IRELAND]

You will need to inspect the ontology provided in the GATE GUI to decide on which instances and properties to link to. When you are assigned your task, the facilitator will assist you with access to the GATE Ontology Viewer. You can also create new classes, subclasses, properties and instances explicitly if desired.

C.2.1.1 Annotation Process

A simple step-by-step process of annotating a piece of text using Type II annotator is given below. An example sentence is provided below and annotated in the following manner:

Sven still has to write Fast project proposal.

1. Identify the relevant entities in the text and surround the instance with quotes(") in case it is a multi-word chunk.

Sven still has to write ‘‘Fast project proposal’’.

Sven, ‘‘Fast project proposal’’ are the relevant entities.

2. Annotate them with <INSTANCE>[IS A <CLASS>]. Use the class labels from the ontology provided. Create your own classes if you have to(See below).

Sven[is a Person] still has to write ‘‘Fast project proposal’’[is a Proposal].

Here Proposal is not an existing class, It is created on the fly.

3. Create your own classes if you have to. The class definition should be inside a sentence snippet and it should be defined before it is used.

[Proposal is a subclass of Document].

Sven[is a Person] still has to write ‘Fast project proposal’’

[is a Proposal].

Also use quotes for the class label when it is multi-word.

[‘‘Conference Paper’’ is a subclass of Document]

This would create a new subclass of Document.

```
[Animal is a Class].
```

This would create a new class as a subclass of `pimo:Thing`.

4. Create links between the instances that you just annotated. Links or properties linking two instances can be added as triples are independent sentence snippets.

```
[Proposal is a subclass of Document].
```

```
Sven[is a Person] still has to write ‘‘Fast project proposal’’
```

```
[is a Proposal].
```

```
[Sven toComplete ‘‘Fast project proposal’’].
```

5. Also create additional properties if you have to. This is similar to class creation.

```
[‘‘has to write’’ is a Property]
```

```
creates a blank property with domain pimo:Agent and range pimo:Thing [‘‘has  
to write’’ is a subproperty of toWrite]
```

```
creates a new property as a subproperty of ‘‘toWrite’’ thereby inheriting the same  
domain and range. So by adding an extra property our example would look like’’
```

```
[Proposal is a subclass of Document].
```

```
[‘‘has to write’’ is a subproperty of toWrite]
```

```
Sven[is a Person] still has to write ‘‘Fast project proposal’’
```

```
[is a Proposal].
```

```
[Sven ‘‘has to write’’ ‘‘Fast project proposal’’].
```

6. Aliases:: You have an option to use aliases for long multi-word chunks. An alias once defined could be used as a replacement. These could be used for Classes, Instances and properties”

```
[<ALIAS> SAME AS <TEXT-TO-BE-ALIASED>]
```

On adding an alias for ’’Fast project proposal’’:

```
[Proposal is a subclass of Document].
```

```
[‘has to write’ is a subproperty of toWrite]
```

```
Sven[is a Person] still has to write ‘Fast project proposal’
```

```
[is a Proposal].
```

```
[FPP is same as ‘Fast project proposal’]
```

```
[Sven ‘has to write’ FPP].
```

C.2.1.2 Other Rules

All sentences must end with a period ‘.’

Meeting Date: 5th May 09.

Project Name: CII

Attendees: Marko, Claudia, Leo .

Chair: Claudia.

Scribe: Brian

Agenda Items:.

Review of last weeks meeting minutes.

Sven still has to write Fast project proposal

CII Deliverable

Marko to finish working on CII Deliverable 6.7 by 25th May 09

Claudia will begin writing the introduction and

related work sections of D6.7.

Any other business

Leo to complete the conclusion of D6.7 by the end of the week.

Roundtable

Marco will commit the interface implementation for WP3000.

Leo to hand up a detailed Ph.D. Proposal by next weekend.

Leo will work on a conference paper for the

Information Visualisation 09 Conference, Istanbul.

Figure C.9: CLANN II - Initial Sample Text

Meeting Date: <Date> .

Project Name: <NAME> .

Attendees: (<Person>,+).

Chair: <Person>.

Agenda Items:

Review of last weeks meeting minutes.

(<CNL_Phrase>.)+

Agenda Items.

AgendaTitle:<Title>.

(<CNL_Phrase>.)+

AgendaTitle: any other business .

(<CNL_Phrase>.)+

Roundtable.

(<CNL_Phrase>.)+

| = disjunction, OR

(...)+ = 1 or more

... = Reserved Words in template.

Figure C.10: CLANN II - Meeting Minutes Template

Meeting Date: 5th May 09.

Project Name: CII.

Attendees: Marco, Claudia, Leo.

Chair: Claudia.

Scribe: Brian.

Agenda Items:.

Review of last weeks meeting minutes.

[Proposal is a subclass of Document].

[‘‘has to write’’ is a subproperty of toWrite] .

Sven[is a Person] still has to write ‘‘Fast project proposal’’[is a Proposal].

[FPP is same as ’’Fast project proposal’’] .

[Sven ‘‘has to write’’ FPP].

Agenda Items:.

AgendaTitle: CII Deliverable.

[Deliverable is a subclass of Document].

Marco[is a Person] to finish working on‘‘CII Deliverable 6.7’’

[is a Deliverable] by ‘‘25th May 09’’[is a Date].

[‘‘D6.7’’ is same as ‘‘CII Deliverable 6.7’’].

[Marco toComplete ’’D6.7’’] .

[‘‘D6.7’’ hasEndDate ’’"25th May 09’’].

Figure C.11: CLANN II - Correct Output

[Section is s subclass of Document] .
 [isPartOf is a Property].
 Claudia[is a Person] will begin writing ‘‘the introduction’’[is a Section]
 and ‘‘related work’’[is a Section] sections of D6.7.
 [Claudia toWrite ‘‘the introduction’’].
 [Claudia toWrite ‘‘related work’’].
 [‘‘the introduction’’ isPartOf ’’D6.7’’].
 [‘‘related work’’ isPartOf ’’D6.7’’].
 Leo[is a Person] to complete the ‘‘conclusion of D6.7[is a Section]
 by the end of the week[is a Date].
 [Leo toComplete ‘‘conclusion of D6.7’’].
 [‘‘conclusion of D6.7’’ isPartOf ’’D6.7’’].
 [‘‘conclusion of D6.7’’ hasEndDate ’’end of the week’’].
 Marko will finalise the ‘‘interface implementation’’
 [is same as ’’Interface T6700’’] for ‘‘WP3000’’[is a Workpackage].
 [Marco toComplete ‘‘Interface T6700’].
 [‘‘Interface T6700 isRelated ‘‘WP3000’’].
 [toSubmit is a Property].

