Abstract

In this project we investigate and develop the LaROS application that takes a text input and uses syntactic parsing tools to produce a knowledge base of linguistic entities. We expand the knowledge base further using references to our own OWL ontological framework and SWRL logic rules. Importantly our outputs are part of the Semantic Web and Linked Data domain. We further expand the knowledge base to include external data sources such as DBPedia thereby add world knowledge to the discovery and categorisation of rhetorical patterns including figures of speech and Rhetorical Structure Theory relations. We conclude that this research contributes to the domain by automatically creating a knowledge base that's published and re-usable with Semantic Web tools, but also adds Meaning Representations using syntactic parsing, Cue Phrase analysis and external world knowledge augmentation.

Table of Contents

Table of Figures

Chapter 1 – Introduction

This report describes the MSc project undertaken by Cliff O'Reilly during the summer of 2010. In the forthcoming chapters we describe the analysis and development of a computer system for the discovery of rhetorical patterns within natural language. We have used the term Lassoing rather than a more standard word like "Discovering" firstly because we find it more interesting to have a slightly different or incongruous project title, but secondly because it highlights a rhetorical form in the title of a project about finding rhetorical forms. The figure of speech of Catachresis is the use of "*an extravagant, unexpected, or farfetched metaphor*" [40] and the use of the word Lassoing fits this description. It is strange to imagine a cowboy with a rope grappling with strings of words, but logically correct and we hope quite memorable.

The computer program we build in this project we call LaROS (<u>La</u>ssoing <u>R</u>hetoric with <u>O</u>WL and <u>S</u>WRL).

The report is divided into chapters that variously describe the background to the project – the Domain, the Problem and Related Works – and the solution, implementation and conclusion of the results of work undertaken on the project.

The purpose of the project is two-fold: to develop a working prototype that can find successfully various language patterns by using Semantic Web tools, but also to develop a working knowledge base for each text input that will be augmented dynamically with information from the Semantic Web. The benefits of having a tool with visible outputs will hopefully be supplemented by the creation of Meaning Representations of the language. This research is an initial step towards enabling Semantic Analysis of language and hence enabling computers to "understand" natural language.

The rest of this chapter describes the Domain and the Problem we hope to address. Chapter 2 describes the project proposal in detail. Chapter 3 is a description of a number of related works that give some background to the issues and an appraisal of some significant research in a similar domain. Chapter 4 is a thorough description of the architecture of LaROS – the solution we put together to address the problems shown previously. In Chapter 5 we refer to the specific implementation details of the solution and Chapter 6 shows our evaluation of the work. Finally Chapter 7 contains our conclusion and a description of future work that we think would benefit the research problem.

1.1 The Domain

This project involves a number of domains including Natural Language Parsing, the Semantic Web, and forms of natural language structure known as Figures of Speech which go to form Rhetorical Devices.

Parsing is a technique from Linguistics that *"means taking an input and producing some sort of linguistic structure for it"* [33]. There are various forms of Parsing of text including Morphological, Syntactic, Semantic and Discourse [33]. For this project we will use Syntactic Parsing.

The Semantic Web is a movement driven by the World Wide Web Consortium to add knowledge and information to the "web of data" [71] via Ontologies (in practice usually Knowledge Bases) and logic reasoning. By creating standards, for example Web Ontology Language (OWL) and Semantic Web Reasoning Language (SWRL), resilient links can be developed between previously disparate data relatively easily. Although still in its infancy its development will significantly change the internet and computing in general by encoding concepts and relationships within domains and allowing computers to "understand" the world. This technique of "encoding a domain" can be applied to any specification, albeit in abstraction, including natural language.

Rhetoric has many definitions and interpretations – unsurprisingly for an art that is thousands of years old. Since Classical times the structure of natural language has been analysed and documented with respect to forms of communication intended to persuade, teach, enlighten, cajole, move or amuse. To facilitate the impact of rhetoric Figures of Speech are often employed to enhance or guide meaning. Recently there has been renewed interest in the art of rhetoric due to the popularity of the United States President Obama's notable oratory and use of rhetoric [46]. There are hundreds of documented possible rhetorical patterns of words; sometimes known as Figures of Speech. Many of these have been described since antiquity [40]. Rhetorical patterns vary from simple repetitions of words or phrases to document-level "hidden" subtexts. Many are very difficult to discover automatically as they require complex knowledge such as context, Anaphora Resolution, Coreference Resolution and, in general, a detailed, almost innate, understanding of discourse (skills that human readers take for granted).

1.1.1 Rhetorical Structure Theory

Hovey and Maier [30] (unpublished) have analysed various forms of rhetorical analysis. Their research suggests that many techniques are similar in the types of rhetorical forms found in discourse. This project will attempt to discover relations from Rhetorical Structure Theory since it is a well-established theory with numerous implementations. It has a published corpus of annotated text - The RST Discourse Treebank contains 385 articles that have been annotated using RST [14] - and it has been used previously with Semantic Web technologies [8].

Rhetorical Structure Theory (RST) [42] is a Linguistic theory describing the organisation of natural language text. It characterises document structure in terms of Relations that hold between different parts of the text. Relations hold between two non-overlapping spans of text called the nucleus and the satellite. Relations exist within Schemas which hold a number of text spans and show the Relations between them.

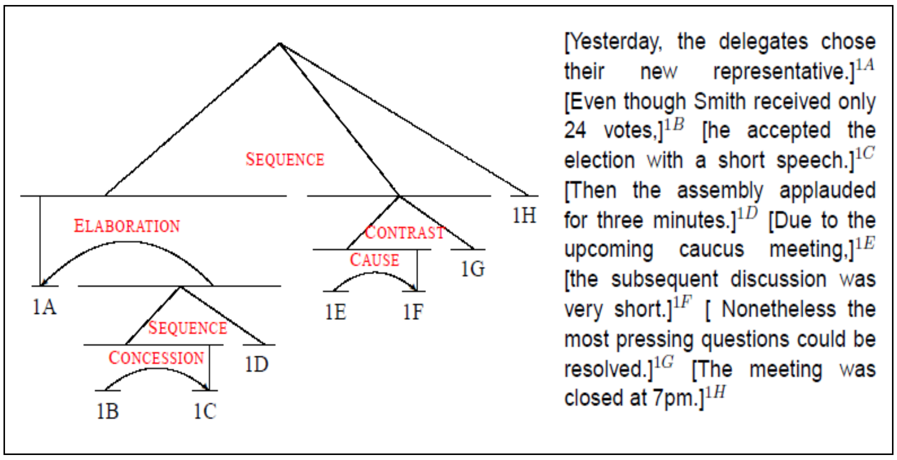
There are various types of Relations with specific meanings, e.g.

Evidence – where a Satellite is intended to increase the reader's belief in the nuclear material,

Concession – where the Nucleus indicates a contrary expectation in light of information expressed in the Satellite.

The number of relation types varies from about 30 [55] up to 65 [30] and 70 in [8], some more difficult to interpret than others. They are grouped into varying numbers of classes.

RST uses a diagrammatic approach [51] an example of which is shown in in Figure 2.



[Yesterday, the delegates chose their new representative.][1A] [Even though Smith received only 24 votes,][1B] [he accepted the election with a short speech.][1C] [Then the assembly applauded for three minutes.][1D] [Due to the upcoming caucus meeting,][1E] [the subsequent discussion was very short.][1F] [Nonetheless the most pressing questions could be resolved.][1G] [The meeting was closed at 7pm.][1H]

Figure 1.1.1.1 - Diagrammatic representation of RST annotation.

Nuclei and Satellites can be arranged hierarchically since they sometimes have internal structure. RST is also well-suited to summarisation since omitting satellite nodes below a certain depth in the hierarchy can result in coherent summarisation of the whole text [19].

Carlson and Marcu [13] have detailed a reference manual to implementing RST. This is a useful resource as it enables users not familiar with the details of the theory to begin analysing texts for rhetorical forms. The main steps to enable tagging of a document are: Segmenting Texts into Elementary Units then Determining Nuclearity and finally Selecting Rhetorical Relations.

The guide also contains a detailed inventory of different Relation types. There are many mechanisms used to determine where Relations exist, including Cue Phrases which are specific words or phrases that can be used as good indicators that Relations exist. These form a good basis for computational approaches as opposed to context or discourse-based cues [19,54,56,65]. The ability to analyse text automatically in this way is not possible using even all the available techniques, but combinations could be used in tandem to produce valuable output.

1.2 The Problem

There is a need for an automated tool to discover all forms of rhetoric in text. Manual annotation is well documented, but automated discovery in this domain is an open research problem. There has been some research and success into the automatic discovery of rhetoric [19, 51], but none have used Semantic Web reasoning mechanisms. Therefore the problem that is proposed to be addressed by this project is the lack of an automated, computational rhetoric-discovery tool which uses Semantic Web technologies (OWL and SWRL).

The manual analysis of language, specifically text, lies in the Linguistics domain and has a long history [5,40]. Building on linguistic theories developed during the 20th century, numerous attempts were made more recently to define rules and theories to better understand rhetorical forms and also to enable their analysis and documentation [57], e.g. Rhetorical Structure Theory (RST) [42], Discourse Representation Theory (DRT) [35], Segmented Discourse Representation Theory (SDRT) [6] and Discourse Lexicalised Tree Adjoining Grammar (DLTAG) [67].

In contrast to manual analysis, the automated discovery of rhetorical patterns in text is much harder to do:

> *"Linguistic accounts for rhetorical structure have the luxury to assume the availability of world knowledge and inference in rhetorical analysis. Automated systems resort, with quite some success, to a combination of little linguistic knowledge and a lot of shallow processing."* [51]

One approach to solving the problem is to furnish automated systems with more world knowledge. Our view is that large amounts of world knowledge are, for the first time in human history, becoming readily computationally useful and available via the Semantic Web and Linked Data. Projects such as Wikipedia (and DBPedia) provide unprecedented amounts of data with good standards [27] and are in effect available free of charge, permanently and persistently.

> *"Wikipedia, one of the most visited sites on the web, outstrips all other encyclopaedias in size and coverage. Its English language articles alone are 10 times the size of the Encyclopaedia Britannica, its nearest rival. But material in English constitutes only a quarter of Wikipedia—it has articles in 250 other languages as well."* [44]

> *“The DBpedia project focuses on the task of converting Wikipedia content into structured knowledge, such that Semantic Web techniques can be employed against it”* [7]

> *“By providing those extra links ... applications may exploit the extra ... knowledge from other datasets when developing an application; by virtue of integrating facts from several datasets.”* [71]

We think that one of the central aims of automated rhetorical analysis is to provide Meaning Representations – a formal structure that captures the meaning of linguistic expressions. These, in turn, can “*bridge the gap from linguistic inputs to the non-linguistic knowledge of the world needed to perform tasks involving the meaning of linguistic inputs*” [33] The meaning of linguistic inputs causes problems for computer analysis of natural language in many areas such as question-answering and language translation and tasks such as “*following a recipe*” or “*deciding what to order at a restaurant by reading a menu … require access to representations that link the linguistic elements involved in the task to the non-linguistic knowledge of the world needed to successfully accomplish them*”. [33]

Chapter 2 - The Proposal

The central task of this project is to utilise tools and theories from Linguistics previously shown, together with the burgeoning resource of information that is the Semantic Web and Linked Data in order to parse text and subsequently create and extend a knowledge base such that Meaning Representations in the form of Rhetorical Devices and Forms can be discovered and stored within the Semantic Web. It’s important also that any mechanisms of inference beyond knowledge-gathering are re-usable as part of the Semantic Web movement. We hope that this combination will provide a bi-directional benefit by enabling better automated analysis of text - specifically rhetorical structures - and also to The Semantic Web and Linked Data through automated tools, documented ontologies with logic reasoning and Linked Data resources.

It is a very difficult task to automate the discovery of natural language elements beyond syntactic analysis since natural language itself is highly complex and contains Meaning Representations operating at a level beyond simple analysis:

> *"Sources typically involved are meanings of words, conventional meanings associated with grammatical constructions, knowledge about the structure of the discourse, common-sense knowledge about the topic at hand and knowledge about the state of affairs in which the discourse is occurring". [33]*

Even the simplest of these sources - word meanings - can be very difficult to analyse since natural language, and especially English, can contain ambiguities, paradoxes and overlapping meanings of words, sometimes even within the same sentence, e.g. "I would like to eat somewhere local this evening."

A fuller description of this proposal that follows is divided into sections each describing an element of the proposed solution.

2.1 Parsing

The field of Natural Language Processing (NLP) is older than modern computing, but shares a similar development path over the last 60 years or so and, as the advent of powerful computing and statistical analyses have blossomed, so the application of algorithms, technologies and tools for the processing of natural language with computers has developed.

The requirement of this project is to apply methods of parsing and analysis of language to our text sources. The NLP computing system/s used must enable automated analysis, e.g. via API or Java implementations, be either open source or have minimal cost, and be able to perform most, if not all, of the following NLP techniques:

i) Tokenisation – the task of separating out words from running text, e.g. segmenting at whitespace or punctuation.

ii) Gazetteer – determining the presence of words from a named-entity list (Gazetteer)

iii) Sentence Segmentation – splitting a text into sentences

iv) Part-of-Speech tagging – the process of assigning a part of speech or other syntactic class marker to a word in a corpus, e.g. tagging types of nouns in a text as such.

v) Syntactic Parsing – the task of assigning syntactic structure to sentences. Parse trees are an output.

(Definitions above taken from [33])

This project proposes to investigate and develop statistical models using the General Architecture for Text Engineering (GATE) [20] and the Robust Accurate Statistical Parsing (RASP) [12] systems (with application to the Semantic Web [2,11,18] and in general use [22]). Both of these systems have the algorithmic facilities to develop appropriate solutions, are either free for non-commercial use or have open sources licences and can be used via automated procedures, e.g. Java implementation.

The General Architecture for Textual Engineering (GATE) is developed by Sheffield University's Department of Computer Science. It comes from a long-standing project and is an industry-standard tool for the analysis of text. It consists of a graphical user interface enabling drag-and-drop of plugin elements for parsing, analysis and output centred on natural language text. The version proposed to use is 5.2.1 which is the latest when this project was undertaken.

The main plugin for GATE is the ANNIE suite of algorithms. This group of modules allows tasks such as Sentence-Splitting, word Tokenisation, Gazetteer (lookup of proper nouns) and Part-Of-Speech tagging. Also included in the default setup for GATE is the Stanford Parser [58], LingPipe, RASP [12] and WordNet [70] plugin suites plus many more. There are also various plugins available individually and it's possible to create bespoke tools since the entire product is free for non-commercial use and the uncompiled code is available.

It is proposed to analyse various plugins for this project's goals and report on the findings. The criteria used to test each one will be based on the output's viability for the next phase of the proposed processing - populating the knowledge base – and also ease of use and effectiveness in a general setting.

2.2 Knowledge Base

“in many cases it is the attributes of a concept that act in the role of meaning representation.” [74]

In order to create Meaning Representations we need a model with the ability to represent objects, properties and relationships between objects in a Model-Theoretic Semantic scheme. This will enable our model to match the representation generated by our data. Figure 1 shows an example Attribute Relation Net for the “concept” of paper. This project is attempting to represent the domain via a “*bridge between the meaning representation and the world being considered*” [33]. In practice this means that we need elements in our model that represent the real world in some way and thereby allow a process to infer some meaning from it. For example a model containing an object type of **car** and property of **speed** can be instantiated to infer whether the car is travelling faster than the legal speed limit (also defined in the model). Similarly if an object type of **word** has a property of **upperInitial** we can infer that the word is either the first word in a sentence or perhaps a proper noun etc. The semantics of the model dictate the output to the real world and are defined by the model contents. Therefore the main mechanism in this project to deliver Meaning Representations is via a knowledge base containing objects, properties and relationships as well as a rule system to add inference along semantic lines. TBox is a term to refer to the vocabulary used to describe the facts of a domain (called the ABox) [33]. In this project we develop a TBox vocabulary and gradually populate the ABox facts via processing. The knowledge base will be populated from the output of the GATE process and then augmented by further processing of the text, the GATE output and also with input from Linked Data and the Semantic Web.

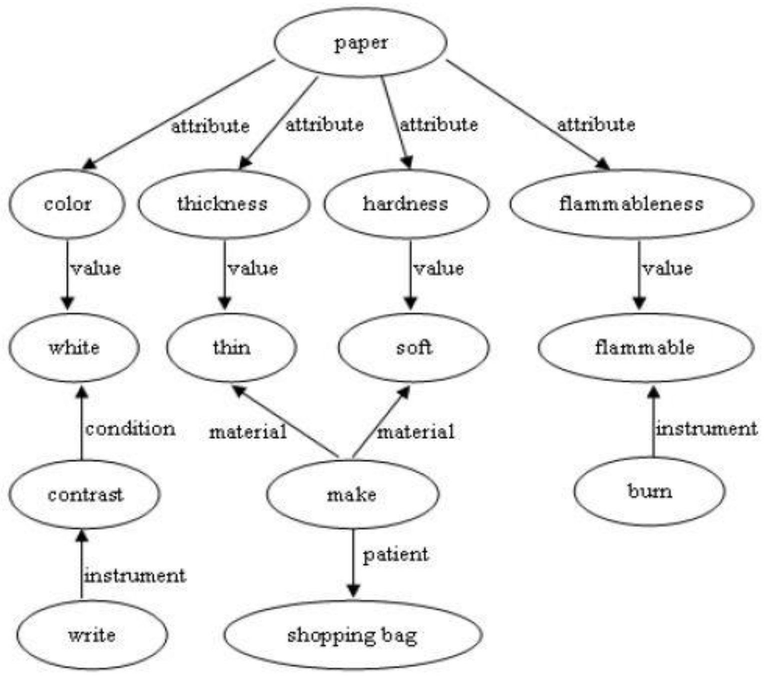


Figure 2.2.1 - Attribute Relation Net of “paper” [74]

In the Semantic Web domain a knowledge base is typically composed of an OWL or RDFS ontology and RDF triple documents. The ontology contains descriptions of domain entities, properties and relationships and the RDF triple store contains instances of entities and their relationships stored in a Subject-Predicate-Object triple format. The storage format is most often XML. We propose to follow these principles as they allow greater interoperability within the proposed output for the solution and the larger Semantic Web due to the ease of machine-readability of XML and the prescribed formats of OWL and RDFS.

The Semantic Web recommendations also make use of Description Logics in the form of the OWL subset OWL DL and Predicate Logic with the Semantic Web Rule Language (SWRL) although the W3C organisation has recently completed work on a Rule Interchange Format which provides a new mechanism for defining rule systems and managing the interchange between different ones. This has only been documented recently so cannot be used in this project. The choice of rule system for this project was dictated by a number of factors:

a) The expressivity of terms, e.g. built-in functions

b) Availability of Reasoning Systems that can use the rule system

c) A reusable storage format that can form part of the Semantic Web

d) Works seamlessly with OWL DL

SWRL seems to be the obvious choice as it meets these criteria (sometimes via a specific mechanism, however, such as the Protégé API). With the logical framework in place deductions can be made on the knowledge base in order to define "new" knowledge. SWRL is not a production rule language so new instances cannot be generated as an output of the reasoning process, however this will be described in more detail in future sections.

There are many other elements of the Semantic Web such as GRDDL and SKOS which deal with extracting RDF triples from XML data automatically and sharing and linking knowledge systems amongst systems on the Web respectively. These are out of scope of this project, but are potential tools for future iterations of the solution.