Figure C.12: CLANN II - Correct Output Continued

```
[“to hand up” is a subproperty of toSubmit].  
Leo to hand up a detailed “Ph.D. Proposal”[is a Proposal] by  
  “next weekend”[is a Date].  
[Leo toSubmit “Ph.D. Proposal”].  
[“Ph.D. Proposal” hasEndDate “next weekend”]  
[isLocatedIn is a Property].  
Leo will work on a “conference paper”[is a Document] for  
  “the Information Visualisation 09 Conference”[is a Conference],  
  Istanbul[is a Location].  
[Leo toWork ’’conference paper’’].  
[“conference paper” isPartOf ’’the Information Visualisation 09 Conference’’].  
[“the Information Visualisation 09 Conference” isLocatedIn Istanbul].
```

Figure C.13: CLANN II - Final Correct Output

C.2.1.3 A note on authoring the ontology

In the above example the underlying Ontology should ideally be pre-populated with the appropriate instances, properties and classes. However, the Type II annotator permits simple ontology authoring in CNL. This must however occur prior to the use of the particular ontology elements. See Figure or further details concerning reserved template key-phrases:

```
[Deliverable is a subclass of Document].
```

```
[Section is a subclass of Deliverable].
```

[isPartOf is a Property]. Once the above statements are declared, you can freely use the elements defined.

```
''The introduction[is a Section] of ''CII Deliverable 6.7''[is a Deliverable].
```

```
[''the introduction'' isPartOf ''CII Deliverable 6.7''].
```

C.2.1.4 Use of Quotes

Quotes are to be used to define the word boundaries for multi-word phrases.

```
''25th December 2006''
```

```
''CII Deliverable 6.7''
```

```
''PhD Proposal''
```

They should also be used with alpha-numeric and any other tokens which contain punctuation or other symbols within the text i.e.

WP3000

D6.7

C.2.1.5 Relation Metadata.

Relation metadata or triples are automatically parsed and generated based on the manual annotations provided. You do not need to be concerned about this.

C.2.2 CLANN II gazetteer (keywords and phrases)

This table lists all the phrases annotated by the CLANN II gazetteer, as described by in Section 6.2.3.2. The gazetteer marks each listed phrase with a `Lookup` annotation with the specified `majorType` feature; some phrases' annotations also have a `minorType` feature.

<code>majorType</code>	<code>minorType</code>	gazetteer entry
CLANN-II-Instance		is a
CLANN-II-Instance		is a type of
CLANN-II-New-Class		is a Class
CLANN-II-New-Class		is a class
CLANN-II-New-SubClass		is a part of
CLANN-II-property		is a property
CLANN-II-property		is a relation
CLANN-II-property		is a property between

majorType minorType	gazetteer entry
CLANN-II-property	is a relation between
CLANN-II-property	is a relationship
CLANN-II-property	is a relationship between
CLANN-II-property	property
CLANN-II-property	relationship
CLANN-II-property	relation
CLANN-II-property	is same property as
CLANN-II-property	is same relation as
CLANN-II-property	is same relationship as
CLANN-II-synonym	is kinda like
CLANN-II-synonym	is the same as
CLANN-II-synonym	is sort of like
CLANN-II-synonym	is equivalent to
CLANN-II-synonym	is same as
CLANN-II-synonym	same as
CLANN-II-synonym	kinda like
CLANN-II-synonym	sort of like
CLANN-II-synonym	sort of like
CLANN-II-synonym	sorta like
CLANN-II-numeric attribute	is a number

majorType	minorType	gazetteer entry
CLANN-II-numeric	attribute	is a Number
CLANN-II-string	attribute	is a String
CLANN-II-string	attribute	is a string

Agenda Items:	comment:	Roundtable:
AgendaItems:	Comment:	Roundtable
AgendaItems	Comments:	roundtable:
AgendaTitle:	comments:	roundtable
Agenda Title	chair:	Scribe:
any other business	Chair	scribe:
Any other business	Date:	.
Attendees:	date:	''
attendees:	Project Name:	,

Figure C.14: Reserved template phrases and punctuation in CLANN 2

C.3 OntoMat Annotatizer

C.3.1 Overview

OntoMat-Annotizer is a user-friendly interactive webpage annotation tool. It supports the user with the task of creating and maintaining ontology-based OWL-markups i.e. creating of OWL-instances, attributes and relationships. It includes an ontology browser for the exploration of the ontology and instances and a HTML browser that will display the annotated parts of the text. The intended user is the individual annotator i.e., people that want to enrich their web pages with OWL-meta data. Instead of manually annotating the page with a text editor, say, emacs, OntoMat allows the annotator to highlight relevant parts of the web page and create new instances via drag n drop interactions. It supports the meta-data creation phase of the lifecycle. Ontomat is considered a de-facto standard semantic annotation tool. It is an example of a semi-automatic/manual annotation tool. Below is a quick start guide to the tool in addition to an example text containing meeting minutes and a further details of how to semantically annotate the meeting minutes note using the OntoMat-Annotatizer. The contents of this section are based on the Ontomat-Annotizer Web Tutorial Guide¹³³.

¹³³See <http://annotation.semanticweb.org/ontomat/tutorial.html> Extracted Sunday 29 Sep 2013 20:56:20 IST

C.3.1.1 Introduction to Ontomat

Step 1: Open Ontology Browser:

Open the Ontology-Browser, if not already open.

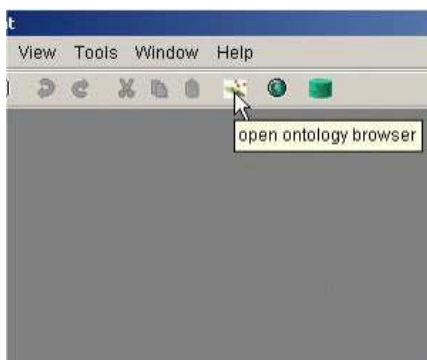


Figure C.15: Opening Ontology Browser in Ontomat

Step 2: Open Web Browser:

Open the Web-Browser if not already open.

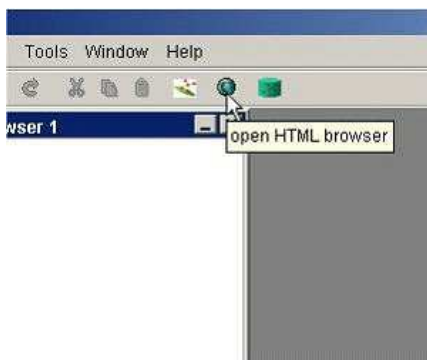


Figure C.16: Opening Web Browser in Ontomat

Step 3: Load HTML Document:

Load an HTML document by entering the URL of the web document that would you like to annotate. OntoMat-Annotizer will use the URL of your document as the namespace of your annotation. If you don't like that, you can change the namespace with "Tools - HTML Browser- change document namespace". You can check the namespace of your document on the bottom of your browser pane (See Figure C.3.1.1).