2.3 Ontologies and reasoning

> "The task of ontology is to explain Being itself and to make the Being of entities stand out in full relief". [43]

> "Ontologies can provide the necessary framework for consistent semantic integration, while additionally delivering formal reasoning capabilities to [Natural Language Processing]" [72]

An ontology, according to R. Studer is an "explicit and formal specification of a conceptualisation" and in the context of the World Wide Web provides a "shared understanding of a domain" [4].

This project will use ontologies to encapsulate knowledge about and reason upon the domain of textual language with specific application to discovering rhetoric. In order to achieve this it is proposed to use OWL DL.

The task of modelling all the domains generated by natural language is enormous, however as more information becomes available online and in a computer-readable format e.g. via DBPedia then the task will diminish. For the purposes of this project the domain of natural language must be modelled first and then topic-specific ontologies added second. Our initial thought with regards to natural language ontologies was the General Ontology for Linguistic Description (GOLD) [23,24,28,29]. This is a detailed online ontology of Linguistics. It was proposed to use this ontology as the main information source for language-specific entities, e.g.

OrthographicWord to represent a textual word, however we had problems maintaining links to the online presence of the ontology so we decided to model our own equivalent ontology. This is not a problem because we can create links to GOLD from our own OWL files and also the GOLD ontology has many more entities than we need for this project.

The output of the GATE suite also needs representation in the knowledge base. GATE has its own ontology-generation algorithm, however this was deemed to be too prescriptive and didn't offer the flexibility of programming the transformation of output into knowledge base that was required. We therefore propose to develop our own OWL ontologies for this task. We propose one ontology to model the gate output itself, e.g. Tokens, StartNodes, words, Dependencies etc., and a second ontology for the specific language tags generated by GATE, e.g. WhAdverb, PastParticipleVerb.

There is some crossover between GOLD and these proposed ontologies that model the GATE output, but this is an acceptable and perhaps even preferred approach. The Semantic Web is predicated on the fact that the same entity can be described in more than one ontology. This allows for greater flexibility, but at the potential expense of validity. As long as ontology designers create relationships between two entities that are genuinely the same then the system works best, e.g. via the owl:sameAs relationship defined in a knowledge base between two instances of objects or owl:equivalentClass between OWL classes.

> *"The Semantic Web's attitude to ontologies is no more than a rationalization of actual data-sharing practice. Applications can and do interact without achieving or attempting to achieve global consistency and coverage."* [73]

It's proposed to use owl:sameAs and owl:equivalentClass in any novel ontologies produced for this project.

Perhaps the most important ontology for the purpose of this project is one that describes the structure of the document (input text). Objects and relationships needed to be modelled include Document, hasFirstWord, hasLastWord, hasNextWord, hasNextSentence etc. This highly interlinked ontology will allow inference on instances that are in the same sentence (by virtue of having the same word instance defined in the sentence instance's hasWord property for example). Many natural language devices and characteristics can be described by reference to their position relative to one another. It would be next to impossible to describe a text without being able to define instances of objects being in paragraph, sentence, clause or phrase proximity to each other.

This document structure ontology will also contain a Hash representation of the input text in order to uniquely identify the text that was input to the application. This is important if the output is to be re-used as it will enable the publication of the rdf triples of the knowledge base online and therefore contribute to the Semantic Web in a small way. A user can define a passage of text and perhaps create links from other applications or ontologies to the unique resource identifier (that will contain the hash code itself) that is output from this project.

In order to classify different types of rhetorical forms an ontology of Rhetorical Devices will be created. This will include various instances of rhetorical devices, e.g. Adjunctio and Parelcon. This ontology will also contain Linked Data resources for various types of language structure characteristics. For example it will be necessary to describe words which have been shortened, e.g. often to oft. This could be described as a FinalCut OWL class since the final two letters have been cut to make this new word. Wherever it is necessary to store in the knowledge base patterns or characteristics of language not already captured they can be logged in the rhetorical devices ontology.

It’s possible that more ontologies will be required to capture or model domains as they become evident.

The overall output of these elements to the knowledge base will ultimately be combined by way of importation into a central OWL ontology. This will enable a consolidated view of the knowledge base to be obtained from a single OWL file and also enable relationships between the different ontological elements to be described. This overall ontology will also be the location of the SWRL rules stored in XML. The mechanism of converting SWRL rules in the form of Horn Clauses into an XML representation will be left to the Protégé SWRL Tab which can take the text input of the SWRL rules as Horn Clauses, validate the syntax based on the current ontology and then save into the OWL file in an XML format that’s reusable and standardised.

As well as designing and developing ontologies we propose to use existing ontologies/knowledge bases where appropriate. There are many ontologies and Linked Data sources already in existence and more are being made available all the time as part of the Semantic Web movement. We will endeavour to use this existing resource where possible.

DBPedia [7] is an online database that uses Wikipedia as the input to create a huge knowledge base of information. It is central to the Linked Data and Semantic Web movement. Accessible via various mechanisms including a SPARQL endpoint we propose to query this data and automatically add knowledge to the knowledge base as required.

Another ontological resource is WordNet [70] which is a lexical ontology used in linguistic applications and now part of the Semantic Web. WordNet provides information about word sense, antonyms, synonyms, and verb groupings etc. linked "*by means of conceptual-semantic and lexical relations*" [50], and can therefore provide relationships from text words to obtain alternative uses, meanings and groupings otherwise not automatically available.

In this project it may be necessary to develop or populate ontologies automatically. There has been numerous research efforts to do this from statistical approaches [10,32,48,63] to linguistic/ontological methods [1,60].

There are numerous other ontologies and knowledge bases currently forming part of the Semantic Web such as SUMO [61], YAGO [36], Cyc [21] and Freebase [25] and many have been used in linguistic projects previously [16,17,38]. This project will investigate the difference and benefits of using either or a combination of data sources.

The new standard method of querying the Linked Data resources and Semantic Web is via the SPARQL language [18]. It is based on the notion that Semantic Web data is to be stored in triples and probably in an RDF store.

The Semantic Web [9,71] has instigated a new wave of technologies, tools and research. Tim Berners-Lee's vision of a future World Wide Web where agents carry out complex tasks on behalf of users making use of a forest of linked ontologies being inferred and reasoned on by logic rules is very often the face of the Semantic Web. However, the ideas behind it – those of categorisation, modelling and documenting domains of knowledge and using rules to make inferences – are not new (for example in Knowledge Management, Pharmaceuticals etc). These two paradigms go together, but can exist in isolation.

Web Ontology Language (OWL) is *"aimed to be the standardized and broadly accepted ontology language for the Semantic Web*" [4]. It provides the necessary expressiveness, well-defined syntax and (in the case of OWL DL) permits efficient reasoning support to construct a representation of a domain.

Semantic Web Reasoning Language (SWRL) is a sublanguage of OWL, often marked up in RuleML [68]. SWRL combines Horn Logic-like axioms with an OWL knowledge base (ontology) in order to define a set of logic rules.

> "*First-Order-Logic (FOL) is a flexible, well-understood, and computationally tractable approach to the representation of knowledge that satisfies many of the desiderata given in Section 17.1 for a meaning representation language. Specifically, it [FOL] provides a sound computational basis for the verifiability, inference, and expressiveness requirements, as well as a sound model-theoretic semantics*." [33]

By combining OWL and SWRL a highly complex knowledge base can be constructed that can very efficiently, accurately and deterministically model a domain and allow both inference on elements in the domain and reasoning on rules based on domain elements.

Many research projects in the linguistics domain have investigated the benefits of using these technologies and techniques with significant success [8,26,38,34], but none use SWRL reasoning to our knowledge.

According to Tim Berners-Lee there are four principles that ought to be adhered to when designing and implementing Linked Data:

1) "*Use URIs as names for things*
2) *Use HTTP URIs so that people can look up those names*
3) *When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)*
4) *Include links to other URIs, so that they can discover more things*" [77]

This project will adhere to these principles wherever possible so that the outputs can go to form a small part of the Semantic Web / Linked Data movement if only in terms of the technologies and formats used rather than being published as part of Linked Data.

2.4 Protégé-OWL API

In this project the Semantic Web technologies previously mentioned will be used within the Stanford University tool Protégé, a domain modelling and knowledge base / ontology designer [59]. Protégé is open source Java software. This software is available freely online and comes with various add-ins, for example the SWRL Rules tab that allows SWRL rules as Horn Clauses to be input and validated. The Protégé GUI also allows Reasoning to be obtained via plugins such as DIG or Pellet. This project will only be using the GUI for designing ontologies and rules, however since the reasoning aspect of the solution will be done automatically as part of a process of analysis and output.

The most important aspect of Protégé for this project is that it has a programmable API in Java. This will be used to populate the knowledge base from within a Java program and use reasoning to infer new knowledge. This tool can also provide an output file of the entire knowledge base (in RDF XML format) for the particular text domain being analysed. This API also enables access to the Jena Semantic Web programming framework [75].

2.5 Rule engine

We intend to use the Protégé built-in API class of SWRLRuleEngineBridge in order to enable reasoning via a Rule Engine. This "*bridge provides a mechanism to:*

i) Import SWRL rules and relevant OWL classes, individuals and properties from an OWL model and write that knowledge to a rule engine

ii) Allow the rule engine to perform inference and to assert its new knowledge back to the bridge

iii) Insert that asserted knowledge into an OWL model" [76]

The Bridge allows users to specify the particular rule engine to be used for inference. In this case we will use the default rule engine: *"An implementation class for the Jess* *rule engine is supplied with the standard Protege-OWL distribution." [76]* The Jess API will be used under an academic licence provided by Westminster University.

Within the Bridge API it is possible to programmatically create SWRL rules (or other kinds of logic rules) and run them dynamically. This is not preferred in this project because, since these rules would be programmed in Java and compiled on a server, they would not be reusable. The reusability of outputs via OWL ontologies and logic rules is one reason why the Semantic Web is a powerful mechanism.

2.6 Storage and outputs

In recent times there has been a movement away from traditional relational database server infrastructures. This is partly due to the explosion in the quantities of semi-structured and unstructured data, e.g. HTML and XML (and a huge increase in data transport via the internet), but also because the costs of computing hardware have dropped significantly that it's possible to host computing and storage relatively cheaply. Services such as Amazon's Web Services [3] (2002), Microsoft's Azure Platform [45] provide Cloud Computing and Storage and thereby enable data-storage and computation remotely which can potentially reduce costs and management requirements. As The Semantic Web movement progresses there will be a requirement to store more and more XML data online, sometimes in flat text files, but for larger datasets more often in database systems that can store and process XML efficiently [15,53]. The increase in XML usage has resulted in many XML-based database systems (XDMS), e.g. Sedna [31], TigerLogic [62] etc.

2.6.1 HTML output

The final output of this project is proposed to be a web application that allows input text to be uploaded and will output a report containing information about the processing of the text and details of where various files are stored, e.g. output RDF. The input text will also be displayed marked-up with any rhetorical forms that were found by the tool. Notes and background information regarding the particular rhetorical patterns will be provided to give valuable information to users.

In order to put this solution online a suitable server solution is required. It's proposed to use an Apache Tomcat server since this provides Java programming environment (by way of jsp files) and is relatively inexpensive and readily available.

2.6.2 Comparison measurements

This tool needs to enable analysis and understanding of rhetorical forms and present them with a comparison measurement. This is an arbitrary process since we are aware of no other measurement in existence. It would only be valuable by comparing different texts and their outputs from the solution, however it's an interesting exercise in obtaining quantitative results. Our proposal is to use a ratio of the number of sentences that contain rhetorical patterns, even if there is more than one pattern per sentence, to the number of sentences contained in the entire text. This measurement will be displayed via the HTML output.

2.6.3 Other outputs

As well as the visible output via the HTML interface of the tool we propose to enhance the information output and benefits to the Semantic Web by publishing the knowledge base online for each text input. This will be formed by various files including:

i) automatically-created RDF output of the Protégé-OWL API Bridge – this is the entire knowledge base

ii) a text file log of the processing steps and comments of the program as it ran through the analysis

iii) the original source text in a txt file

iv) the output of the GATE analysis in XML format

All these output files will be named using the unique hash file generated automatically by the program at execution.

By storing all these documents and notifying the user of the hash code and the online location of these files they can be reused within the Semantic Web framework.

2.6.4 The Application

The HTML output already described is the proposed delivery method of an application that will enable users to upload text and receive annotated text back as an output. This annotation serves a number of purposes, of which the benefits to natural language processing and the Semantic Web are already described. Further to this the application itself is proposed to provide benefits to users who want to locate automatically rhetorical patterns and figures of speech in their own text and that of others. This may be useful for a user for various reasons such as for a scientist trying to reduce the figures of speech in a scientific paper, or an English student trying to increase the rhetorical content of an essay and thereby the persuasive power of their prose. Political analysts or historians could also highlight rhetoric in speeches or historic texts for various benefits.

2.7 Method

We propose to create a program written in Java that will provide the functionality to go some way to solve the problems described above. The method to create the program should follow these broad steps:

i) Develop a suite of OWL ontologies to describe elements within the domain of natural language text and the concomitant domains resulting from further analysis

ii) Develop a suite of logic rules using SWRL that can augment the knowledge base with rhetorical devices, figures of speech and RST annotations from reasoning on the knowledge base

iii) Enable text to be input by a user

iv) Automate parsing by the algorithms enabled via GATE

v) Automate population of a knowledge base using the Protégé-OWL API

vi) Test the validity of the output by inputting text known to have rhetorical patterns and also text that does not

vii) Develop a user interface published on the internet that enables all the above plus outputs a suite of files that effectively publishes the knowledge base for the input text and forms part of the Semantic Web by virtue of the technologies used

This project is seen as part of a first step along the long road of research necessary to solve the problems described previously, e.g. computers cannot represent natural language meaning very well. As such it's important that any outcomes are re-usable not just within the Semantic Web domain, but within the field of Computational Linguistics involved in language processing. For example the proposal must create generic and specific knowledge around language elements and the knowledge base built by this project will contain items that are usable both for textual processing, but also can be re-used in future analyses of natural speech or non-English text.

Chapter 3 - Related Works

Although Discourse Parsing, rhetorical analysis and the Semantic Web have many years of research history the fields have rarely been combined. We review here four research papers which have either used an ontological approach to language processing or have investigated automatic processing of rhetoric, specifically Rhetorical Structure Theory or a combination of these approaches.

3.1 Paper 1 – OWL ontologies as a resource for discourse parsing – ontological approach to discourse parsing using RST

Bärenfänger et al [8] developed an ontological approach to discourse parsing with RST. They produced two ontologies in order to develop their Generalised Annotation Parser (GAP): a novel RST taxonomy and the GermaNet German lexicon both using OWL DL ontologies. Using Prolog rules, German text is analysed through a multilayer, iterative parser in order to annotate the text for RST relations.

Using the classed relations of RST their RRSET ontology further categorises RST relations, see Figure 3. Their extended set of relations was formed by examination of previous research (Carlson and Marcu (2001) and Hovy and Maier (1995)), but specifically for the analysis of rhetorical structure in scientific journal articles.

Interestingly their RRSET ontology modelled RST relation as OWL classes rather than OWL object properties. This was because they wanted to declare disjointedness between relations and also allow hierarchical inheritance of relation properties. This is recommended as good practice for some situations [69]. The relationships between relations in the ontology are facilitated via rdfs:subclassof constructs, underspecified classes and multiple inheritance. Object properties hasNucleus and hasSatellite are used to indicate the role of the relation [41]. Figure 4 shows a fragment of the RRSET OWL ontology.

The second most common RST relation in their corpus was Elaboration. It was split into several sub relations based on the types of signals or cues that could be used to discover them. The following table shows examples of how this was broken down:

RRSET Relation (subset)	**Cue(s) / Discourse Marker(s)**	**Lexical-semantic relations**
Elaboration-Definition	Document structure cues, e.g. "<doc:glosslist>"	-
Elaboration-Example	Often signalled by lexical discourse markers "z.B.", "Beispiel" or "beispielweise"	-
Elaboration-Specification	Syntactic and punctuational discourse markers (e.g. non-sentential phrase within parentheses)	-
Elaboration-Derivation	-	Concept relations like Hyperonymy/Hyponymy, Holonymy or Meronymy
Elaboration-Continuation	-	Lexical relations like Synonymy, Pertainymy
Elaboration-Restatement	-	Lexical relations like Synonymy, Pertainymy
Elaboration-Drift	-	Lexical relations like Pertainymy

Table 3.1.1 – Examples of RRSET Relations

In order to identify Lexical-semantic relations a second OWL ontology was defined based on the lexico-semantic net GermaNet (Kunze et al., 2007). Two approaches to using this ontology in the GAP parsing system were direct consultation and by calculation of auxiliary lexical-semantic components by a lexical chainer or anaphora resolution system and then made available to GAP as additional layers in the input text. According to the authors GermaNet's coverage of German seemed not high enough (59.17% only) so they discarded the first approach.

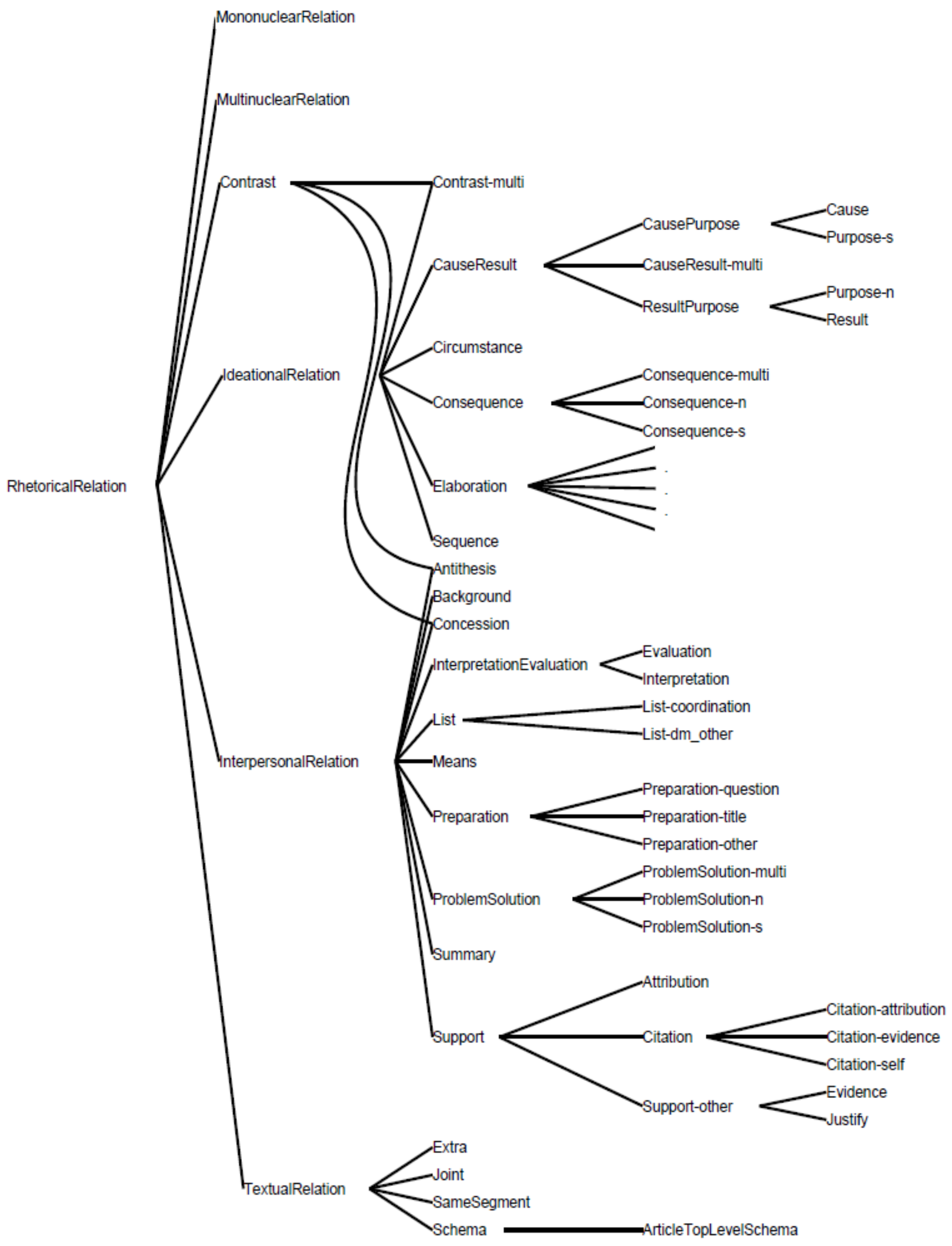
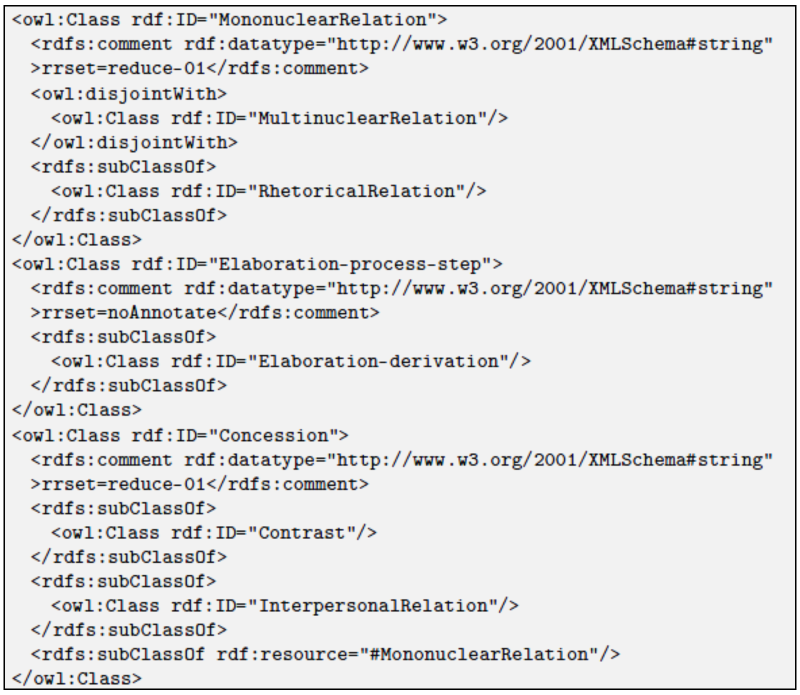


Figure 3.1.2 - RRSET ontology of RST relations

The GAP parser takes primary textual data and XML annotation layers as input which are converted to a Prolog fact base. The behaviour of the parser is controlled by a set of XML Reduce Rules. These rules are formulated from the cues, markers and lexical-semantic rules found to discover the RST relations, shown above. See Figure 5 for schematic of GAP.



```
<owl:Class rdf:ID="MononuclearRelation">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >rrset=reduce-01</rdfs:comment>
  <owl:disjointWith>
    <owl:Class rdf:ID="MultinuclearRelation"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="RhetoricalRelation"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Elaboration-process-step">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >rrset=noAnnotate</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Elaboration-derivation"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Concession">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >rrset=reduce-01</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Contrast"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="InterpersonalRelation"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#MononuclearRelation"/>
</owl:Class>
```

Figure 3.1.3 - Fragment of OWL ontology RRSET

The Reduce Rules are converted to Prolog to be used directly by the parser and contain access predicates which express connections between annotation layers, e.g. **identity(layer$_i$:element$_x$, layer$_j$:element$_y$)** or **text-inclusion(textvalue, layer$_i$:element$_x$)** or also application-dependent predicates which refer to the schema information of layers.

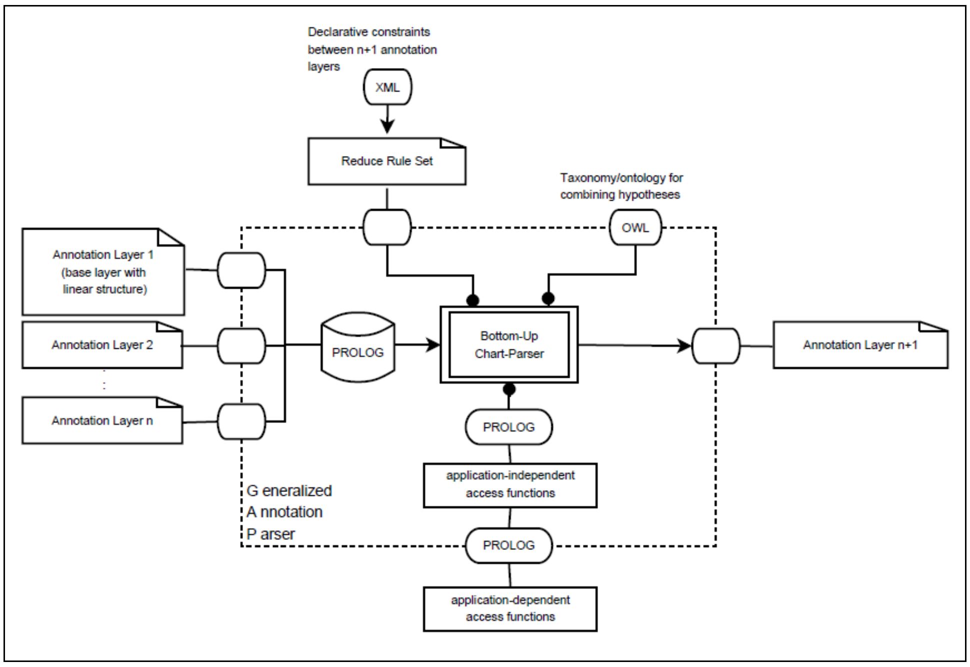

Declarative constraints between n+1 annotation layers
XML
Reduce Rule Set
Taxonomy/ontology for combining hypotheses
OWL
Annotation Layer 1 (base layer with linear structure)
Annotation Layer 2
Annotation Layer n
PROLOG
Bottom-Up Chart-Parser
Annotation Layer n+1
PROLOG
application-independent access functions
Generalized Annotation Parser
PROLOG
application-dependent access functions

Figure 3.1.4 - Generalised Annotation Parser (GAP)

In practice due either to ambiguities in language or the structure of the input and when there are no further indicators there results in more than one Reduce Rule that can apply to a text segment. In this case the lowest common superordinate relation according to the RRSET ontology will be used. This is derived from the subclassOf property that holds between classes.

3.2 Paper 2 – Using part-of-speech patterns and domain ontology to mine imprecise concepts from text documents – ontological approach to language processing

Abulaish et al [1] present "*a system that uses part-of-speech patterns and domain ontology to extract imprecise concepts by using linguistic variables – very, more light, strong*".

There is extensive research in the field of Information Extraction, but usually any information extracted by a system is assumed to adhere to precise concepts. In practice precise concepts are not often apparent and may actually be vague and imprecise. The authors' chosen research domain is wine documents, but can be applied to any domain. In this domain terms such as "very light" or "less strong" are not precisely defined. Different actors on this resource have not agreed the precise semantics, however by using a common model (ontology) interpretation an information exchange can be achieved without prior knowledge.

The W3C wine ontology is analysed in terms of its properties flavour, color and body (e.g. flavour{delicate, moderate, strong} ; color{red, white, rose} ; and body{light, medium, full}). The authors looked at wine documents online and show that there are many terms used which do not match to the W3C wine ontology. They propose:

a) development of a fuzzy ontology

b) a schema of a knowledgebase created from a) and using the two together to extract information from unstructured text

c) store unstructured information in a structured knowledgebase

The system proposed (see Figure 6 for schematic diagram) consists of an Ontology Editor which creates the fuzzy ontology based on the domain ontology and resources qualifiers (fuzzy concept modifiers), the Document Processing Agent (DPA) which comprises a document parser, a part-of-speech (POS) tagger and a term filter. The DPA chunks the input text, tags the words with the POS tagger and filters unwanted tags. The output is a ternary tree structured document which, together with the fuzzy ontology are used by the Knowledgebase Instance Generation Agent (KIGA) to populate the new knowledgebase.

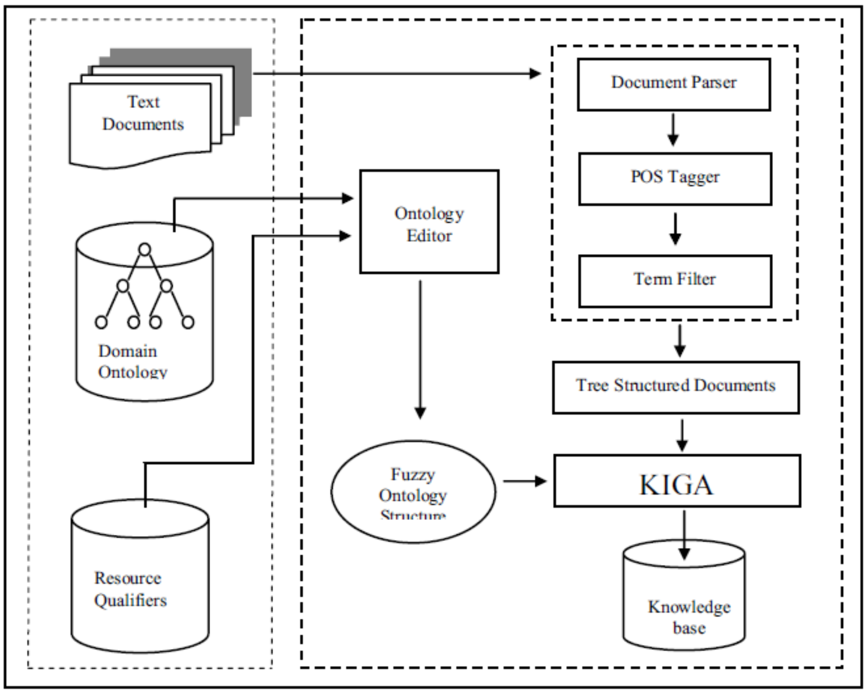


Figure 3.2.1 - proposed system schematic

Using Protégé the authors define a fuzzy ontology structure incorporating **qualifiers** - usually English adverbs, e.g. “very”, “slightly”, “less” etc. These are used in combination with the pre-defined domain value restrictions, e.g. “delicate”, “light” in the case of the wine ontology. In this paper the qualifiers considered in relation to the wine ontology properties are:

Domain Property	**“Fuzzy” Qualifiers**
Color	{null, light, pale, bright, dark, deep}
Flavor	{null, slightly, very}
Body	{null, very}
Taste	{null, slightly, semi, medium, very}

Table 3.2.2 – Domain Properties and Fuzzy Qualifiers

Null values are used to capture example in the text without qualifiers, e.g. “red wine” as opposed to “pale red wine”.

New inherited classes are defined FuzzyColor, FuzzyFlavor, FuzzyBody and FuzzyTaste which are subclasses of both the value and qualifier classes. Property values are next constrained using the allValuesFrom constraint to be either null or an instance of the corresponding fuzzy class.

The DPA divides the input text into record-sized chunks and presents them to the POS tagger. An example of types of tags and some text tagged with parts of speech are shown below in Figure 7.

T	Article	N	Noun	R	Preposition	D	Determiner
X	Auxiliary verb	V	Verb	P	Pronoun	I	Interjection
A	Adverb	J	Adjective	C	Conjunction	i	“to” as an infinitive marker

```
Dolcetto is a red table wine which is quite dry and has a slightly fruity flavor. Syrah has a full body, less fruity flavor and dark red
N        X T J  N    N    P    X A   J  C  X  T  A    J     N     N    X  T J  N    A  J    N    C  N  J

color that grow originally in California's coastal areas.
  N   C  V      A       R  N         J        N
```

Figure 3.2.3 - Part-Of-Speech tag types and an example tagged text

The last step in the DPA is to filter out unwanted tags. In this case the X (Auxiliary verb) and T (article) tags are removed. The output is put into a tree structure by dividing segments based on commas, semicolons, prepositions (e.g. “with”), conjunctions (e.g. “and”) and full stops.

The next step is for the KIGA element to parse the fuzzy ontology and create an SQL schema for the knowledgebase, implemented as a series of create table statements. Then the output of the DPA is used to populate the new relational database schema.

Population takes a two-way approach – first, given a property name, the system looks for values to fill up the object description and then, for missing property names, property values from the ontology are used as pointers to fill up the attributes in the database. This process caters for language that doesn’t necessarily use property names when property values are present, e.g. “*Roussanne is a light bodied, light red and very sweet wine from France’s Loire Valley*” where the Color property is not named directly. There may be many null values passed to the database where properties are missing for the particular object being described.

Based on the following measures the results achieved were:

$$\text{Precision} = \frac{\text{No. of relevant concepts extracted}}{\text{Total no. of irrelevant and relevant concepts extracted}} = 92.89\%$$

$$\text{Recall} = \frac{\text{No. of relevant concepts extracted}}{\text{Total no. of concepts}} = 80.69\%$$

Future work for this project is to learn from extracted information, e.g. incorporate new qualifiers, values and properties.

3.3 Paper 3 – Beyond string matching and cue phrases: Improving efficiency and coverage in discourse analysis – automatic processing of RST

Corston-Oliver's 1998 paper [19] describes a discourse analysis component within the Microsoft English Grammar (MEG) system called Rhetorical Structure Theory Analyzer (RASTA).

The author discusses various methods of recognising discourse relations, e.g. using references to world knowledge, reasoning with representations of propositional content and reliance on cue words, phrases or lexical repetition. The last method is preferred in this paper since the other two have questions that are "*by no means resolved*".

Pattern matching techniques like Regular Expressions have problems such as an unreliable ability to identify terminal nodes in RST analysis and potentially identifying two syntactic interpretations for a single lexical term. To alleviate these problems RASTA combines pattern matching (cue phrase identification) with an examination of syntactic analysis and logical form to identify discourse units. RASTA assigns heuristic scores to its judgements and by a series of checks against a set of necessary criteria. If the criteria match then a second series of cues is checked, see Figures 8 and 9 for examples.

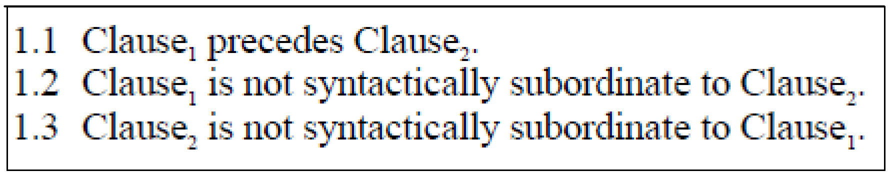

1.1 Clause$_1$ precedes Clause$_2$.
1.2 Clause$_1$ is not syntactically subordinate to Clause$_2$.
1.3 Clause$_2$ is not syntactically subordinate to Clause$_1$.

Figure 3.3.1 - Necessary criteria for the Elaboration relation

The text to be analysed (a subset of the Encarta encyclopaedia) is first segmented by MEG into clauses and RASTA then examines them relative to one another in turn (e.g. Clause$_1$ versus Clause$_2$) for the following RST relation types: AsymmetricContrast, Cause, Circumstance, Concession, Condition, Contrast, Elaboration, Joint, List, Means, Purpose, Result and Sequence.

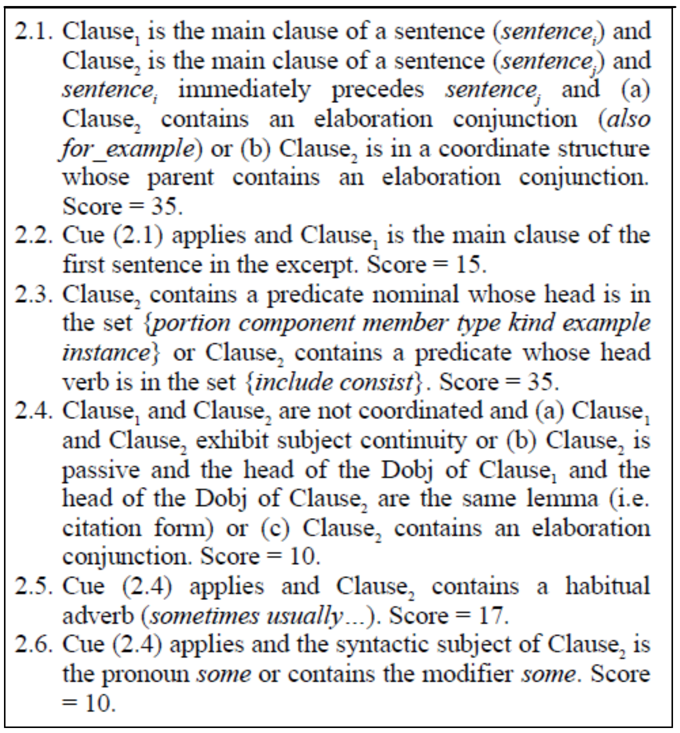

2.1. Clause$_1$ is the main clause of a sentence (*sentence$_i$*) and Clause$_2$ is the main clause of a sentence (*sentence$_j$*) and *sentence$_i$* immediately precedes *sentence$_j$* and (a) Clause$_2$ contains an elaboration conjunction (*also for_example*) or (b) Clause$_2$ is in a coordinate structure whose parent contains an elaboration conjunction. Score = 35.
2.2. Cue (2.1) applies and Clause$_1$ is the main clause of the first sentence in the excerpt. Score = 15.
2.3. Clause$_2$ contains a predicate nominal whose head is in the set {*portion component member type kind example instance*} or Clause$_2$ contains a predicate whose head verb is in the set {*include consist*}. Score = 35.
2.4. Clause$_1$ and Clause$_2$ are not coordinated and (a) Clause$_1$ and Clause$_2$ exhibit subject continuity or (b) Clause$_2$ is passive and the head of the Dobj of Clause$_1$ and the head of the Dobj of Clause$_2$ are the same lemma (i.e. citation form) or (c) Clause$_2$ contains an elaboration conjunction. Score = 10.
2.5. Cue (2.4) applies and Clause$_2$ contains a habitual adverb (*sometimes usually*…). Score = 17.
2.6. Cue (2.4) applies and the syntactic subject of Clause$_2$ is the pronoun *some* or contains the modifier *some*. Score = 10.

Figure 3.3.2 - Cues for Elaboration relation

RASTA assigns higher heuristic scores to cue words and phrases than syntactic evidence, however a combination of several syntactic scores can have a higher influence on the overall score.

The outcome of the criteria and cue tests is a plausible RST analysis. For example the Elaboration relation discovered (in Figure 10) via cues 2.4, 2.5 (between clauses

5.1 and 5.2) and via cues 2.4, 2.6 (between clauses 5.3 and 5.4). A Contract relation is also located in this example.