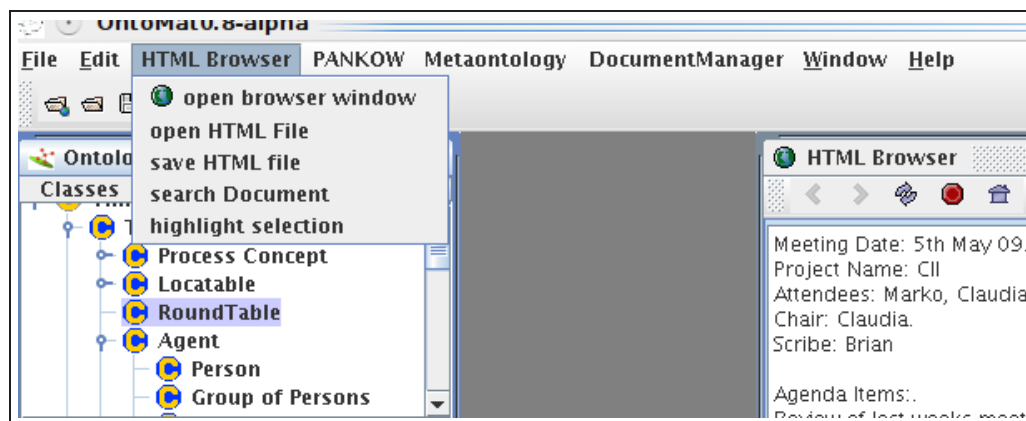


Figure C.17: Opening a HTML file in Ontomat

Step 4: Opening OWL Ontology in Ontomat:

Browse throughout the ontology to get familiar with it.



Figure C.18: Opening an OWL Ontology in Ontomat

Step 5: Select Text:

Select the text fragment which you like to use for your annotation.

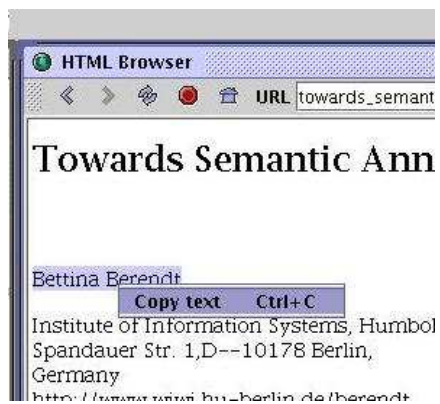


Figure C.19: Selecting text in Ontomat

Step 6: Create Instance

Select in the ontology the class where the text fragment fits in. Drag the selected text on the appropriate class (drag'n drop). The annotation gets created and thus the text fragment will be shown as an instance of the selected class in the ontology in the ontology browser.



Figure C.20: Creating an instance in Ontomat

Step 7: Create Attributes

To each created instance, literal attributes can be assigned. The choice of the predefined attributes depends on the class the instance belongs to, e.g. the class “Meeting” has the attributes duration, topic, and location. The attributes can be assigned to the instance by selecting the appropriate text fragment of the web document and copy it to the related property field (drag’n drop).

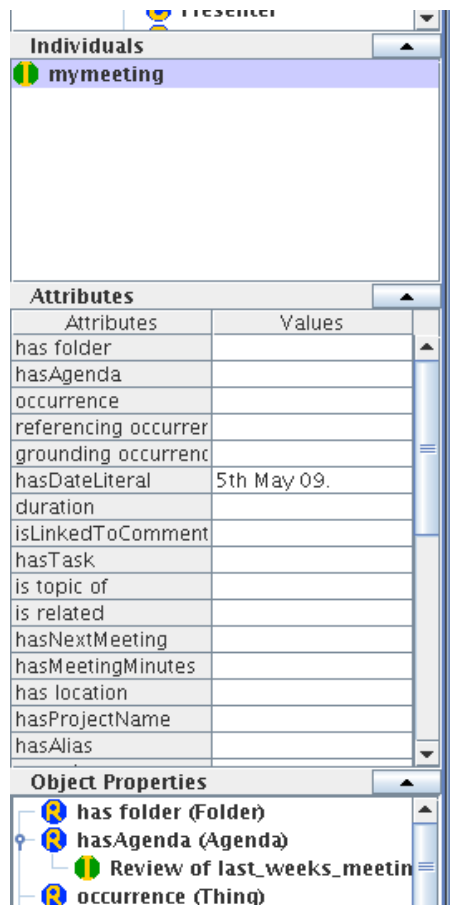


Figure C.21: Creating Attributes in Ontomat

Step 8: Create Relations

Furthermore, the relationships between the created instances can be set, e.g. the Researcher “Sven” has to write a proposal for the “Fast Project”. OntoMat-Annotizer preselects class instances according to the range restrictions of the chosen relation, e.g. the “toWrite” of a Person must be a Document. Therefore only these are offered as

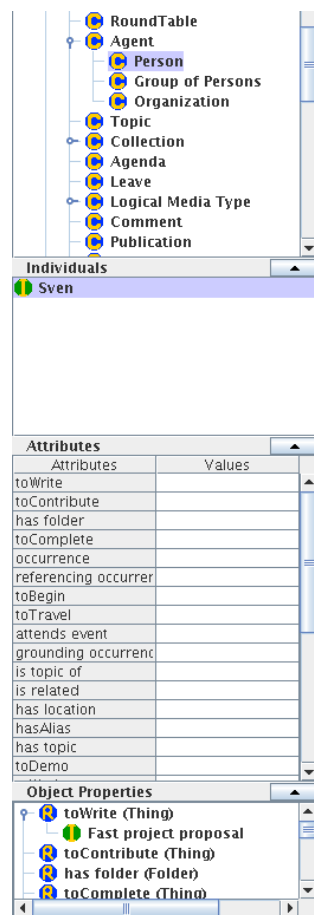


Figure C.22: Creating Relations in Ontomat

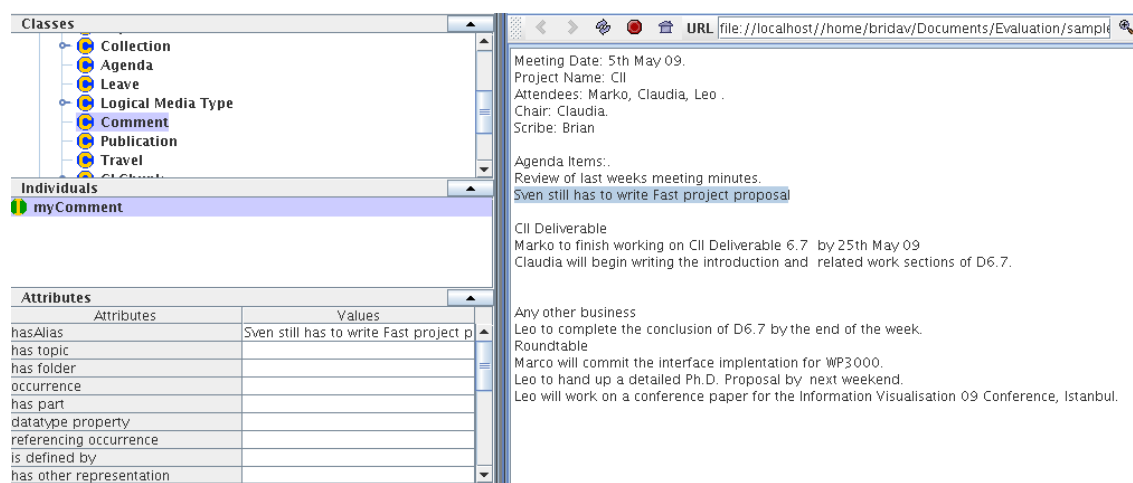
potential fillers to the ‘toWrite’ relation. Choose the appropriate instance and drag it

to the relation (drag'n drop). Due to implementation reasons, this is not consistent with right mouse click for the other steps!