Given the relations found by the analysis there are many ways that they could be joined into a structure over the whole document. RST trees provide a mechanism to achieve this, but there are potentially many such trees on the same document. The author constructs trees using Marcu's algorithm. This algorithm analyses the different types of relations in terms of their symmetry of nuclei and satellite relations. Trees are then built from the bottom up by joining text spans that are in a relationship. As the number of trees grows the computations involved can become very large. This is made worse by the fact that Marcu's algorithm grows all possible trees first and then cuts back ill-formed trees afterwards. Proposed in this paper is a backtracking algorithm that "*applies the relations with the highest heuristic scores first in the bottom up construction of the tree*".

The backtracking algorithm uses the counts of heuristic scores for each relation between nodes, ordered by value and at each step of constructing a tree discards trees that will end up as ill-formed. In this way the algorithm is more efficient and less computationally demanding.

The heuristic scores used in the analysis of relations were initially set by the author's knowledge and intuition, however subsequently have been modified with reference to the results. At the time of writing there were no corpora of RST analyses available (shortly after this paper Calrson, Marcu and Okurowski developed their tagged corpus based on 385 Wall Street Journal articles [14]). The author asserts that manual tuning of heuristic scores is necessary. Statistical analysis suggests that machine learning techniques would not improve the output. Performance indicates that the preferred tree is output after only 1.92 trees being calculated (with significant tolerances, however).

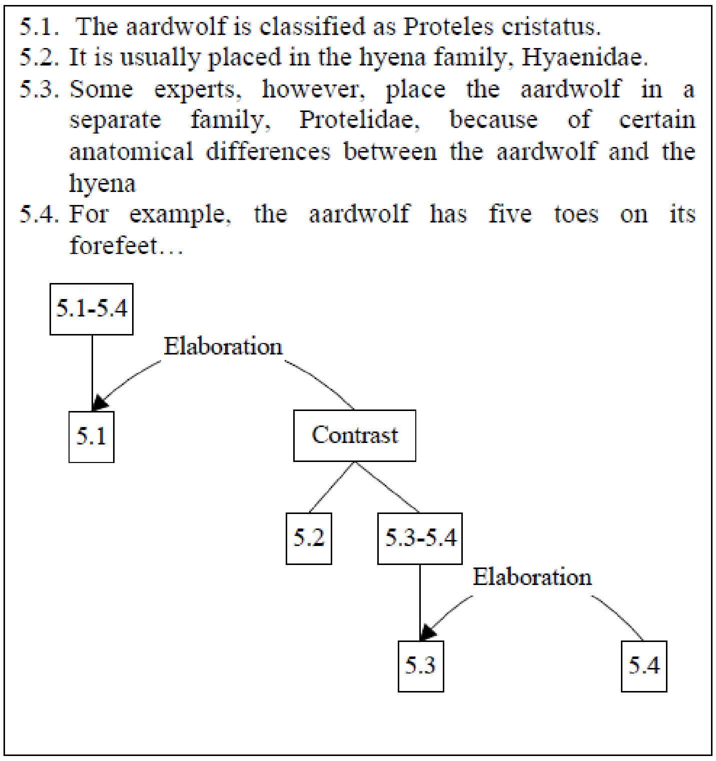

5.1. The aardwolf is classified as Proteles cristatus.
5.2. It is usually placed in the hyena family, Hyaenidae.
5.3. Some experts, however, place the aardwolf in a separate family, Protelidae, because of certain anatomical differences between the aardwolf and the hyena
5.4. For example, the aardwolf has five toes on its forefeet…

Figure 3.3.3 - RST analysis on Aardwolf text

The author suggests extensions to RASTA that could improve performance, e.g. using domain-specific cues during the relation discovery phase and, by varying the *type* of heuristic scores, different types of analyses could be obtained, e.g. some texts can have simultaneously intentional and informational representations which cannot be distinguished in the current model, but by using two sets of heuristic scores (one for intentional and one for informational) the output could be split depending on the requirement.

3.4 Paper 4 – Simple signals for complex rhetorics: On rhetorical analysis with Rich-Feature Support Vector Models – automatic processing of RST

This 2003 paper by Reitter [51] takes a statistical – specifically machine learning – approach to analysing rhetorical patterns in text. After describing Coherence (the property of textual language elements to interrelate to one another to form a discourse) the author explains that world knowledge and inference are difficult to automate therefore a little linguistics knowledge and shallow processing of text are the best approaches. He next describes RST by explaining that relations can be either Paratactic, containing two or more equally weighted spans of text with the same role assigned to each, or Hypotactic which is unequally weighted elements with the roles of either Nucleus or Satellite. This paradigmatic idea is extended into the formation of tree-structured analysis suggesting that there is only one structure that is faithful to the writer's intentions, however Reitter recognises that he attempts to derive the most likely intention.

Rhetorical parsing is mentioned with reference to RASTA and Marcu's algorithm and also Marcu's and Echihabi's 2002 investigation using unsupervised machine learning.

The author next looks at URML, an XML-based standard for rhetorical annotations. Since the ability for computers to "read" texts for linguistic analysis has become more important over time a standard markup format has been used based on SGML (a superclass of XML). URML has been used as a solution to various problems with documentation/markup of texts due to its underspecified approach (not requiring explicit markup of all information) and standardised methods. URML uses a referential rather than in-situ markup therefore necessitating indexing of relations and nodes which is relatively easy in XML.

Reitter proposes to diverge slightly from the original RST by representing only binary relations, but this is allowed since the nucleus will be preserved and this is the more important element in terms of conveying meaning and also since multi-satellite relations can be preserved using binary structure in a hierarchical branching structure. Also, by analysis of the LDC corpus [14] he shows that only 9 out of 17962 relations share a nucleus with others of a different type.

The method proposed focuses on using various surface cues at various linguistic levels, e.g. morphosyntax, punctuation, lexical choice and discourse marker. These cues, however, do not map simply to RST relations, e.g. "*even though*" can signal either Concession or Contrast. The process for the analysis progresses as follows:

1. Split relations into Classification Instances:

The parser proceeds segment by segment through the document where each Paratactic node is split into several Classification Instances and Hypotactic nodes are converted to one Classification Instance. The Classification Instance is an intermediate data structure made up of a relation, a list of text span indexes and a single span (in Hypotactic relations). A Feature is a function that assigns a scalar value to a linguistic pattern based on a single property of the Classification Instance.

2. Transforming Classification Instances into Feature Vectors:

Templates for Features derive from the training corpus and include:

i) Cue words and pronouns: defined by the corpus and encoded with a positional value, e.g. beginning or end of the text span

ii) Introduction of concepts: noun phrases occurring in notable locations can indicate specific meaning, e.g. in journalistic texts definite noun phrases with a common noun tend to refer to a concept already introduced in the text

iii) Punctuation: colons, dashes, question marks etc

iv) Part-of-Speech categories: types of words used at the borders of segments can help disambiguate relation types

v) Lexical similarity: using Hearst (1997) a similarity measure can be used to reflect the semantic similarity of two text spans

vi) Span length: the ratio of text span lengths (in words) is used as a feature, hypothesised based on intuition.

Using the statistical machine learning approach of Support Vector Machines (SVM) the author supplies a training sample which outputs a model of inference which can be used on unseen examples. This is a classification technique which attempts to differentiate the target classes based on input vectors in effect creating a linear boundary between classes. Figure 11 shows an example graph.

The data needs to be transformed into a linear feature space first using a **kernel trick** (in this case a radial basis function). The location on either side of the hyperplane (straight line in two dimensions) indicates the class of the instance and the distance to the line shows the confidence in the outcome. A binary SVM is trained on each class and the best solution chosen to be the one with the highest confidence score.

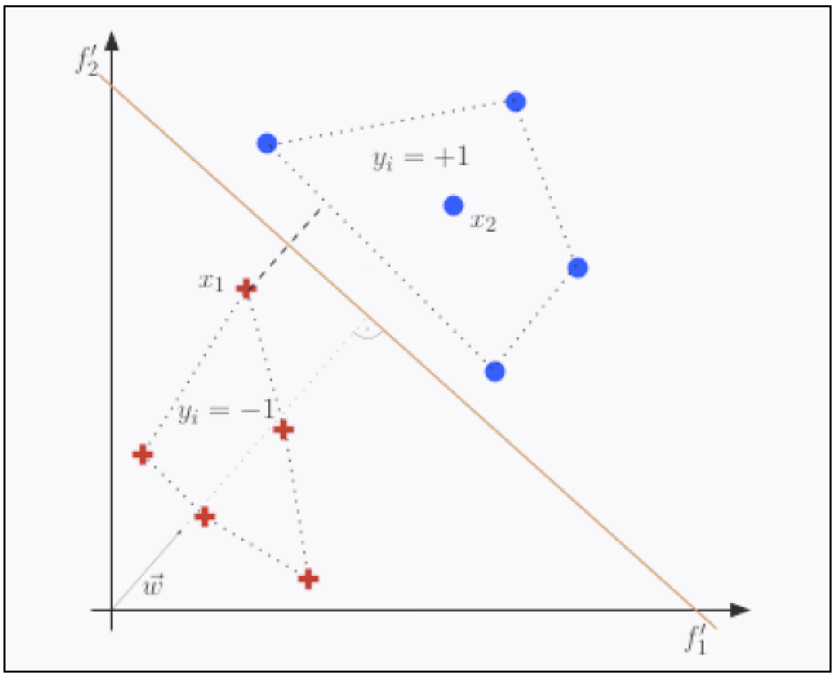
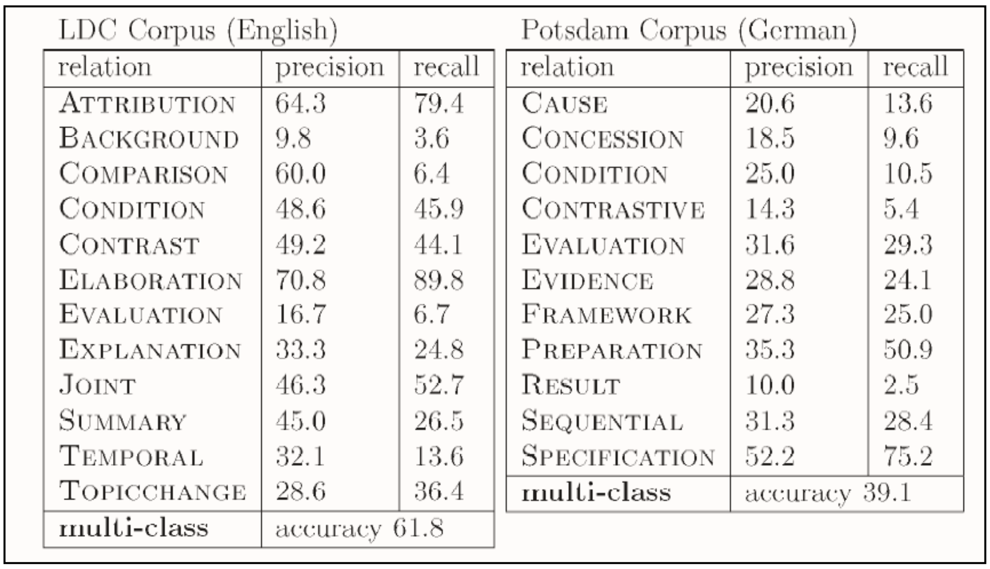


Figure 3.4.1 - large-margin separation output of SVM analysis

The test corpora in this paper are the LDC Wall Street Journal collection of RST-annotated texts (385 English language newspaper articles) and the Potsdam Corpus – a series of German language news commentaries published in the Märkische Allgemeine Zeitlung. The LDC corpus was converted into URML and part-of-speech tagged with all the existing relations normalised into 16 general RST relations. The Potsdam Corpus was manually annotated with RST using the RSTTool 3.1 and cross validated by different analysers to ensure an unbiased sample.

The results are shown in Figure 12. They indicate excellent performance (for comparison the baseline measure of the Elaboration relation is 33.6%).

Some of the more simple relations, e.g. Elaboration, Specification and Attribution are successfully annotated with this model, however some harder-to-discover relations, e.g. Cause and Result are not solved well. Cue phrases and Punctuation are determined to be the most important signals for classification accuracy.



LDC Corpus (English)

relation	precision	recall
Attribution	64.3	79.4
Background	9.8	3.6
Comparison	60.0	6.4
Condition	48.6	45.9
Contrast	49.2	44.1
Elaboration	70.8	89.8
Evaluation	16.7	6.7
Explanation	33.3	24.8
Joint	46.3	52.7
Summary	45.0	26.5
Temporal	32.1	13.6
TopicChange	28.6	36.4
multi-class	accuracy 61.8	

Potsdam Corpus (German)

relation	precision	recall
Cause	20.6	13.6
Concession	18.5	9.6
Condition	25.0	10.5
Contrastive	14.3	5.4
Evaluation	31.6	29.3
Evidence	28.8	24.1
Framework	27.3	25.0
Preparation	35.3	50.9
Result	10.0	2.5
Sequential	31.3	28.4
Specification	52.2	75.2
multi-class	accuracy 39.1	

Figure 3.4.2 - English and German model performance

Chapter 4 – Architecture of LaROS

In creating the solution for this project we actually employed an iterative development process, however each task will be described here in a logical order reflecting the proposed method from section 4.7.

4.1 Ontological Design

We realised that we would need ontological influence in a number of areas to create a working knowledge base. Therefore the first tasks involved in creating the solution were ontology investigation and design. In order to have as much re-use as possible we analysed existing ontologies for suitability for inclusion in the solution. Suitability was mostly confined to two characteristics: did the ontology have the entities and relationships required for the task and was it available in OWL format. The areas we would require ontologies for were

a) the GATE processing output, including the Stanford Parser

b) text entities and relationships

c) higher level text entities and relationships

d) external entities and relationships and associated intra-ontology relationships

e) rhetorical devices, figures of speech and RST relations

The general approach of analysis taken was to model entities when they referred to real things or concepts, for example a word can be thought to exist in the context of written natural language; similarly a sentence or paragraph. It can also be said that a word will generally be related to another word that precedes it in a text, and also a word that succeeds it too (obviously unless it is the first or last word in the section), in a number of ways, but most obviously in the fact that the words are adjacent. Various words can also be said to be the first or last of their kind that appear within a parent entity such as a sentence or paragraph etc. Entities can be related to one another by relationships such as **hasPararaph** or **hasWord**. This kind of modelling can be done to a very detailed level if all relationships are modelled, however in this case we included entities and relationships only if we thought they would be either generally useful or had a direct implication on the rule system to be developed later in the project.

We were unable to find an ontology that had this kind of model so we created one. This model eventually became the Document Structure ontology (see Appendix 1). This general approach of modelling entities, properties and relationships that can be thought to exist rather than a contrived set of entities was preferred. An alternative approach to modelling would be to use more specific entities such as **firstWordInSentence** or **lastParagraphInDocument**, but this seemed not to allow the flexibility of a more generic method and also would have created larger and more difficult-to-manage Domains or Ranges for OWL properties created later.

The Document Structure ontology is central to many of the rules developed in this project. Most meaning in language involves the relationships between words or collections of words and their individual meaning and also the locational representation of the entities. This might be less important in a more inflexed language, however it would still be important to build into any knowledge base the notion of whether a word is a part of a sentence that has some specific kind of relationship with another part of the same document. This is achieved in this ontology by some of the properties mentioned previously.

Other important properties in this ontology include **sameAsWord** which relates two word class instances to each other by virtue of the fact that their orthographic representation is the same. It does not mean that the words have the same meaning and in fact one of the output rules from this project uses the fact that the same word in appearance can have different meanings in a particular manner giving rise to a known rhetorical device. The **hasHashOf** property relates the Doc class instance to a string value of the hash code which identifies uniquely the input text.

The outputs of the various modules within GATE that create mark-up tags for the overall GATE output also needed to be modelled in the knowledge base. We couldn't find OWL ontologies that provided what we needed so we developed two new OWL ontologies for this task. The LangTag ontology contains a mapping of each output of the Category tag, for example Verbs, Nouns and various sub categories are entities that are mapped to, based on the output category code from GATE. See appendix 2 for the LangTag ontology diagram.

Also derived from the GATE output is the Gate ontology (gate.owl). This is shown diagrammatically in Appendix 3. This ontology attempts to model some of the main entities that describe our input text and in fact most text documents. Derived from GATE are, for example, the **word**, **Sentence** and **paragraph** entities, which are OWL classes in the ontology. Also the properties that define the location of entities within the text, such as **hasStartNode** and **hasEndNode** were included in the Gate ontology.

Where necessary the object terminology output from GATE was changed to reflect the ownership or entity type, for example the **Length** characteristic from GATE becomes the **hasLength** datatype property in our Gate ontology. Similarly the **hasString** datatype property represents the string value that is the word's actual orthographic character representation. Another important property contained in the Gate ontology is the **hasDependency** object property. This relationship between two words is the output of the Stanford Parser GATE plugin. It allows words to be related to each other based on the role in the sentence. For example if a verb is related to the object noun in a sentence then the dependency values will be output by GATE. This is translated into the object property of **hasDependency** in the knowledge base. This allows us to analyse whether words perform certain roles together with other properties and if so to infer new knowledge.

Once we were able to describe the different aspects of our domain in terms of our ontological framework we began to develop the framework of further ontologies that model more complex behaviour and also links to outside information such as Linked Data. The VerbNounCombo ontology is a small OWL ontology that has two important classes: **UnusualNounInCombo** and **UnusualVerbInCombo**. In our domain it's possible that a verb/noun combination will be an unlikely pairing. This could be for a number of reasons (we rule out errors), but one in particular is the deliberate use of an unusual pairing of words for rhetorical effect. Details of this rule will be shown later, but these two classes are used to define verbs and nouns as being used unusually.

Two more ontologies are used to model rhetorical content. The Rhetorical Devices ontology has the class of **RhetoricalDevice** which also has a number of instances in the same OWL file which equate to the rhetorical devices (or figures of speech) that are to be located by the tool. This list is very small compared with the number of possible devices that have been documented, however this project doesn't have all patterns in scope so only about 20 are listed as of this project's completion. Related to the **RhetoricalDevice** class is the **hasRhetoricalDevice** object property. This OWL object property relates the devices to the Doc instance in question thereby allowing the population of the knowledge base with the knowledge that the device has been found. In order to specify the exact location of the pattern the Gate properties **hasStartNode** and **hasEndNode** are used. This enables the user to pinpoint exactly where in the document any patterns have been found.

This ontology also contains some more complex descriptive classes that are used to add more information to the knowledge base. For example when a word has a number of characters removed from its middle for effect this is classed as a **MedialCut**, for example "libary". This is not classed as a misspelling but a deliberate play on words. Similarly for characters cut from the end of a word, such as "oft", the word would be classed as **FinalCut**. The **EpitheticalName** class is used to hold specific names of people used in a rhetorical fashion. The use of instances of these classes contained in the OWL ontology enables the processing mechanism to scan the input text for these particular patterns and tag the resulting information in the knowledge base.

The second rhetorical content ontology is the Rhetorical Structure ontology which is designed to model RST analysis within the knowledge base. This simple OWL file contains one important class – **RhetoricalStruct**. The class has a number of instances which equate to RST relation types. In a similar way to the RhetoricalDevices ontology the instances can be related to a Doc instance and also have start and end locations logged.

The last ontology used in this project is the Lassoing Rhetoric OWL ontology. This ontology brings together, by importation, all the previously-mentioned OWL files. It also imports, by namespace reference, classes from DBPedia and WordNet. Once all the classes, properties and individuals are combined in one ontology SWRL rules can be defined more easily. The ontology has a suite of SWRL rules that act upon the knowledge base to add new knowledge when executed in the correct context.