Step 9: Control and Edit your Annotation manually

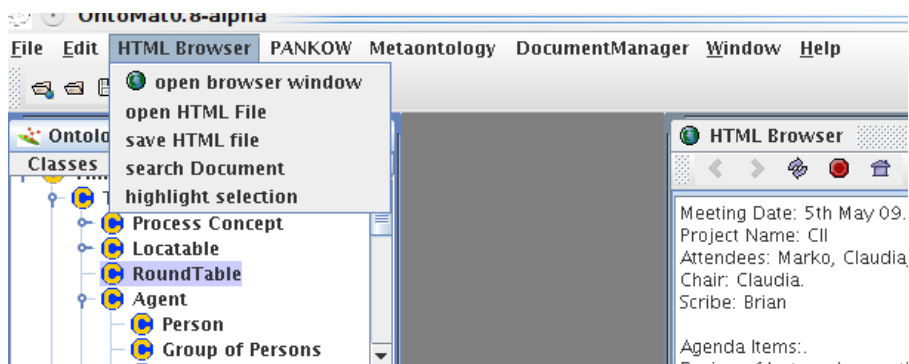
With the annotation tab you can switch to the textual representation of your annotation.

You can do here some changes manually, but take care of consistency.



Step 11: Save the HTML file and FINISH!

Finally save your annotated HTML table page.



Meeting Date: 5th May 09.

Project Name: CII

Attendees: Marco, Claudia, Leo .

Chair: Claudia.

Scribe: Brian

Agenda Items:.

Review of last weeks meeting minutes.

Sven still has to write Fast project proposal

CII Deliverable

Marco to finish working on CII Deliverable 6.7 by 25th May 09

Claudia will begin writing the introduction and related work sections of D6.7.

Marco will commit the interface implementation for WP3000.

Leo to hand up a detailed Ph.D. Proposal by next weekend.

Leo will work on a conference paper for the Information

Visualisation 09 Conference, Istanbul.

Figure C.23: Example Meeting Minutes for OntoMat

C.3.2 Structural Annotation

In the case of Ontomat, it will be necessary to make explicit links and provide structural annotations for structured elements in the note such as ‘ ‘ Meeting Title’ ’ “Attendees” etc. Examples are provided below.

Step 1: Create a new instance of a Meeting in Ontomat.

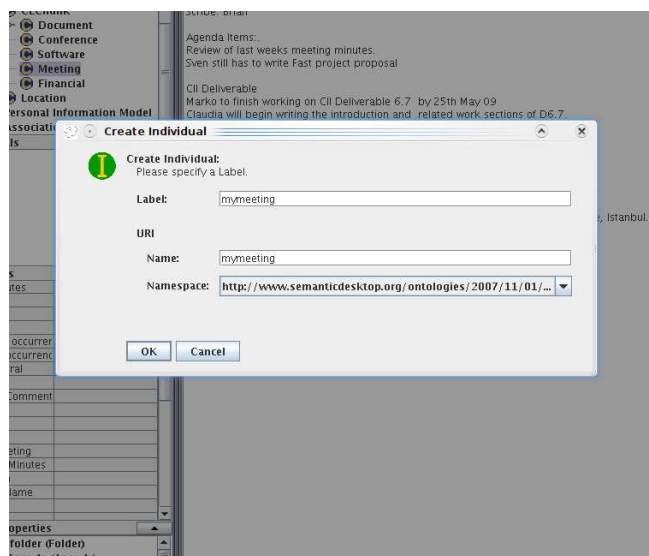
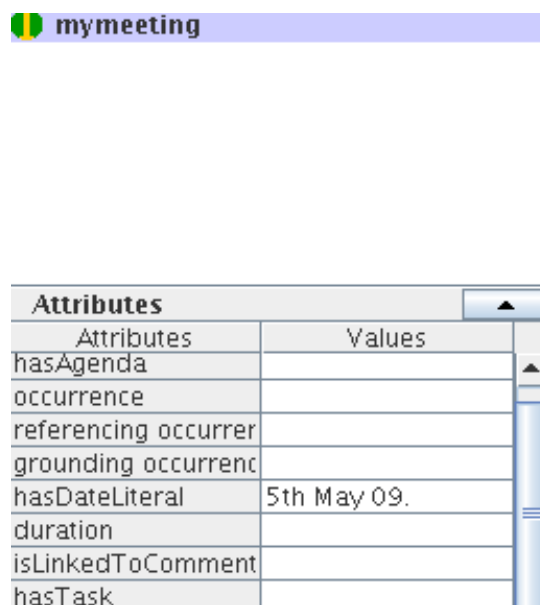


Figure C.24: Creating a new instance of a Meeting in Ontomat.

Step 2: Annotate the Date of the Meeting in the text using “hasDateLiteral”.




The screenshot shows a software interface with a header bar labeled "mymeeting" and a table titled "Attributes". The table has two columns: "Attributes" and "Values". The "hasDateLiteral" attribute is highlighted, and its value is "5th May 09.".

Attributes	Values
hasAgenda	
occurrence	
referencing occurrence	
grounding occurrence	
hasDateLiteral	5th May 09.
duration	
isLinkedToComment	
hasTask	

Figure C.25: Annotating the Date of the Meeting in the text using “hasDateLiteral”.

Step 3: Annotate the Project name of the Meeting in the text using the same approach.

Individuals	
 mymeeting	

Attributes	
Attributes	Values
hasAgenda	
occurrence	
referencing occurrence	
grounding occurrence	
hasDateLiteral	5th May 09.
duration	
isLinkedToComment	
hasTask	
is topic of	
is related	
hasNextMeeting	
hasMeetingMinutes	
has location	
hasProjectName	CII
hasAlias	

Figure C.26: Annotate the Project name of the Meeting in Ontomat

Step 4: Create Person instances for each person in the meeting minute note.