4.2 Rhetorical forms and logic rules

This project's goal was to discover rhetorical forms in text. An example of a rhetorical form is Anaphora. This occurs when there is "*repetition of the same word at the beginning of successive clauses or versus*" [40]. An example is found in Winston Churchill's speech to Parliament in 1940 when he uses the phrase "We shall" at the beginning of various clauses, for example "We shall fight them on the beaches". In order to discover this pattern in text using a knowledge base and logic rules we needed a number of entities namely a Document, a paragraph, a sentence and at least two clauses. We also need the knowledge elements to be related such that a clause that has an adjacent clause and has the same first two words (by orthography) can be related to a sentence and also a parent paragraph and then to a parent Document. This is a very basic logical test and can be summarised by the following syllogism:

IF Document HAS Paragraph
AND IF Paragraph HAS Sentence
AND IF Sentence HAS Clause A
AND IF Sentence HAS Clause B
AND IF Clause A HAS Word X
AND IF Clause A HAS Word Y
AND IF Clause B HAS Word F
AND IF Clause B HAS Word G
AND IF Word X IS THE SAME AS Word F
AND IF Word Y IS THE SAME AS Word G
THEN Document HAS Anaphora

We assumed for this rule that the meaning of the word is irrelevant. It's simply the repetition of the same orthographic form that signals Anaphora. An Anaphoric pattern could, of course, extend to more than two matching clauses and vary over more than two adjacent clauses also, however for this project we are only concerned with developing a single rule for each rhetorical pattern selected. This solution is not intended to be a complete rhetorical device locator.

This syllogism can be converted to Horn Clauses quite easily and therefore we were able to generate a suite of SWRL rules as Horn Clauses based on the logic of discovering new knowledge from the knowledge base built up for the text under analysis. The SWRL rule for Anaphora is this:

```
DocStruct:Doc(?h) ∧ DocStruct:hasParagraph(?h, ?i) ∧
DocStruct:hasSentence(?i, ?z) ∧ gate:word(?x) ∧ DocStruct:hasNextWord(?x, ?y) ∧
gate:Sentence(?z) ∧ DocStruct:hasFirstWord(?z, ?x) ∧ gate:word(?a) ∧
DocStruct:hasNextWord(?a, ?b) ∧ gate:Sentence(?c) ∧
DocStruct:hasFirstWord(?c, ?a) ∧ DocStruct:hasNextSentence(?z, ?c) ∧
gate:hasString(?x, ?d) ∧ gate:hasString(?y, ?e) ∧ gate:hasString(?a, ?f) ∧
gate:hasString(?b, ?g) ∧ swrlb:equal(?d, ?f) ∧ swrlb:equal(?e, ?g) ∧
gate:hasStartNode(?x, ?j) ∧ gate:hasEndNode(?b, ?k) →
RhetDev:hasRhetoricalDevice(?h, RhetDev:Anaphora) ∧
gate:hasStartNode(RhetDev:Anaphora, ?j) ∧ gate:hasEndNode(RhetDev:Anaphora, ?k)
```

The pre-existing instance of RhetDev:Anaphora is used in the consequent to create an object property relationship to the Doc instance. The Anaphora object is then related to start and end nodes for future use within the tool.

Since Protégé allows the input and validation of SWRL rules in the form of Horn Clauses all the rules in this project were validated using Protégé against the Lassoing Rhetoric OWL ontology. Each rhetorical form was analysed and then a SWRL rule was designed using the knowledge base elements. Our SWRL rules are listed in Appendix 9.

Since SWRL is not a productive rule system it was not possible to have instances generated from a rule. All variables in the rule must refer to existing entities or literals. The effect of this was that in order to have rhetorical forms "discovered" they must already exist at the time of inference, i.e. be a part of the knowledge base when the rule is executed. As already mentioned the various instances for the RhetoricalDevice class are contained in the Rhetorical Devices ontology. These were used in the Consequent of each rule and came into play wherever all the atoms in the Antecedent of the rule were true.

As the SWRL rules were designed and saved into the Lassoing Rhetoric OWL ontology via Protégé they were converted into an XML representation. The XML representation consists of a swrl:Imp outer tag and contains inner swrl:body and swrl:head tags. These hold the antecedent and consequent parts of the Horn Clause respectively.

An important output of this project is that these rules are available online as part of an ontology and in a format that is re-usable in other work.

4.3 Parsing with GATE

The GATE infrastructure is a software environment that allows the development of processing on human natural language text. It is Java software that comes with a sophisticated user interface from which various Pipelines can be constructed. A Pipeline takes a document as input and allows various plug-in algorithms, called Processing Resources, to be run in sequence with a final marked-up document output at the end (the plug-in framework is called CREOLE). The output desired for this project was the text marked-up with characteristics such as Paragraph, Sentence, Word, Space and for each Word there should have been the position in the document and type of word, e.g. noun, and also whether any letters were capitalised etc. We investigated various Pipelines and plug-ins to achieve the desired output. Despite the good quality output of the default Processing Resources and the ANNIE framework of plug-ins we settled on using the Stanford Parser plug-in combined with a Regular Expression sentence-splitter and the default ANNIE Tokeniser. The Regular Expression sentence-splitter performed better than others dues to its categorisation of punctuation being part of the sentence. The ANNIE Tokeniser is the preferred method of locating Tokens (words, spaces and punctuations) by the Stanford Parser and we saw no reason not to use this plug-in. We investigated a number of different Processing Resources parsers such as the OpenNLP, OpenCalais, LingPipe, RASP and SUPPLE. All these parsers were found not to work as effectively for our needs as the Stanford Parser for various reasons such as having a requirement to be unlocked with a licence key and requiring specific executables to be in specific folders at runtime. We found Stanford to be the only one that worked without much tweaking and to our needs.

The Stanford Parser output is a Syntax Tree Node and Dependency relationship. This parsing of the text creates a tree-like representation of the sentence with the

ROOT (or Sentence) at the top and branches that divide below depending on the clauses, phrases and words and their roles in the sentence. An example tree parse is shown below:

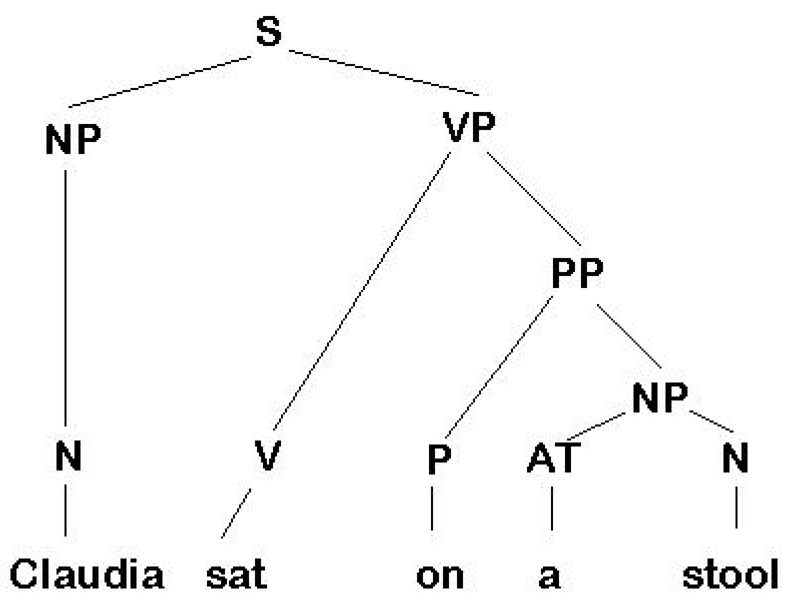


Figure 6.4.1 An example parse tree [80]

This hierarchy is achieved in XML by a dependency chain being created between entities at each level of the hierarchy. Only a small part of the output of the Stanford Parser is used in this project to determine where words are related to one another so that rhetorical forms can be shown, however this output is very powerful and could be used for much more.

The automated output from GATE specified in this solution is stored as an XML file (titled gateprocessing_hash.xml – where hash is the specific hash code of the input text).

4.4 Populating the Knowledge Base

Armed with the GATE XML data we next set about creating our knowledge base. This was done using the Protégé-OWL API which enabled the creation of a virtual knowledge base initially consisting of OWL classes and properties. The API is pointed to the OWL ontology - LassoingRhetoric.owl – for validation of the OWL entities. Step by step the GATE output is analysed and instances are created for the various classes and properties from the ontology. This occurs entirely in memory.

At the stage where all the paragraphs, sentences, spaces and words with their associated properties such as **hasWord** or **hasFirstSentence** are added based on the GATE output (thereby the input text) we started on the creation of more knowledge based on external data. By reference to the ontologies created earlier of RhetoricalDevices and VerbNounCombo we were able to scan the text and add properties for the **MinusComparison**, **PlusComparison EpitheticalName** and **SuperfluousWord** classes.

We were also able to scan the text for three adjacent words with the same initial letter. Since this might be a signal for the Alliteration rhetorical device we add the **hasFirstCharacter** datatype property for relevant instances of the word class.

The next step was to refer to external knowledge bases not previously created by us. The British National Corpus is an online resource of word frequencies derived from thousands of source texts. We were able to remotely query this resource with a particular combination of noun and verb words selected from our text in order to detect previous usage of the combination. If the British national Corpus had no record for the query then we were able to add the UnusualNounInCombo and UnusualVerbInCombo properties to our knowledge base for the two word instances in question.

A further Linked Data analysis was undertaken based on the DBPedia online resource. The rhetorical form of Historic Present can be signalled by a present tense reference to a person no longer living. By querying DBPedia dynamically with the name of the individual from the text we were able to determine whether firstly they were a person at all i.e. did they have a birthdate property and also are they alive, i.e. did they have a deathdate property. The resulting records were then used to populate the knowledge base with the Person class as a relation of the word instance being considered.

A further example of a Linked Data query we were able to produce was the information extracted from WordNet. WordNet is another Linked Data resource, but related to the English language entities and their synonyms, antonyms and various other relationships that words have. As far as we could determine there is no WordNet SPARQL interface, but we were able to query their online XML resource. It’s a very large resource and we only were able to use a tiny fraction of its size, but whenever we encountered a verb word in the input text we were able to query the online presence to have returned the synsetID. The synsetID is a unique reference to the group of synonyms to which the particular word belongs, for example the verb to have belongs to the synonym set of “have, have got, hold. The same word can also belong to different synonym sets, however for our purposes it was enough to be able to return a single synonym set and then query a local copy of Extended WordNet [81] to furnish the application with the synonyms of the word in question.

Various other data sources on the Semantic Web were investigated for this project, however not many of them provide the kinds of information easily obtainable that would benefit this simple knowledge base we were creating. Sources such as SUMO, YAGO, Cyc and Freebase provide excellent resources, but without having undertaken further research into this problem it was not possible to use them as resources at that stage.

After all these various analyses the virtual knowledge base was complete for our purposes. It was then possible to execute the SWRL rules dynamically. Our program was enabled so that the report file that was output would contain descriptions of each rhetorical pattern located. The entire knowledge base was then output to an RDF file which contained all the original ontology data (from LassointRhetoric.owl) and also all the new individuals populated from the processing. Crucially this RDF file was to be uploaded to an online location.

4.5 User Interface

We like the OpenCalais approach to displaying annotated text [82] so we tried to emulate this with our own annotations used instead. Creating a page that replicated the OpenCalais approach was straightforward in that the input box is a straightforward html text field that is submitted via an HTTP request to the server and then translated into the input text for the Java classes to process.

Once we have the knowledge base populated and the SWRL rules have run we were able to cycle through each inferred axiom and insert it into the text with some HTML and Javascript that enabled us to display the original text together with coloured spans showing the rhetorical patterns discovered. This is the output viewable on screen and the user can change the checkboxes to view any or all of the discovered rhetoric.

Chapter 5 - Implementation

In the initial phase of this project we weren't sure how to implement the design, but it soon became obvious that since GATE and Protégé-OWL both offered extensive Java APIs that we should follow this lead and develop a Java application. This enabled us to link directly to these APIs and take advantage of the functionality already present. Java is a computationally-complete language which was important and is also free to program and implement which suited our needs.

The overall architecture of the Java project is shown in Figure 5.1 and a more detailed diagram showing the interactions with the Java classes is shown in Figure 5.2.

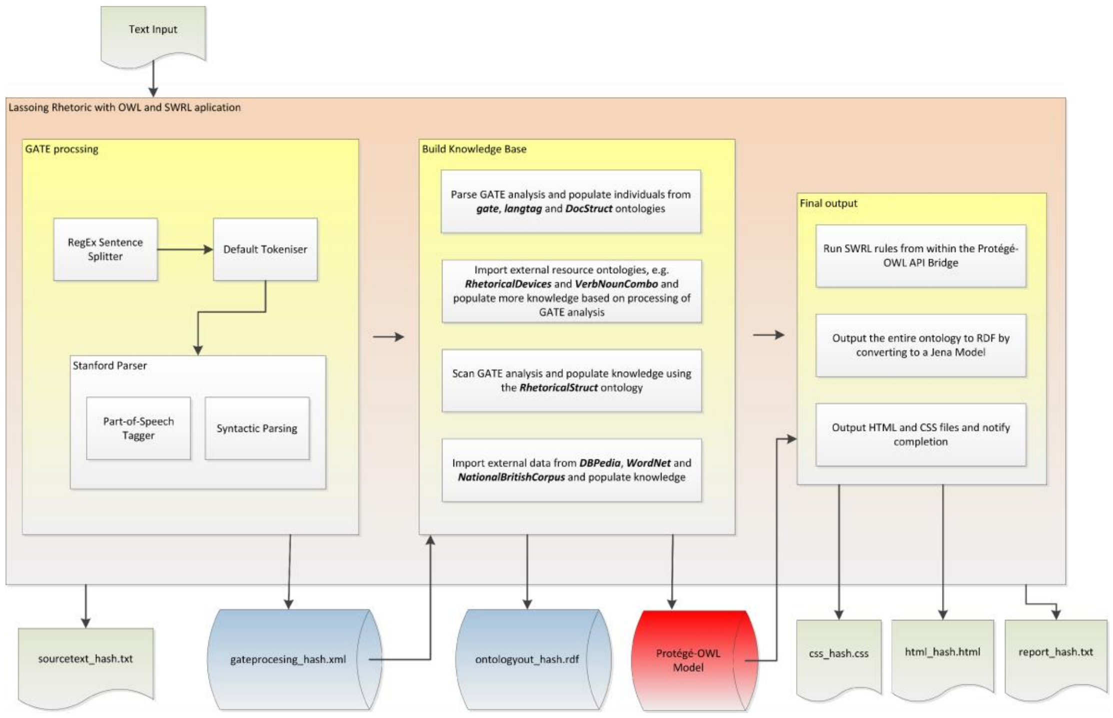

Text Input
Lassoing Rhetoric with OWL and SWRL aplication
GATE procssing
RegEx Sentence Splitter
Default Tokeniser
Stanford Parser
Part-of-Speech Tagger
Syntactic Parsing
Build Knowledge Base
Parse GATE analysis and populate individuals from gate, langtag and DocStruct ontologies
Import external resource ontologies, e.g. RhetoricalDevices and VerbNounCombo and populate more knowledge based on processing of GATE analysis
Scan GATE analysis and populate knowledge using the RhetoricalStruct ontology
Import external data from DBPedia, WordNet and NationalBritishCorpus and populate knowledge
Final output
Run SWRL rules from within the Protégé-OWL API Bridge
Output the entire ontology to RDF by converting to a Jena Model
Output HTML and CSS files and notify completion
sourcetext_hash.txt
gateprocesing_hash.xml
ontologyout_hash.rdf
Protégé-OWL Model
css_hash.css
html_hash.html
report_hash.txt

Figure 5.1 – The overall Java architecture

There are a number of methods of creating a Java program from basic text files coupled with a compiler to an IDE which manages many of the variables required to design, write and compile a project. We chose NetBeans 6.9 which is a mature IDE and is available under a GNU General Public Licence.

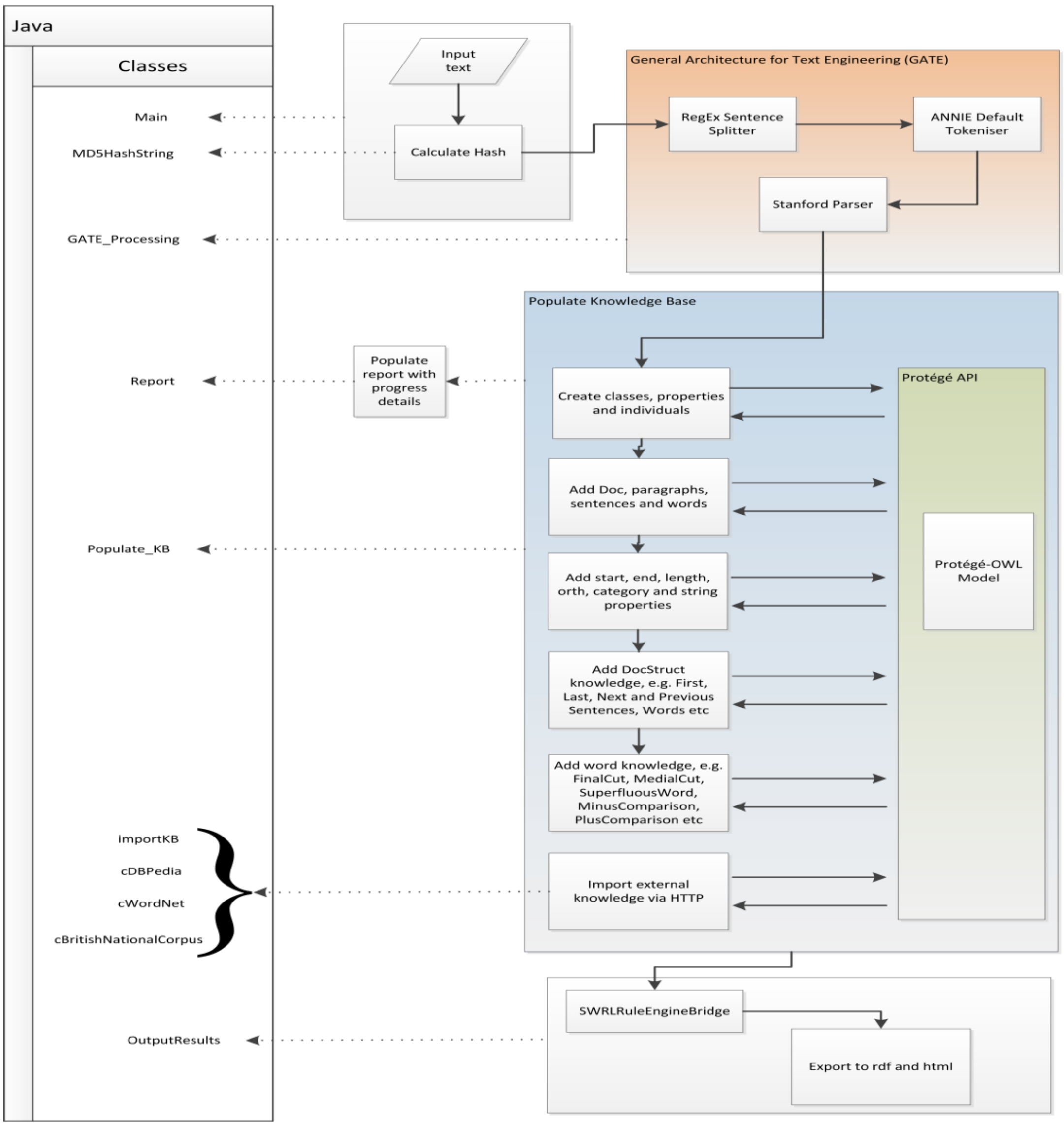

Java
Classes
Main
MD5HashString
GATE_Processing
Report
Populate_KB
importKB
cDBPedia
cWordNet
cBritishNationalCorpus
OutputResults
Input text
Calculate Hash
General Architecture for Text Engineering (GATE)
RegEx Sentence Splitter
ANNIE Default Tokeniser
Stanford Parser
Populate Knowledge Base
Populate report with progress details
Create classes, properties and individuals
Add Doc, paragraphs, sentences and words
Add start, end, length, orth, category and string properties
Add DocStruct knowledge, e.g. First, Last, Next and Previous Sentences, Words etc
Add word knowledge, e.g. FinalCut, MedialCut, SuperfluousWord, MinusComparison, PlusComparison etc
Import external knowledge via HTTP
Protégé API
Protégé-OWL Model
SWRLRuleEngineBridge
Export to rdf and html

Figure 5.2 Algorithmic representation with reference to Java classes

5.1 GATE API

We began by investigating the APIs of GATE and Protégé to understand the best implementation. The APIs were fairly straightforward even if the terminology was quite technical, but when implementing them it took us a long time to develop a working solution. The key feature for us from the GATE API is that a Pipeline can be designed in the GUI and then saved as a gapp XML file that describes the Pipeline in enough detail so that the file can then be imported programmatically via the API without the need to create Java bespoke routines to do the same thing. This saved a lot of effort and in the end we decided to use the Stanford Parser-provided gapp file called sample_parser_en.gapp. This file provided a Regular Expression sentence splitter, default tokeniser and the Stanford Parser algorithm too with the following settings (amongst others):

1. addPOSTags=true - this setting ensured that the parser added Part-of-Speech tags for each Token
2. addDependencyFeatures=true – this setting ensure that parse tree dependencies were included in the output
3. addDependencyFeatures=true – this setting adds dependency details to the Token elements in the output

To implement the GATE algorithms in a Java NetBeans project the Java Archives must be imported to the project. The jar files are supplied with the standard GATE download so it was a simple task to achieve the import. Once the gapp file has been specified it's a fairly straightforward task within the GATE API to produce the XML output. Initially a **CorpusController** class is instantiated and then a **Corpus** object is passed as a parameter in the **setCorpus()** method. A Document object of the file containing the input text is passed via the **add()** method and then the **execute()** method is called. It's then just a case of streaming the result out to an XML file via the FileOutputStream class and clearing up the objects.

5.2 Protégé API

The Protégé-OWL API is similarly straightforward. In our case it is a wrapper for the Jena OWL Model created by the **createJenaOWLModelFromURI()** method which takes the URL of the OWL ontology as a parameter. In our case this is the LassoingRhetoric.owl ontology since it contains all our entities and logic either through imported OWL ontologies, for example, or the SWRL rules. After adding the relevant jar files to the project we next had to create an instance of the **OWLNamedClass** class for each OWL class that was to be used in the knowledge base, e.g. gate:word. The same was required for object and datatype properties via the **OWLObjectProperty** and **OWLDataTypeProperty** classes respectively.

Next the GATE output was scanned many times to populate the OWL Model with instances – as **RDFIndividual** objects – and the concomitant properties – via the **addPropertyValue()** method that were discovered by manual manipulation of the

XML via the Java code. Some characteristics were represented by the rdf:type predicate rather than OWL properties – this was done using the **addRDFType()** method of the **RDFIndividual** object.

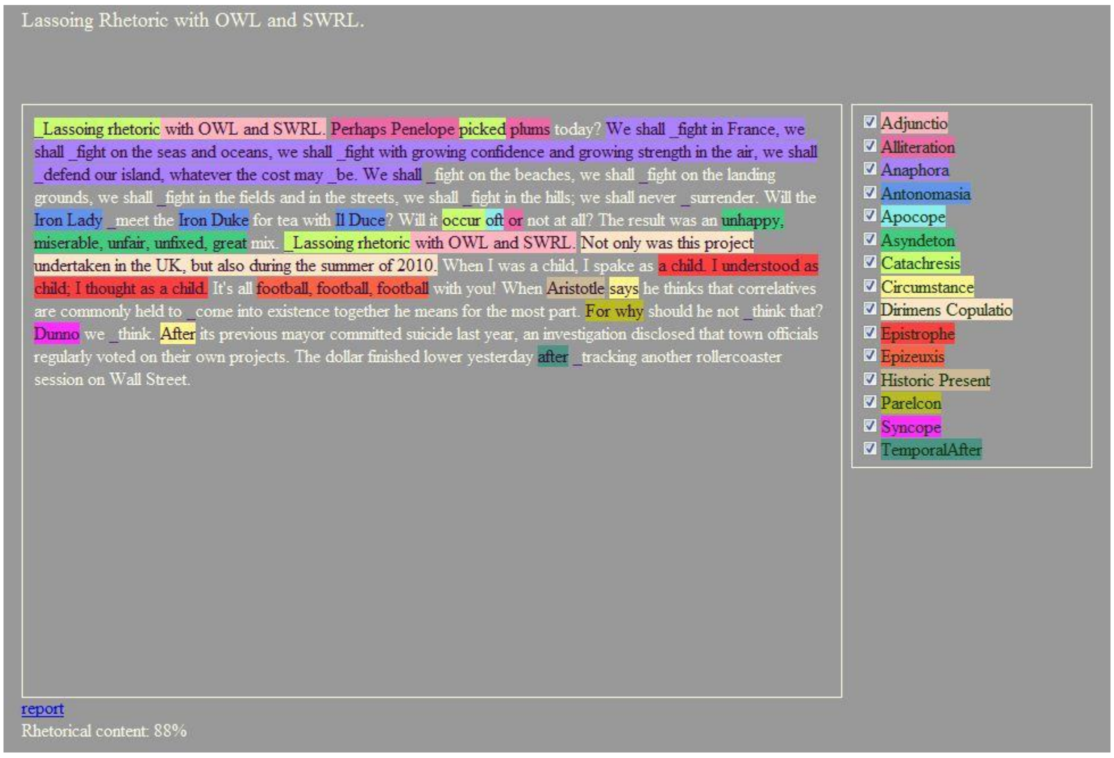
The decision to represent some characteristics as properties and some as rdf:types was based on both the nature of the characteristic, but also the use to which the characteristic would be put. For example we decided to represent categories of words such as Noun and Verb within the LangTag ontology as classes rather than have a single class of, for example, LangTag which would have many instances with which to relate to via object properties. We made this choice because of the nature of the hierarchy of classes, i.e. there are many types of verb which are all sub classes of the Verb class. This cannot be represented easily as an OWL property relationship.

Finally a SWRLRulEngineBridge class was instantiated for the OWL Model which then enabled reasoning to be undertaken with the **importsSWRLRulesAndKnowledge()** and **run()** methods. The final output to XML was handled by the **getJenaModel()** and **write()** methods.

After the Bridge has run it was possible to access the Inferred Axioms via the **getInferredAxioms()** method. These were cycled through and sorted in order to populate an array that goes on to form the final output.

5.3 HTML output

The final output is an HTML page containing the original input text and a set of checkboxes each referring to a particular rhetorical pattern that was discovered. When the box is checked then the appropriate spans of text that contain the rhetorical form are highlighted in the same colour as the checkbox – see figure 5.3.1.


Lassoing Rhetoric with OWL and SWRL.

_Lassoing rhetoric with OWL and SWRL. Perhaps Penelope picked plums today? We shall _fight in France, we shall _fight on the seas and oceans, we shall _fight with growing confidence and growing strength in the air, we shall _defend our island, whatever the cost may _be. We shall _fight on the beaches, we shall _fight on the landing grounds, we shall _fight in the fields and in the streets, we shall _fight in the hills; we shall never _surrender. Will the Iron Lady _meet the Iron Duke for tea with Il Duce? Will it occur oft or not at all? The result was an unhappy, miserable, unfair, unfixed, great mix. _Lassoing rhetoric with OWL and SWRL. Not only was this project undertaken in the UK, but also during the summer of 2010. When I was a child, I spake as a child. I understood as child; I thought as a child. It's all football, football, football with you! When Aristotle says he thinks that correlatives are commonly held to _come into existence together he means for the most part. For why should he not _think that? Dunno we _think. After its previous mayor committed suicide last year, an investigation disclosed that town officials regularly voted on their own projects. The dollar finished lower yesterday after _tracking another rollercoaster session on Wall Street.

- ✓ Adjunctio
- ✓ Alliteration
- ✓ Anaphora
- ✓ Antonomasia
- ✓ Apocope
- ✓ Asyndeton
- ✓ Catachresis
- ✓ Circumstance
- ✓ Dirimens Copulatio
- ✓ Epistrophe
- ✓ Epizeuxis
- ✓ Historic Present
- ✓ Parelcon
- ✓ Syncope
- ✓ TemporalAfter

report

Rhetorical content: 88%

Figure 5.3.1 screen shot of LaROS output

The HTML is generated as a text string piece by piece within the Java code. The Inferred Axioms are cycled through and used to create the checkboxes which link up via Javascript to various HTML span tags that contain the background-coloured rhetorical forms, also generated via cycling through the inferred axioms. The completed HTML string is output via a PrintWriter class to the html_hash.html file.

At the same time a css file is generated and saved as css_hash.css in the same folder as the html file.

5.4 Java class structure

Figures 5.4.1 and 5.4.2 show UML Class Diagram and the UML Sequence Diagram for this application. The application design did not follow strict Object-Oriented Programming principles since the task wasn’t actually appropriate for it. The overall task was more sequentially-based therefore a mixture of OOP and sequential-type programming was used. The main classes are GATE_Processing, PopulateKB and OutputResults which are controlled from the Main class.

The sequence diagram shows the various logical paths the program takes to produce the output.

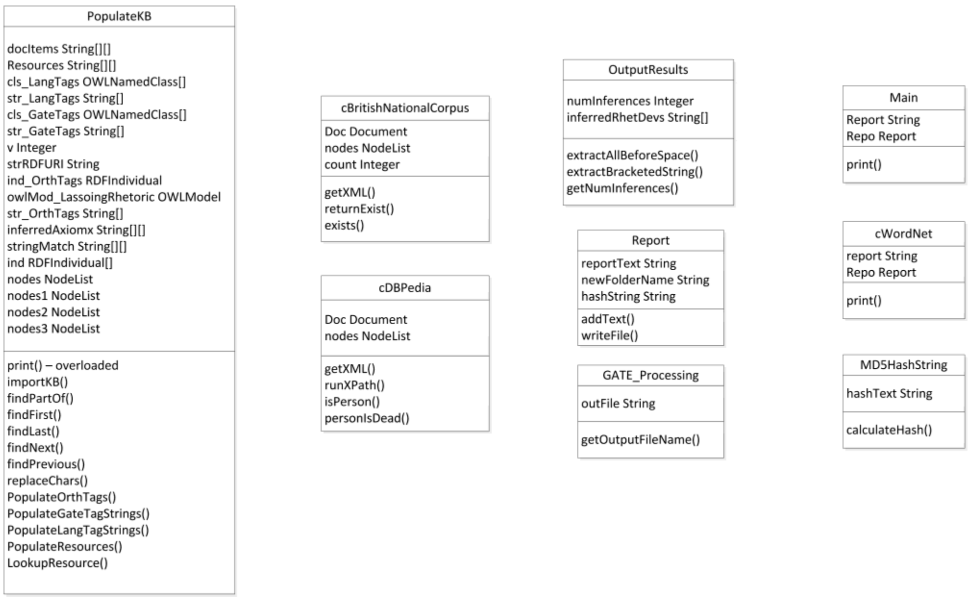


Figure 5.4.1 UML Class diagram

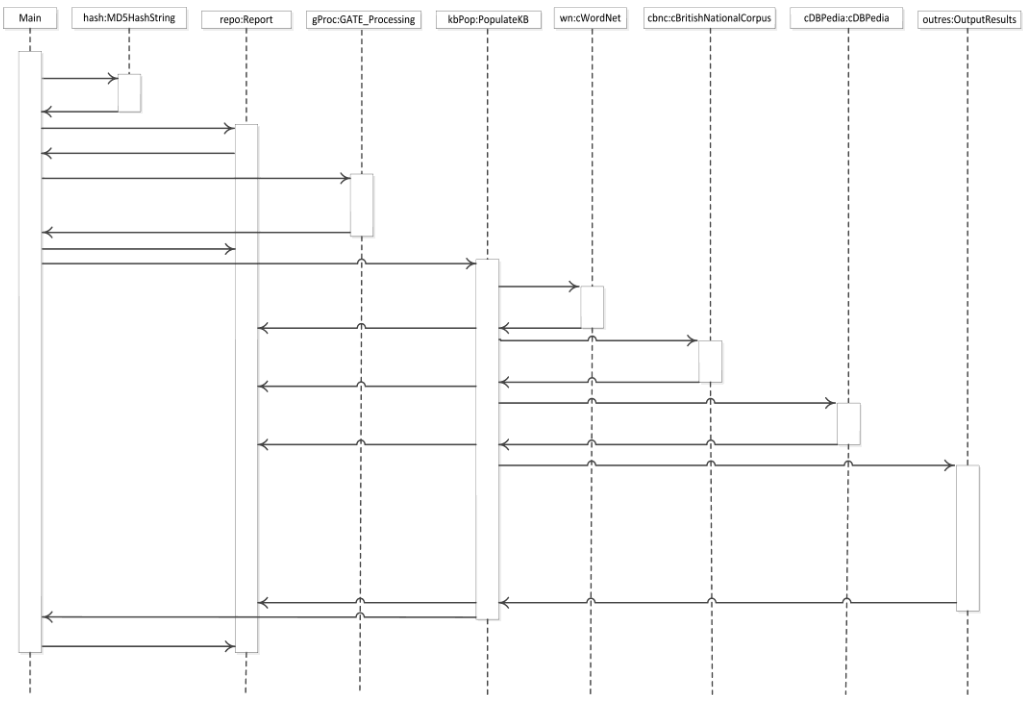


Figure 5.4.2 UML Sequence diagram

5.5 Data storage

There are a number of data storage requirements for this project.

1. the ontologies need to be stored and made accessible online
2. the various outputs from the applications such as the GATE processing and knowledge base should be held online after production
3. the application code needs to be stored and executable

Since the OWL ontologies are flat files storing them online is a simple task as long as an appropriate domain name and mechanism for Dereferencing can be undertaken in line with the Semantic Web principles. All our ontologies are stored online at our own domain of repositori.com. This enabled us to create folders and upload and amend easily our own files. These files should not be moved once published in order to maintain any applications built in the future that use them. It is proposed that this domain be a repository for these and future ontologies as they are developed in this domain and others.

We undertook an investigation of a variety of online storage and processing facilities. Initially we thought we would be able to purchase a dedicated server and have it connected online permanently in order to publish the application. This proved to be too expensive and not as flexible as have a virtual computing solution. We trialled Amazon’s Elastic Compute Cloud product for several months and found it to be easy to use, flexible, efficient and reasonably priced. We will be using Amazon’s Web Services in future for the development of this project and others.

Chapter 6 – Evaluation

6.1 Addressing objectives

This project’s main objective was to develop a computer application that could discover (“lasso”) rhetorical patterns of text using Semantic Web technologies. We have succeeded in achieving that objective. The over-riding objective, however, was to produce Meaning Representations in order to assist with the semantic analysis of natural language. This objective is more difficult to assess, but if we consider that a meaning representation can be thought of as simply a representation of a language element formulated in a particular meaning representation language then we can say that have succeeded in this objective too: First Order Logic is considered a meaning representation language [33] and SWRL (in Horn Logic terms) is a variant or First Order Logic. Our work in this project has produced a number of Horn Logic terms that describe natural language elements and in fact go further to enable inference on these elements to "discover" new knowledge, i.e. a particular rhetorical form whether it's a figure of speech or a Rhetorical Structure Theory relation.

Evaluating the proposal we can say that we did create a parsing solution which lead to the development of the knowledge base and that we organised the output along the lines of Semantic Web development. Going further we can say that the creation of the complex knowledge base of entities, properties and relationships for each text input and the fact that it is available online and accessible as part of the Semantic Web is another success.

Our ontologies that drive the knowledge base are all published online at domains we own and maintain. The SWRL rules designed to discover the textual forms mentioned previously are all available online in the Lassoing Rhetoric OWL ontology and in XML structure.

There were a number of problems and drawbacks, however which will be discussed later in this section.

To test the application we selected a string of text for each rule we wanted to verify. This text was selected deliberately, knowing that it should be discovered because it surely contained the rhetorical form searched for – in fact most of the text selections were from our rhetorical device handbook [40]. The full list of test texts is shown in Appendix 14. When entered into the application each test correctly showed the rhetorical form to be highlighted. We are happy that the rules described in this project do discover successfully the rhetorical patterns as intended. We also input various exceptional speeches from history that we thought would contain rhetorical content to view the results. Whilst not necessary to show that our process works it was interesting to view the output. In some cases we were not expecting to see rhetorical forms where we did. This shows a valuable benefit of this tool – it can highlight knowledge not previously known.

We must mention that the rules generated are not complete. It cannot be said that all instances of the rhetorical forms would be found by the rules designed herein. This is, however, expected. Natural language and rhetorical devices are complex forms of communication and we cannot expect to pinpoint all with simple logic rules.

We also assume that text input is well formed and correct, e.g. in order for our rule to find successfully the form of Catachresis we have to assume that the verb/noun combinations are intentional, e.g. shave grass / lasso rhetoric etc. Any mistaken text input could easily be misconstrued as rhetorical even though it is not. This is a complex issue, however. An example is the rhetorical device of Enallage. This works by deliberate mistakes being taken at face value for rhetorical effect, e.g. the use of the word “wot” instead of “what”.

Our major focus in this analysis has been on figures of speech. It was deemed appropriate to start with simpler forms of rhetoric before moving on to try and find more complex forms and also to search for RST relations. We have been successful in lassoing many kinds of rhetoric including RST although we have not had extensive time to extend the RST analysis and build on the discovery of two relation types only.

The development of a Rhetorical Content measure albeit simple and easily calculated is again another success for this project. It was an objective to have a repeatable, comparable measurement between input texts and the measurement we put in place completes the requirement capably.

6.2 Comparison

With regards to related works previously mentioned we will take each paper in turn for comparison with our research outcomes. There are numerous research papers in existence that have similar goals to ours. None that we have seen have used the same mechanisms as our project, however, but in principle could be said to be very similar.

Our approach is similar in parts to Bärenfänger et al [8] in that an ontological basis for language processing is undertaken. Further, logic rules (although in their case Prolog rather than SWRL) are used to annotate texts for Rhetorical Structure Theory forms. This again is similar to our approach. While we concentrate initially on the discovery of figures of speech this paper searches for the more complex RST relations. They do not show results as yet so we cannot compare the outputs. They also do not mention the adjunction of Semantic Web technology and Linked Data to add world knowledge whereas this is a central theme in our project.