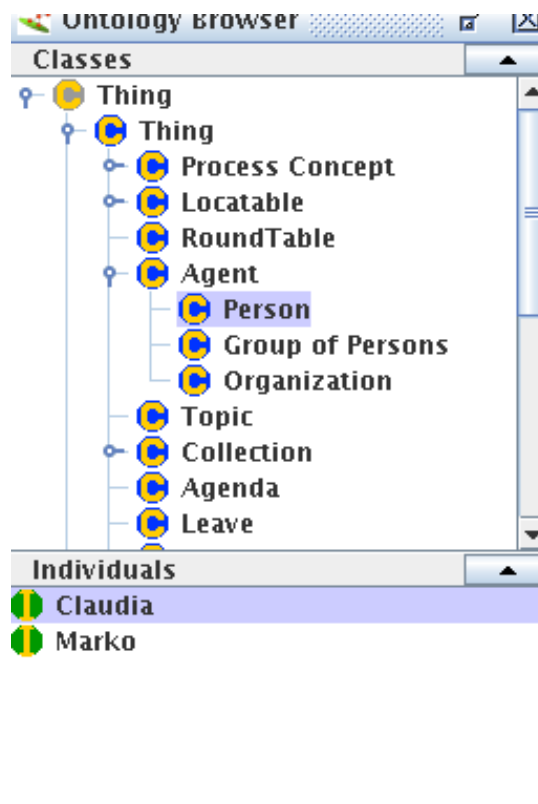


Figure C.27: Create Person instances for each person a note

Step 5: Create and instance of Meeting Chair.

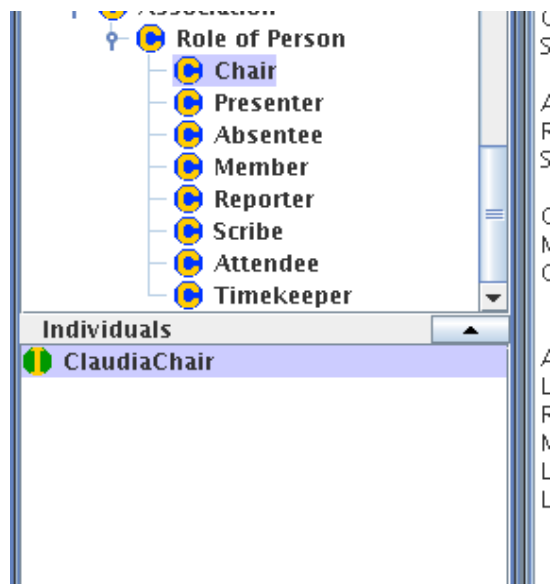


Figure C.28: Create and instance of Meeting Chair

Step 6: Assign the Role of Chair to a Person via the relationship “role holder(Person)”.

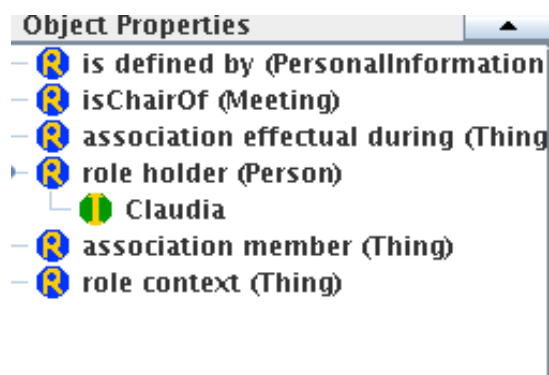


Figure C.29: Assign the Role of Chair to a Person

Step 7: Assign Claudia (ClaudiaChair) as Chair to the meeting instance using the object property hasChair(Chair).

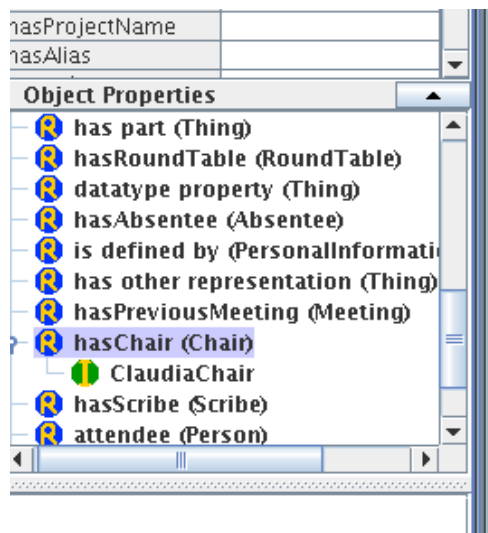


Figure C.30: Assign Claudia (ClaudiaChair) as Chair

Step 8: Repeat the Steps above for the Scribe in the meeting minutes note.

Step 9: Create an Agenda Item from "Review of last weeks meeting minutes."

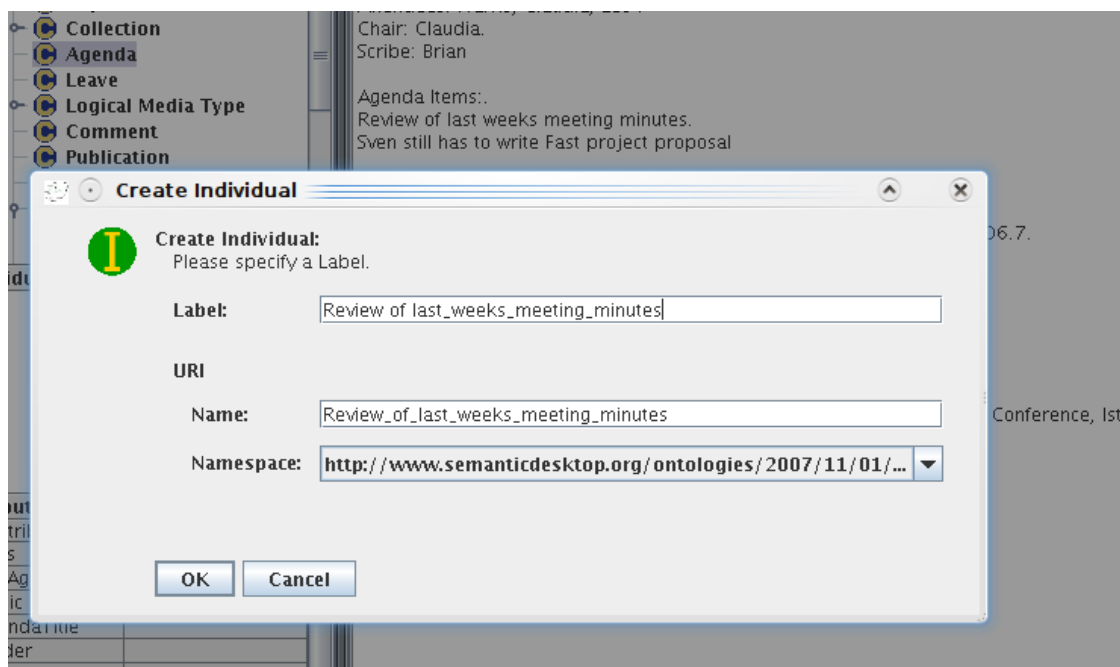


Figure C.31: Create an Agenda Item in Ontomat

Step 10: Link the Agenda Item to the current Meeting instance in the Ontology using the “hasAgenda(Agenda)” property.

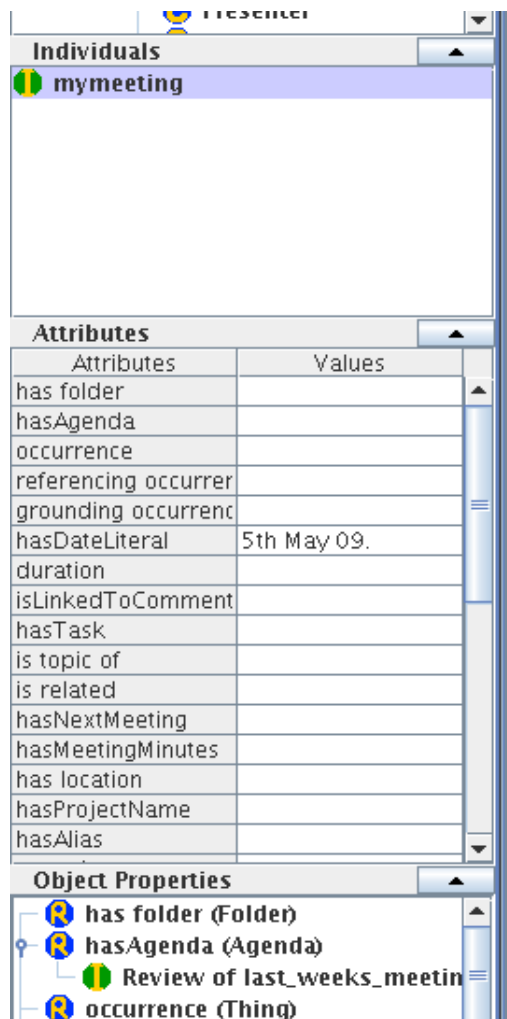


Figure C.32: Link the Agenda Item to the current Meeting instance

Step 11: Create relation Metadata for the sentence “Sven still has to write Fast Project Proposal” with object property “toWrite(Document)” where “Fast Project Proposal” is an instance of Document and Sven is an instance of Person.

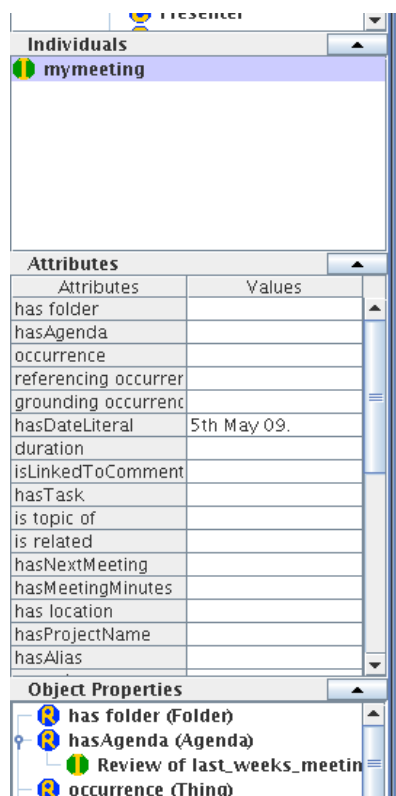


Figure C.33: Create relation Metadata

Step 12: We also want to link this relation to the Agenda item “Review ...”. To achieve this we use the object in “Sven still has to write Fast Proposal” , which is ”Fast project Proposal” of type Document to link to the Agenda Item Review

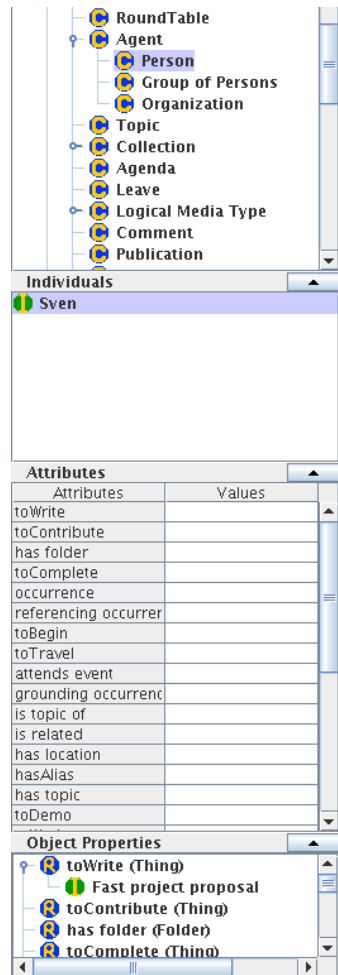


Figure C.34: Linking to the Agenda Item

Step 13: Create an instances of type Comment and annotate the Sentence “Sven still has to write Fast Proposal” with the attribute “hasAlias”. You can also create an Agenda Item in the same manner above for “CII Deliverable” and “Any other business”. Create ontological instances for the text and link them again appropriately as well creating a link to the Agenda item in question as before. Finally you can create a “Roundtable instance similar to Agenda like so:

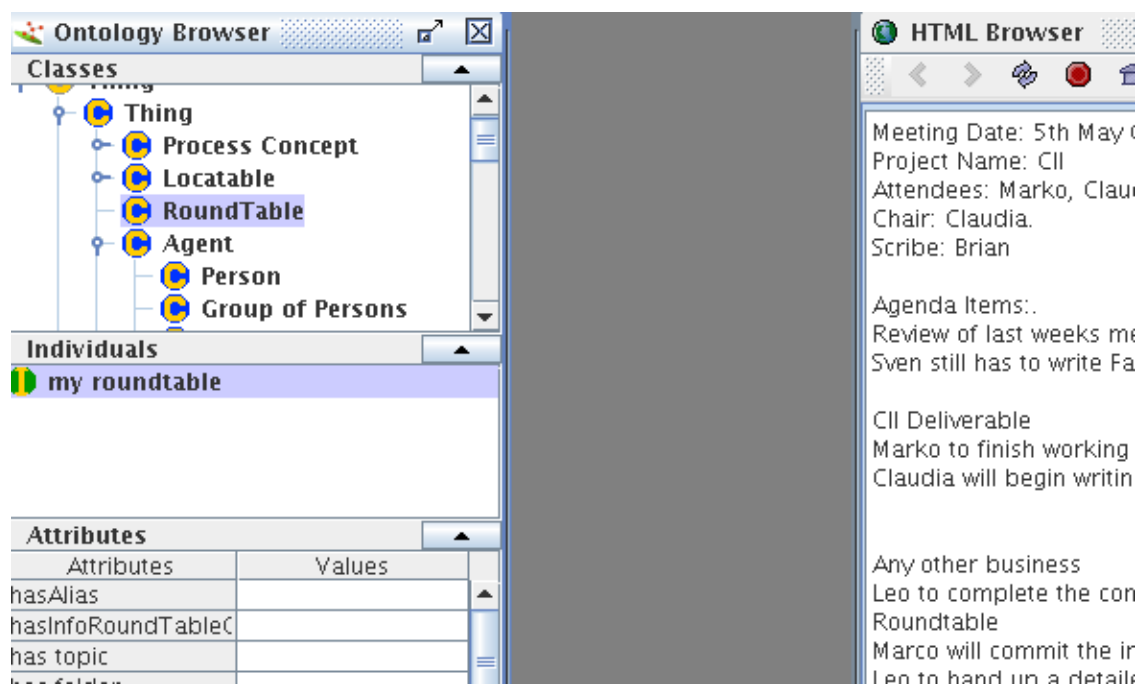


Figure C.35: Create an instances of type Comment

Step 14: You may save the Ontology afterwards if you wish like so:

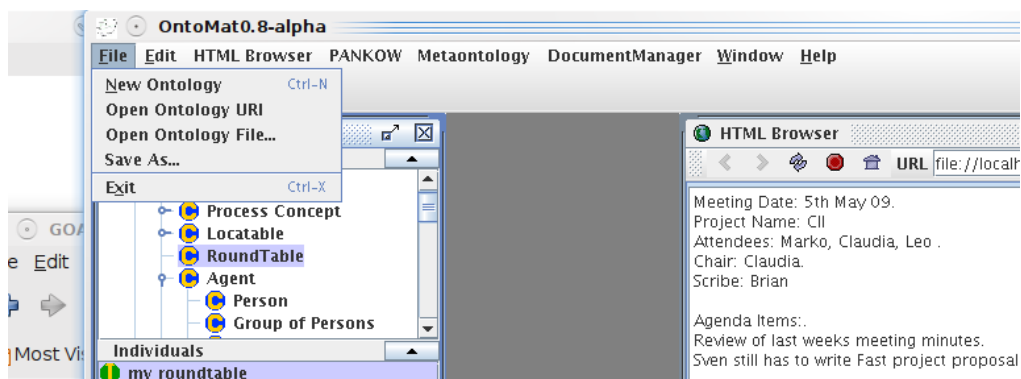


Figure C.36: Saving the Ontology in Ontomat

Step 15: Finally, there is the possibility to export your metadata to the document in the HTML browser like so:

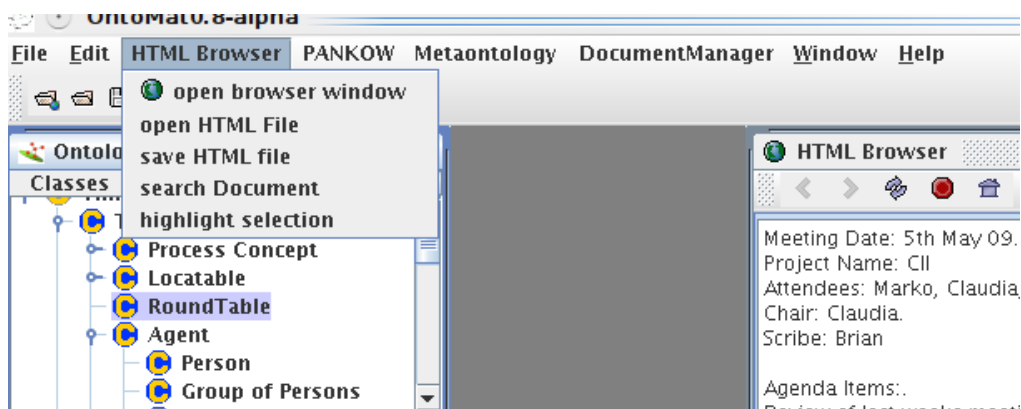


Figure C.37: Exporting Metadata in Ontomat

C.4 Test procedure, tasks and questionnaires

First we asked each subject to complete the pre-test questionnaire shown in Figure C.41. We then gave each subject one tool (CLANN I, CLANN II or Ontomat, respectively). They were asked carry out the tasks shown below, i.e., Task A (See Figure C.38) with the first tool, Task B (See Figure C.39) with the second tool and Task C (See Figure C.40) with the third tool. The task and tool order were varied equally across all participants.

We asked the subject to complete the questionnaire shown in Figure C.42 and then separately the questionnaires in Figures C.43, C.44 C.45 and C.46, depending on the tool used.

Meeting Date: 5th May 09.

Project Name: CII.

Attendees: Marko, Claudia, Ambrosia, Dirk.

Chair: Claudia.

Scribe: Brian

Agenda Items:.

Review of last weeks meeting minutes.

Dirk still has to provide a contribution to E-Health proposal

CII Deliverable 7.1

Dirk to finally complete CII Deliverable 7.1 by 25th June 09

<You> to work on Sections 4 and 5 tomorrow.

Any other business

No other business

Roundtable

Marko will fix bugs for interface implementation in WP3000.

Ambrosia to hand up a cost statements by the next weekend.

Dirk to present WP3000 slides next Monday.

Claudia to begin the introduction and the literature review of D7.2

Figure C.38: Task A

Meeting Date: 5th May 09.

Project Name: CII.

Attendees: Marko, Claudia, Ambrosia, Dirk.

Chair: Claudia.

Scribe: <You>

Agenda Items:.

Review of last weeks meeting minutes.

Marko to start working on Ph.D proposal

CII Deliverable 7.4

Dirk to work on Section 5.

EU Cost Statements

Ambrosia to work on Cost statement section of May CII Financial Report .

Any other business

Marko submit subversion code for interface implementation of CII middleware.

Roundtable

Ambrosia to hand up a tax forms by next weekend.

Dirk to start writing conference paper for International Semantic Web

Technologies Conference 2009 in Crete.

<YOU> to write the meeting minutes to Wiki.

Figure C.39: Task B

Meeting Date: 5th May 09.

Project Name: CII.

Attendees: Marko, Claudia, Ambrosia, Dirk.

Chair: Claudia.

Scribe: Brian.

Agenda Items:.

Review of last weeks meeting minutes.

Dirk to finish work on thesis plan.

CII Deliverable 6.7B

Marko worked on CII D6.7B

Claudia to travel to CII meeting in Brussels on Monday 25th May 09.

Any other business

No

Roundtable

Dirk to start writing conference paper for International Semantic Web

Technologies Conference 2009 in Crete.

<YOU> writing a Masters thesis chapter on a discovery architecture for portable,

linked photograph annotation metadata.

Ambrosia to upload EU FP7 slides to Wiki.