Abulaish et al [1] focus their research on analysing text patterns with a view to adding knowledge via fuzzy matching. In order to facilitate the fuzzy matching an ontological approach is taken to managing knowledge of the domain (in this case a wine domain). Their results show significant success. Our project has distinct similarities with this paper in the knowledge discovery area. Our project utilises an ontological approach to adding new data to a knowledge to the knowledge base, e.g. words that are matched with our Rhetorical Devices ontology class of SuperfluousWord are deemed to have an unnecessary word (rhetorical device of Parelcon [40]) and are therefore tagged in the knowledge base as such and we can then derive the Parelcon rhetorical form based on this. Our approach is not fuzzy as in Abulaish at el's work, but the principle is the same.

In "Beyond String Matching and Cue Phrases: Improving Efficiency and Coverage in Discourse Analysis" [19], Corston-Oliver has a very similar objective to our project: to shed light on discourse analysis (strongly-linked to meaning representations) using RST relations. His approach is two-fold – to use Cue Phrases and syntactic analyses coupled with statistical analysis i.e. heuristic scoring. We have made much use of cue phrases in this project, but where our approach differs is that we are attempting to add world knowledge to the analysis rather than relying on cue phrases. This is no criticism of Corston-Oliver's paper since it is now12-years-old and pre-dates the Semantic Web by a few years and certainly even now we could argue that there is not enough information in data form in the Semantic Web and Linked Data domain to add significant world knowledge to analyses let alone in this earlier time. The principle stands, however, that cue phrase matching is crucial to language analysis and both his and our projects have had some success in this field.

David Reitter takes a different approach entirely in his paper from 2003 [51]. His pure statistical analyses (Vector Support Models) based on syntactic data attempt to discover RST relations. As he says "*Linguistic accounts for rhetorical structure have the luxury to assume the availability of world knowledge and inference in rhetorical analysis. Automated systems resort, with quite some success, to a combination of little linguistic knowledge and a lot of shallow parsing*", it would be fair for us to say that our project has made use of plenty of shallow parsing and a little linguistic knowledge. Where we differ, however, is that we are attempting to solve the problem of automating the introduction of world knowledge. We have shown that, even in a very rudimentary manner, we can import data dynamically and automatically from DBPedia and other Linked Data sources which can then provide world knowledge in order to add reasoning capabilities to our knowledge base.

The ultimate aim of this analysis is to disambiguate natural language such that a computer can formulate a meaning representation of it. It is our belief that we have gone some way towards that aim.

Overall in comparing our project with the related works it seems clear that the basis of language parsing and analysis coupled with rhetorical pattern discovery (in our case figures of speech mainly, but with some RST relations) is the same. Where this project differs is in attempting to address the issue of world knowledge inference. The related papers mention world knowledge, but often imply that the problem is too difficult to address.

6.3 Contribution

Since this project spans a number of domains the contribution it makes is (at least) two-fold. Firstly, in the field of linguistics the development of knowledge bases upon which to base language analyses is not new. Even the relatively new Semantic Web tools (e.g. OWL ontologies) have been used previously in his field. It is a powerful mechanism that enables effective investigation and persistency of results (via OWL ontology publication). We have taken the same view and utilised the GATE suite of algorithms and OWL ontologies. We feel that the outputs of the textual analysis are valuable because they add knowledge that was not previously available and in a format that is reusable and published online thereby enabling greater accessibility and expandability. Linguistic studies have even used logical reasoning previously. We take a similar approach, but have tried to stay within the Semantic Web domain by using SWRL. Our contribution then has been to analyse text and publish the resultant knowledge base online and as an input to the Semantic Web. It is possible for a resultant RDF file to be referenced in future studies in many ways including dynamic querying and expansion. We believe this to be something that has not been done often before in this domain.

Secondly, the process of adding world knowledge to this kind of analysis is not completely new. The mechanism that we have shown to work is based on sound computing premise – the Semantic Web. This resource is blossoming and with an appropriate framework in place can add knowledge to investigations of meaning that previously would not be possible to automate.

6.3.1 Benefits

The benefits of our work are that we have shown a mechanism to add world knowledge to linguistic analysis. We expect that this field will grow significantly as the Semantic Web grows.

6.3.2 Drawbacks

There are many drawbacks that we have discovered as we progressed through this project. Many of them are minor issues such as that a particular rhetorical pattern does not match our rule in all cases. For example our discovery of the rhetorical form of Historical Present searches online for data related to the individual person being references. Our knowledge base only searches for people with a single name, e.g. Aristotle or Socrates. Of course we could expand our processing to include more complex names, but it’s important to mention that in order to expand the forms of rhetorical discovered we sometimes were happy with a single rule that worked in a limited number of cases.

There are many other small issues that could be addressed (such as the fact that WordNet is based on American English!), but we feel that the principle of information discovery suffices in our goal rather than completeness of the solution.

We also must mention that while we believe GATE and Protégé to be excellent tools they work correctly only within certain tolerances. For example the statistical approaches to parsing employed by GATE plugins are not always 100% correct. This has an effect on the output and we have seen a number of words misclassified. This is rare, however, but does have an impact on the eventual output through our knowledge base since our processing relies on a correct analysis.

Chapter 7 – Conclusion / Future Work

We have taken a number of free software tools and developed our own ontological approach to building a knowledge base. With the knowledge base we have developed a suite of logic rules that has enabled us to infer the existence of rhetorical forms within text. This has been done before (although not, to our knowledge, by using SWRL). Further to this we have shown that by adding world knowledge, input from the Semantic Web and Linked Data, we are able to expand our knowledge base and also our capability to “lasso” rhetoric and various rhetorical patterns. Perhaps more importantly for our purpose we have shown that the development of Meaning Representations within our knowledge base by virtue of the Description Logic and Horn Logic entities (within our OWL ontologies and SWRL rules respectively) is possible with Semantic Web technologies and that the dynamic addition of world knowledge can increase the potential for Meaning Representation-based inference and analysis.

In our time computers cannot “understand” natural language. This is a significant problem that is being addressed in many ways. We argue in our project that the Semantic Web and Linked Data provide a mechanism to solve this problem. We believe that with the recent advances in computing technologies, specifically the Semantic Web that an old paradigm of knowledge base inference can be extended usefully to include world knowledge for the first time. We show that it is possible even if in a limited manner.

The data sources available online are growing rapidly, but at the moment are not intended for augmenting Meaning Representations and linguistic analyses. Over time we expect that more data will become available that is more easily integrated into this kind of processing of language and that perhaps greater ontological frameworks will be developed that will allow highly complex knowledge bases, rules and inference to work in this important and expanding domain.

7.1 Future Work

Since this work belongs to a relatively new field we expect that there are many future works that will bring benefits. We summarise some of them below.

1. Extension of search parameters: in this project we attempted to find rhetorical patterns. There is an equally important problem of discovering other language

forms, perhaps idioms or grammatical errors. This could be achieved in a similar way by using a knowledge base and logic rule system

2. Inadequacies of SWRL: Our project has also highlighted the potential drawbacks of the SWRL language. Whilst we have been able to prove our thesis we find that SWRL is limited in its potential due to the facts that it is not computationally complete, doesn’t support negated atoms [79[and also that it is not productive. We think that SWRL has a very useful position within the Semantic Web, however for our purposes we would prefer to use a different rule system – one that we haven’t seen yet. With the advent of the RIF framework we will expect to see more rule systems being developed for specific purposes. Our view is that a linguistic-based rule system that has the capabilities of SWRL, but also enables entity-creation and computational completeness, but crucially within the Semantic Web domain is required for future work in this area

3. In the area of performance we think that our solution is adequate. As the principle of querying online resources is developed further, however, it will be necessary to investigate performance and the overhead of sending many HTTP requests over the internet. A solution might be to use offline storage mirrors of some large resources, e.g. DBPedia.

4. Statistical linguistic analyses are mature and would prove beneficial to future work in this area by providing probability breakdowns for hard-to-decide algorithm outputs. For example trying to determine if a word is simply misspelt or is deliberately used in this way is very difficult. Humans find it easy to Code-Shift between dialects or languages, but a computing system cannot easily do this. A machine-learning approach would add significant benefits to this domain

References

[1] Abulaish, M. and Dey, L., (2004). Using Part-Of-Speech Patterns and Domain Ontology to Mine Imprecise Concepts from Text Documents. *Proceedings of the 6th International Conference on Information Integration and Web Based Applications and Services (iiWAS'04).* Jakarta, Indonesia. Sept. 2004.

[2] Amardeilh, F., Damljanovic, D., Bontcheva, K., (2009) CA Manager: a Framework for Creating Customised Workflows for Ontology Population and Semantic Annotation. *Proceedings of the Semantic Authoring, Annotation and Knowledge Markup Workshop (SAAKM 2009) co-located with the 5th International Conference on Knowledge Capture (K-CAP 2009)*. Redondo Beach, California, USA. September 2009.

[3] Amazon, (2010). *Amazon Web Services*. [online] Available from: http://aws.amazon.com/ [Accessed May 2010].

[4] Antoniou, G. and van Harmelen, F., (2008). *A Semantic Web Primer*. 2nd ed. Cambridge: MIT Press.

[5] Aristotle, (translated Freese, J. H.)., (2006). Art of Rhetoric. Cambridge: Harvard University Press.

[6] Asher, N. and Lascarides, A., (2003). *Logics of Conversation*. Cambridge: Cambridge University Press.

[7] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z., (2007). DBPedia: A Nucleus for a Web of Open Data. *Lecture Notes in Computer Science* 4825:722-735.

[8] Bärenfänger, M., Hilbert, M., Lobin, H., Lüngen, H., (2008). OWL ontologies as a Resource for Discourse Parsing. *LDV-Forum. GLDV-Journal for Computational Linguistics and Language Technology*. 23(1):17-26.

[9] Berners-Lee, T., Hendler, J., Lassila, O., (2001). The Semantic Web. [online]. *Scientific American*. Available from: < http://www.sciam.com/article.cfm?id=the-semantic-web> [Accessed April 2010].

[10] Bhole, A., Fortuna, B., Grobelnik, M., Mladenić, D., (2007). Extracting Named Entities and Relating Them over Time Based on Wikipedia. *Informatica (Ljubljana).* 31(4):463-468.

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

[12] Briscoe, E., Carroll, J., Watson, R., (2006). The Second Release of the RASP System. *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, Sydney, Australia.

[13] Carlson, L. and Marcu, D., (2001). Discourse tagging reference manual. *Technical report, Information Science Institute*, Marina del Rey, CA. ISI-TR-545.

[14] Carlson, L., Marcu, D., Okurowski, M., (2003). Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. *Text Speech and Language Technology*. 22:85-108.

[15] Champion, M., (2001). Storing XML in Databases. *eAI Journal*. 53-55.

[16] Chen, Z., Webster, J.J., Chow, I., Hao, T., (2009). Ontology Oriented Computation of English Verbs Metaphorical Trait. *Advances in Computational Linguistics. Research in Computing Science*. 41:135-142.

[17] Chung, S., Ahrens, K., Huang, C., (2004). Using WordNet and SUMO to Determine Source Domains of Conceptual Metaphors. In: Donhong, J., Teng, L.K., Hui, W. (Eds). *Recent Advancement in Chinese Lexical Semantics: Proceedings of 5th Chinese Lexical Semantics Workshop (CLSW-5).* Singapore: COLIPS. pp. 91-98.

[18] Corby, O., Kefi-Khelif, L., Cherfi, H., Gandon, F., Khelif, K., (2009). Querying the Semantic Web of Data using SPARQL RDF and XML. *Institut National De Recherche En Informatique Et En Automatique*. [online] Available from: http://hal.inria.fr/docs/00/36/23/81/PDF/RR-6847.pdf [Accessed April 2010].

[19] Corston-Oliver, S.H., (1998). Beyond String Matching and Cue Phrases: Improving Efficiency and Coverage in Discourse Analysis. *Technical Report – American Association for Artificial Intelligence* SS No. 06:9-15

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

[21] Cycorp, (2010). *OpenCyc Brings meaning to the Web.* [online] Available from: http://cyc.com/cyc/company/news/OpenCyc%20Brings%20Meaning%20to%20the%20Web [Accessed April 2010].

[22] Dimitrov, M., Bontcheva, K., Cunningham, H., Maynard, D., (2002). A Light-weight Approach to Coreference Resolution for Named Entities in Text. *Proceedings of the Fourth Discourse Anaphora and Anaphor Resolution Colloquium (DAARC)*, Lisbon, Portugal.

[23] Farrar, S., and Langendoen, T., (2003). A linguistic ontology for the semantic web. *Glot International*. 7(3):97-100.

[24] Farrar, S. and Lewis, W.D., (2007). The GOLD Community of Practice: an infrastructure for linguistic data on the Web. *Language Resources and Evaluation*. 41(1):45-60.

[25] Freebase, (2010). Freebase: *A Wealth of Free Data.* [online] Available from: http://www.freebase.com/ [Accessed April 2010].

[26] Galanis, D. and Androutsopoulos, I., (2007). Generating Multilingual Descriptions from Linguistically Annotated OWL Ontologies: the NaturalOWL System. *Proceedings of the 11th European Workshop on Natural Language Generation (ENLG 2007),* Schloss Dagstuhl, Germany.

[27] Giles, G., (2005). Internet encyclopaedias go head to head. *Nature*, 438:900-901.

[28] Goecke, D., Lüngen, H., Sasaki, F., Witt, A., and Farrar, S., (2005). GOLD and discourse: Domain and community-specific extensions*. In Proceedings of the 2005 E-MELD-Workshop*, Boston, MA.

[29] *GOLD Community* [online] Available from: http://linguistics-ontology.org/ [Accessed April 2010].

[30] Hovy, E. and Maier, E., (1995). Parsimonious or profligate: How many and which discourse structure relations? *Unpublished paper* [online] Available from: http://www.isi.edu/natural-language/people/hovy/publications.html. [Accessed April 2010].

[31] Institute for System Programming, (2010). *Sedna*. [online] Available from: http://modis.ispras.ru/sedna/ [Accessed May 2010].

[32] Jiang, X. and Tan, A., (2009) CRCTOL: A Semantic-Based Domain Ontology Learning System. *Journal of the American Society for Information Science and Technology*. 6(1):150-168

[33] Jurafsky, D. and Martin, J., (2009). *Speech and Language Processing*. 2nd ed. New Jersey: Pearson Education Inc.

[34] Kaljurand, K. and Fuchs, N., (2006). Bidirectional mapping between OWL DL and Attempto Controlled English. *Principles and Practice of Semantic Web Reasoning*. 4187:179-189.

[35] Kamp, H. and Reyle, U., (1993). From Discourse To Logic. Introduction to Modeltheoretic Semantics of Natural Language, *Formal Logic and Discourse Representation Theory*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

[36] Kasneci, G., Ramanath, M., Suchanek, F., Weikum, G., (2008). The YAGO-NAGA Approach to Knowledge Discovery. *SIGMOD Record*. 37(4):41-47.

[37] Kehler, A., Kertz, L., Rohde, H., Elman, J., (2008). Coherence and Coreference Revisited. *Journal of Semantics*. 25(1):1-44.

[38] Köhler, J., Philippi, S., Specht, M., Rüegg, A., (2006). Ontology based text indexing and querying for the semantic web. *Knowledge-Based Systems*. 19(8):744-754.

[39] Knott, A. and Dale, R. (1994) Using linguistic phenomena to motivate a set of coherence relations. *Discourse Processes*. 18(1), 35-62

[40] Lanham, R.A., (1991). *A Handlist of Rhetorical Terms*. Berkeley: University of California Press.

[41] Lüngen, H., (2008). RRSet – Taxonomy of rhetorical relations in SemDok. *Interner Bericht, Justus-Liebig-Universität Gießen, Fachgebiet Angewandte Sprachwissenschaft und Computerlinguistik* Available from: http://samba.germanistik.uni-giessen.de/~semdok/resources/FG-Report-RRSet.pdf

[42] Mann, W. C. and Thompson, S. A. (1988). Rhetorical Structure Theory: Toward a functional theory of text organisation. *Text*, 8(3):243–281.

[43] Martin Heidegger, (translated Macquarrie, J. And Robinson, E.)., (2009). *Being and Time*, London: Blackwell Publishing.

[44] Medelyan, O., Milne, D., Legg, C., Witten, I.H., (2009). Mining Meaning from Wikipedia. *International Journal of Human Computer Studies*. 67:716-754.

[45] Microsoft, (2010). *Windows Azure Platform*. [online] Available from: http://www.microsoft.com/windowsazure/ [Accessed May 2010].

[46] Mieder, W., (2009*). "Yes we can": Barack Obama's proverbial rhetoric*. New York: Peter Lang Publishing.

[47] Milne, D. and Witten, I.H., (2008). Learning to Link with Wikipedia. *Proceeding of the 17th ACM conference on Information and knowledge management*, Napa Valley, California, USA. October 2008. Vol 2:982-991.

[48] Missikoff, M., Velardi, P., Fabriani, P., (2003). Text Mining Techniques to Automatically Enrich a Domain Ontology. *Applied Intelligence*. 18(3):323-340.

[49] *OpenCCG*, (2010). [online] Available from: http://openccg.sourceforge.net/index.html [Accessed April 2010].

[50] Princeton University, (2010). *WordNet: A Lexical database for English*. [online] Available from: http://wordnet.princeton.edu/ [Accessed March 2010].

[51] Reitter, D., (2003). Simple Signals for Complex Rhetorics: On Rhetorical Analysis with Rich-Feature Support Vector Models*. Journal for Computational Linguistics Language Technology*. 18(1/2):38-52.

[52] Reitter, D. and Stede, M., (2003). Step by step: underspecified markup in incremental rhetorical analysis. *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03) (at EACL 2003)*, Budapest, 2003.

[53] Seligman, L. and Rosenthal, A., (2001). XML's Impact on Databases and Data Sharing. *IEEE Computer*. 34(6):59-67.

[54] Shinmori, A., Okumura, M., Marukawa, Y., Iwayama, M., (2002) Rhetorical Structure Analysis of Japanese Patent Claims Using Cue Phrases. *Proceedings of the Third NTRCIR Workshop*. Sep. 2001-Oct. 2002.

[55] Simon Fraser University, (2010). *RST Rhetorical Structure Theory* [online] Available from: http://www.sfu.ca/rst/ [Accessed April 2010]

[56] Sporleder, C., Lascarides, A., (2007). Exploring Linguistic Cues to Classify Rhetorical Relations. *Amsterdam Studies in the Theory and History of Linguistic Science Series 4*. Vol 292:157-166.

[57] Sporleder, C. And Lascarides, A. (2008). Using Automatically Labelled Examples to Classify Rhetorical Relations: An Assessment. *Natural Language Engineering*, 14(3):369-416.

[58] Stanford University. *The Stanford Parser: A statistical parser*. [online] Available from: http://www-nlp.stanford.edu/software/lex-parser.shtml [Accessed April 2010].

[59] Stanford University, (2010). *Welcome to Protégé*. [online] Available from: http://protege.stanford.edu/ [Accessed March 2010]

[60] Suchanek, F., Sozio, M., Weikum, G., (2009). SOFIE: A Self-Organising Framework for Information Extraction. *Proceedings of the 18th international conference on World Wide Web*. Madrid, Spain. April 2009. pp. 631-640.

[61] *Suggested Upper Merged Ontology (SUMO),* (2010). [online] Available from: http://www.ontologyportal.org/ [Accessed April 2010].

[62] TigerLogic, (2010). *TigerLogic XDMS*. [online] Available from: http://www.rainingdata.com/products/tl/index.html [Access May 2010].