Figure C.40: Task C

- | | | | | |
|---|--|--------------------------------|------------------------------------|--------------------------------|
| 1 | I understand the term “Semantic Annotation”. | No <input type="checkbox"/> | A little <input type="checkbox"/> | Yes <input type="checkbox"/> |
| 2 | I am familiar with Semantic Annotation Tools. | No <input type="checkbox"/> | A little <input type="checkbox"/> | Yes <input type="checkbox"/> |
| 3 | I have worked with Ontologies. | Never <input type="checkbox"/> | Sometimes <input type="checkbox"/> | Often <input type="checkbox"/> |
| 4 | I have built or designed a Semantic Annotator. | Never <input type="checkbox"/> | Sometimes <input type="checkbox"/> | Often <input type="checkbox"/> |
| 5 | I understand the term “controlled language”. | No <input type="checkbox"/> | A little <input type="checkbox"/> | Yes <input type="checkbox"/> |
| 6 | I have used a controlled language. | Never <input type="checkbox"/> | Sometimes <input type="checkbox"/> | Often <input type="checkbox"/> |

Figure C.41: Pre-test questionnaire

		Strongly	Disagree	Neutral	Agree	Strongly
		disagree				agree
1	I think that I would like to use this system frequently.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	I found the system unnecessarily complex	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	I think that I would need the support of a technical person to be able to use this system.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	I found the various functionalities in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7	I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8	I found the system very cumbersome to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9	I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
10	I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure C.42: Post-test questionnaire for each Tool

		CLANN1			CLANN1			Ontomat			Ontomat		
1	I found one tool's documentation easier to understand.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
2	I particularly disliked using one tool.	disliked	<input type="checkbox"/>	disliked	<input type="checkbox"/>	neutral	<input type="checkbox"/>	disliked	<input type="checkbox"/>	disliked	<input type="checkbox"/>		
		strongly								strongly			
3	I found one tool easier to use.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
4	One annotator was harder to learn.	much	<input type="checkbox"/>	harder	<input type="checkbox"/>	neutral	<input type="checkbox"/>	harder	<input type="checkbox"/>	much	<input type="checkbox"/>		
		harder								harder			
5	I would prefer to use one tool again.	strongly	<input type="checkbox"/>	prefer	<input type="checkbox"/>	neutral	<input type="checkbox"/>	prefer	<input type="checkbox"/>	strongly	<input type="checkbox"/>		
		prefer								prefer			
6	I found one tool more complicated.	much more	<input type="checkbox"/>	more	<input type="checkbox"/>	neutral	<input type="checkbox"/>	more	<input type="checkbox"/>	much more	<input type="checkbox"/>		
		complex		complex				complex		complex			
7	I found it easier to control classes and subclasses in one tool.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
8	Properties were easier to link with one annotator.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>		
9	It was difficult to link instances in one annotator.	much	<input type="checkbox"/>	harder	<input type="checkbox"/>	neutral	<input type="checkbox"/>	harder	<input type="checkbox"/>	much	<input type="checkbox"/>		
		harder								harder			
10	It was awkward to link(or annotate) properties in one annotator.	very	<input type="checkbox"/>	awkward	<input type="checkbox"/>	neutral	<input type="checkbox"/>	awkward	<input type="checkbox"/>	very	<input type="checkbox"/>		
		awkward								awkward			

Figure C.43: Post-test questionnaire comparing the tools

		CLANN2					Ontomat		Ontomat		
1	I found one tool's documentation easier to understand.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>
2	I particularly disliked using one annotator.	disliked	<input type="checkbox"/>	disliked	<input type="checkbox"/>	neutral	<input type="checkbox"/>	disliked	<input type="checkbox"/>	disliked	<input type="checkbox"/>
		strongly								strongly	
3	I found one tool easier to use.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>
4	One tool was harder to learn.	much	<input type="checkbox"/>	harder	<input type="checkbox"/>	neutral	<input type="checkbox"/>	harder	<input type="checkbox"/>	much	<input type="checkbox"/>
		harder								harder	
5	I would prefer to use one tool again.	strongly	<input type="checkbox"/>	prefer	<input type="checkbox"/>	neutral	<input type="checkbox"/>	prefer	<input type="checkbox"/>	strongly	<input type="checkbox"/>
		prefer								prefer	
6	I found one tool more complicated.	much more	<input type="checkbox"/>	more	<input type="checkbox"/>	neutral	<input type="checkbox"/>	more	<input type="checkbox"/>	much more	<input type="checkbox"/>
		complex		complex				complex		complex	
7	I found it easier to control classes and subclasses in one tool.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>
8	Properties were easier to link with one annotator.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>
9	It was difficult to link instances in one annotator.	much	<input type="checkbox"/>	harder	<input type="checkbox"/>	neutral	<input type="checkbox"/>	harder	<input type="checkbox"/>	much	<input type="checkbox"/>
		harder								harder	
10	It was awkward to link (or annotate) properties in one annotator.	very	<input type="checkbox"/>	awkward	<input type="checkbox"/>	neutral	<input type="checkbox"/>	awkward	<input type="checkbox"/>	very	<input type="checkbox"/>
		awkward								awkward	

Figure C.44: Post-test questionnaire comparing the tools

		CLANN1					CLANN2		CLANN2		
1	I found one tool's documentation easier to understand.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>
2	I particularly disliked using one .	disliked	<input type="checkbox"/>	disliked	<input type="checkbox"/>	neutral	<input type="checkbox"/>	disliked	<input type="checkbox"/>	disliked	<input type="checkbox"/>
		strongly								strongly	
3	I found one tool easier to use.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>
4	One tool was harder to learn.	much	<input type="checkbox"/>	harder	<input type="checkbox"/>	neutral	<input type="checkbox"/>	harder	<input type="checkbox"/>	much	<input type="checkbox"/>
		harder								harder	
5	I would prefer to use one tool again.	strongly	<input type="checkbox"/>	prefer	<input type="checkbox"/>	neutral	<input type="checkbox"/>	prefer	<input type="checkbox"/>	strongly	<input type="checkbox"/>
		prefer								prefer	
6	I found one tool more complicated.	much more	<input type="checkbox"/>	more	<input type="checkbox"/>	neutral	<input type="checkbox"/>	more	<input type="checkbox"/>	much more	<input type="checkbox"/>
		complex		complex				complex		complex	
7	I found it easier to control classes and subclasses in one tool.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>
8	Properties were easier to link with one tool.	much easier	<input type="checkbox"/>	easier	<input type="checkbox"/>	neutral	<input type="checkbox"/>	easier	<input type="checkbox"/>	much easier	<input type="checkbox"/>
9	It was difficult to link instances in one tool.	much	<input type="checkbox"/>	harder	<input type="checkbox"/>	neutral	<input type="checkbox"/>	harder	<input type="checkbox"/>	much	<input type="checkbox"/>
		harder								harder	
10	It was awkward to link (or annotate) properties in one tool.	very	<input type="checkbox"/>	awkward	<input type="checkbox"/>	neutral	<input type="checkbox"/>	awkward	<input type="checkbox"/>	very	<input type="checkbox"/>
		awkward								awkward	

Figure C.45: Post-test questionnaire comparing CLANN I and CLANN II

Do you have any comments on any of the systems?

Do you have any specific problems to report?

Do you have any suggestions for improving either CLANN I and CLANN II?

Figure C.46: Post-test questionnaire comparing the tools