[63] Tsoi, L.C., Patel, R., Zhao, W., Zheng, W.J., (2009) Text-mining approach to evaluate terms for ontology development*. Journal of Biomedical Informatics*. 42:824-830

[64] University of Tokyo, (2008). *Enju*. [online] Available from: http://www-tsujii.is.s.u-tokyo.ac.jp/enju/index.html [Accessed April 2010].

[65] Verbenne, A., (2007). Evaluating Answer Extraction for Why-QA using RST-annotated Wikipedia texts. In: Nurmi, A. and Sustretov, D. (editors). *Proceedings of the Twelfth ESSLLI Student Session*. Dublin, Ireland. August, 2007.

[66] Webber, B.L., (1983). So what can we talk about now?. Brady, M. And Berwick, R.C (Eds.), *Computational Models of Discourse*, pp. 331-371. The MIT Press. Reprinted in Grosz et al. (1986)

[67] Webber, B. L., Knott, A., Stone, M., and Joshi, A., (2003). Anaphora and discourse structure. *Computational Linguistics*. 29(4):545–588.

[68] World Wide Web Consortium, (2004). *SWRL: A Semantic Web Rule Language Combining OWL and RuleML* [online] Available from: http://www.w3.org/Submission/SWRL/ [Accessed April 2010].

[69] World Wide Web Consortium, (2006). *Defining N-ary Relations on the Semantic Web*. [online] Available from: http://www.w3.org/TR/swbp-n-aryRelations/ [Accessed April 2010].

[70] World Wide Web Consortium, (2006). *RDF/OWL Representation of WordNet*. [online] Available from: http://www.w3.org/TR/wordnet-rdf/ [Accessed April 2010].

[71] World Wide Web Consortium, (2010). *Semantic Web* [online] Available from: http://www.w3.org/standards/semanticweb/ [Accessed April 2010].

[72] Witte, R., Kappler, T., Baker, C.J.O., (). Ontology Design for Biomedical Text Mining. Semantic Web: *Revolutionizing Knowledge Discovery in the Life Sciences*, Chapter 13:281-313.

[73] World Wide Web Consortium, (2009). *W3C Semantic Web Frequently Asked Questions* [online] Available from http://www.w3.org/2001/sw/SW-FAQ#whgiantont [Accessed October 2010]

[74] Dong , Z., Dong, Q., (2006) *Hownet and the Computation of Meaning*. World Scientific Publishing Co Pte Ltd

[75] Hewlett-Packard. *Jena – A Semantic Web Framework for Java*, (2009). [online] Available from: http://jena.sourceforge.net/ {Accessed June 2010]

[76] Stanford University. *SWRLRuleEngineBridge FAQ*. [online] Available from: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineBridgeFAQ [Accessed June 2010]

[77] Tim Berners-Lee, World Wide Web Consortium, (20). *Linked Data* [online] Available from http://www.w3.org/DesignIssues/LinkedData.html [Accessed July 2010]

[78] Chalker, S. and Weiner, E., (1994). *The Oxford Dictionary of English Grammar.* London, BCA.

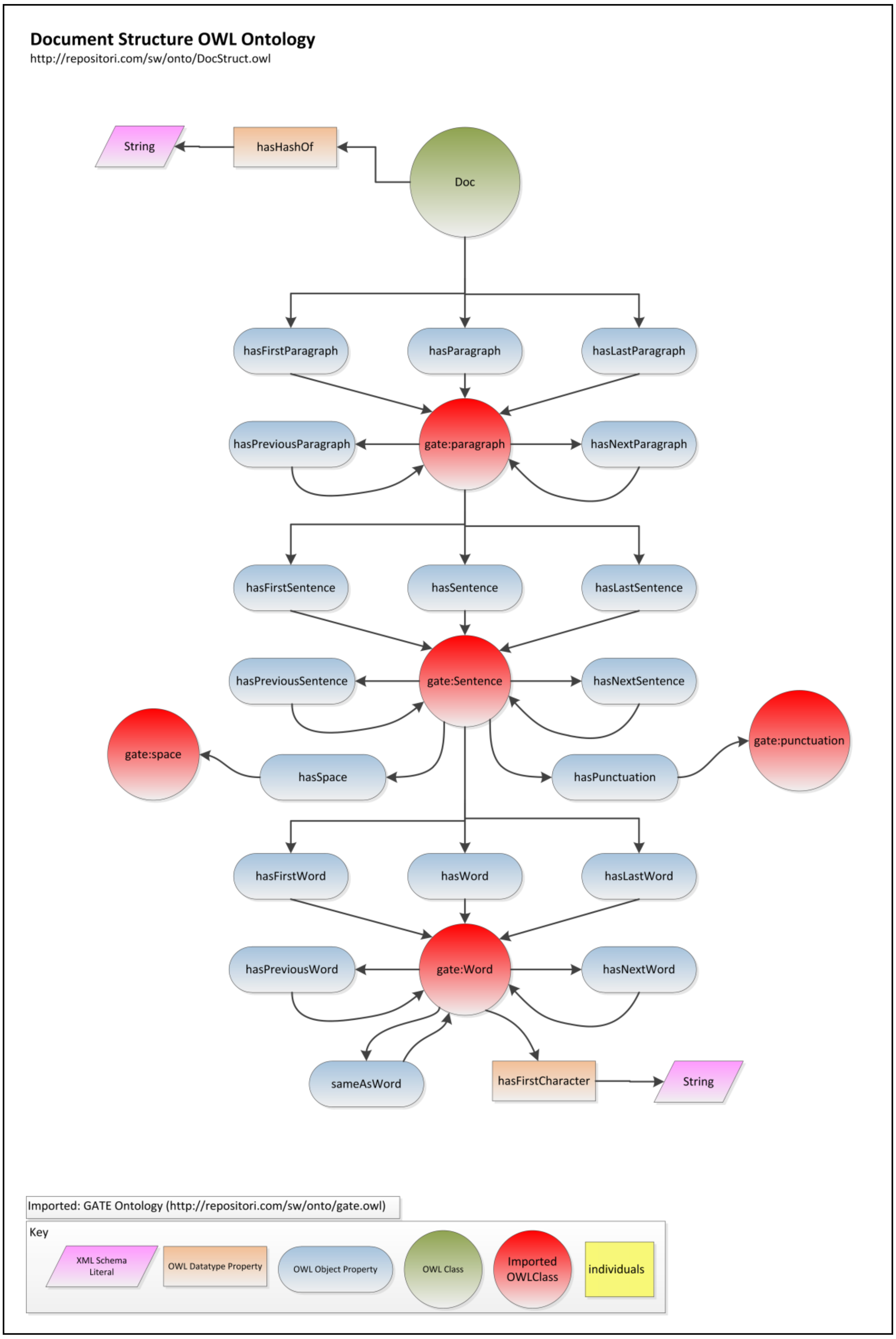
[79] Stanford University. *SWRL Language FAQ.* [online] Available from: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ [Accessed June 2010]

[80] Lancaster University. *Parsing*. [online] Available from: http://www.lancs.ac.uk/fss/courses/ling/corpus/Corpus2/2PARSE.HTM [Accessed October 2010]

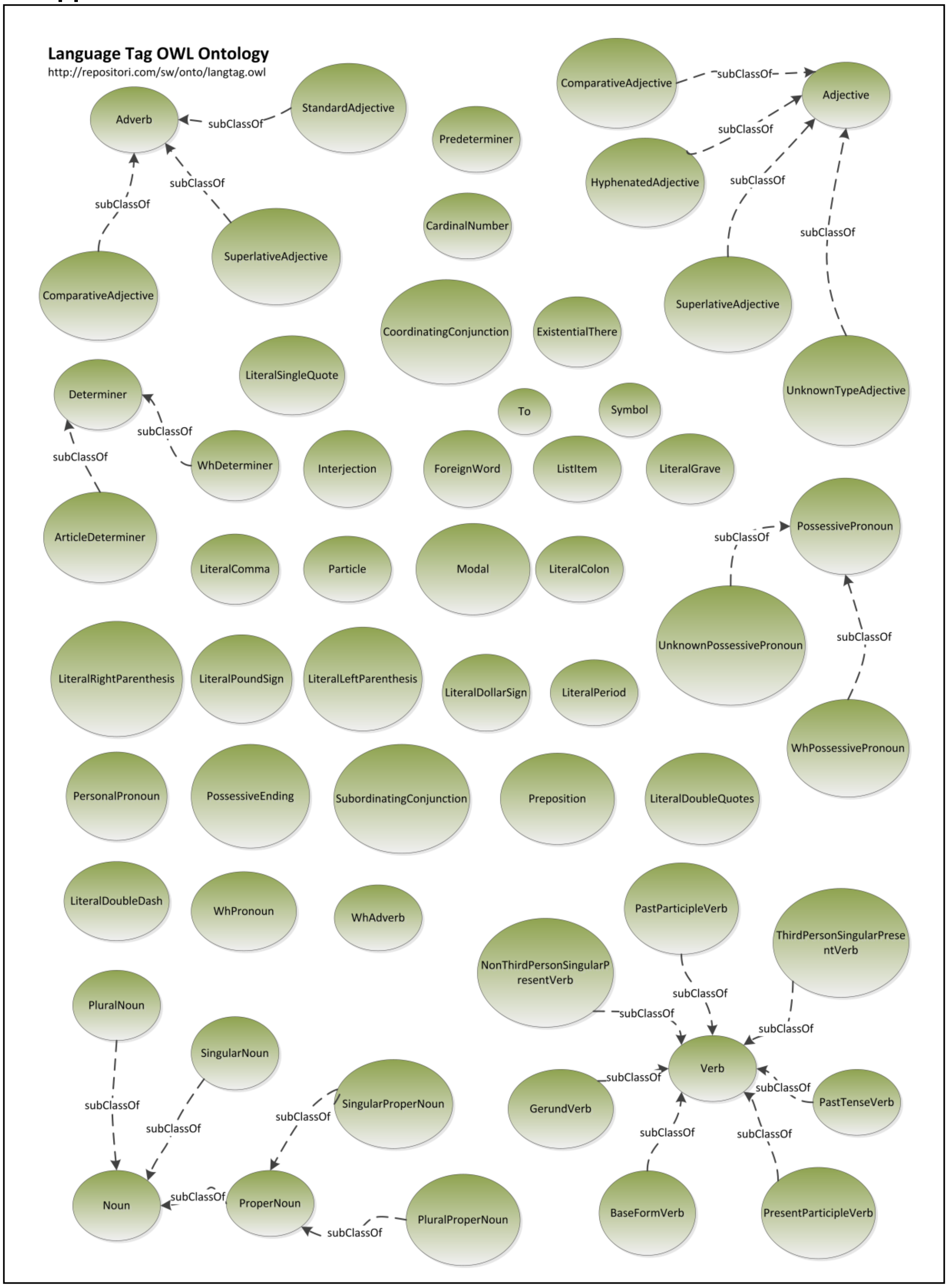
[81] University of Texas. *eXtended WordNet*. [online] Available from: http://xwn.hlt.utdallas.edu/ [Accessed September 2010]

[82] OpenCalais Document Viewer [online] Available from: http://viewer.opencalais.com/ [Accessed August 2010]

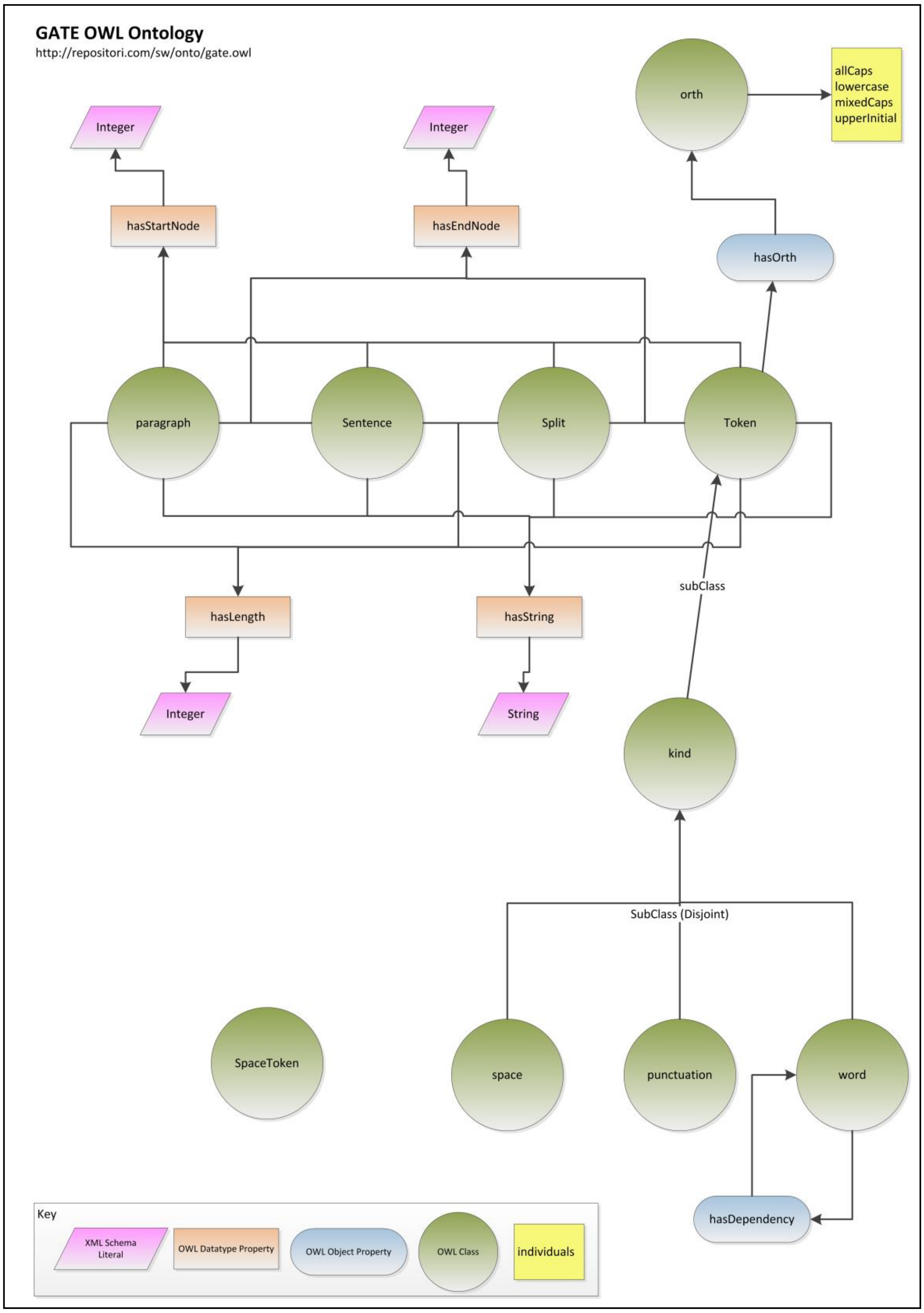
Appendix 1


Document Structure OWL Ontology
http://repositori.com/sw/onto/DocStruct.owl
String
hasHashOf
Doc
hasFirstParagraph
hasParagraph
hasLastParagraph
hasPreviousParagraph
gate:paragraph
hasNextParagraph
hasFirstSentence
hasSentence
hasLastSentence
hasPreviousSentence
gate:Sentence
hasNextSentence
gate:space
hasSpace
hasPunctuation
gate:punctuation
hasFirstWord
hasWord
hasLastWord
hasPreviousWord
gate:Word
hasNextWord
sameAsWord
hasFirstCharacter
String
Imported: GATE Ontology (http://repositori.com/sw/onto/gate.owl)
Key
XML Schema Literal
OWL Datatype Property
OWL Object Property
OWL Class
Imported OWLClass
individuals

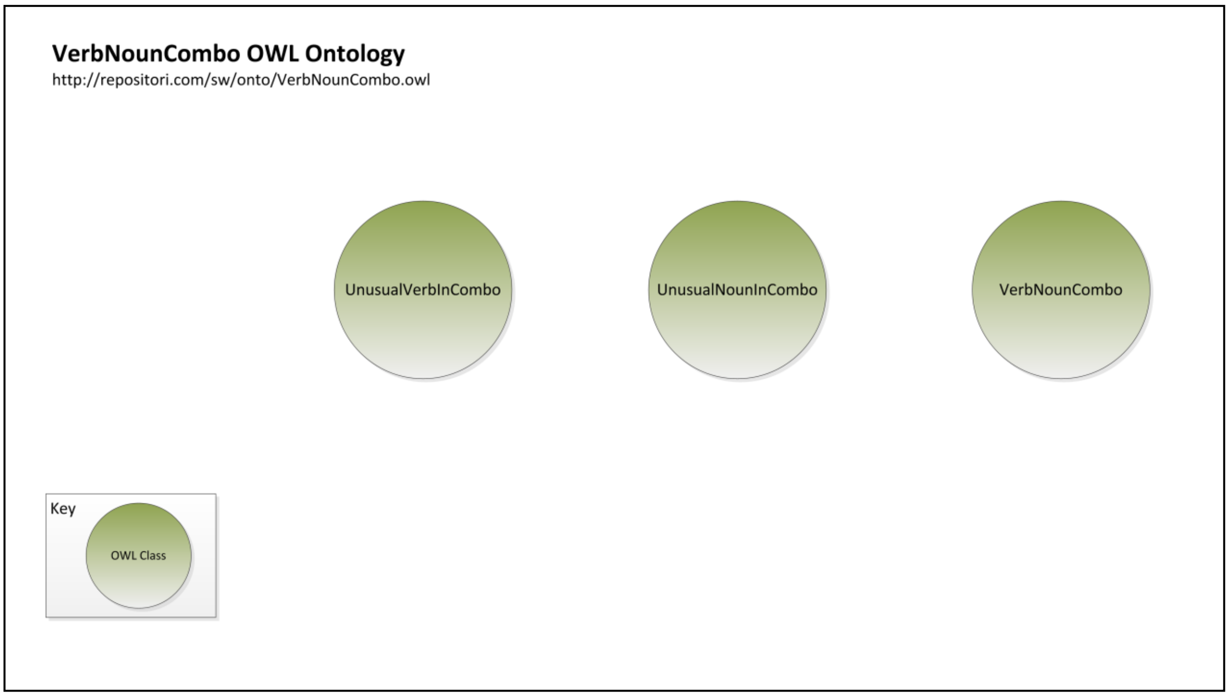
Appendix2


Language Tag OWL Ontology
http://repositori.com/sw/onto/langtag.owl
Adverb
StandardAdjective
ComparativeAdjective
SuperlativeAdjective
Predeterminer
CardinalNumber
ComparativeAdjective
Adjective
HyphenatedAdjective
SuperlativeAdjective
UnknownTypeAdjective
CoordinatingConjunction
ExistentialThere
LiteralSingleQuote
Determiner
WhDeterminer
ArticleDeterminer
To
Symbol
Interjection
ForeignWord
ListItem
LiteralGrave
PossessivePronoun
UnknownPossessivePronoun
WhPossessivePronoun
LiteralComma
Particle
Modal
LiteralColon
LiteralRightParenthesis
LiteralPoundSign
LiteralLeftParenthesis
LiteralDollarSign
LiteralPeriod
PersonalPronoun
PossessiveEnding
SubordinatingConjunction
Preposition
LiteralDoubleQuotes
LiteralDoubleDash
WhPronoun
WhAdverb
PastParticipleVerb
NonThirdPersonSingularPresentVerb
ThirdPersonSingularPresentVerb
Verb
GerundVerb
PastTenseVerb
BaseFormVerb
PresentParticipleVerb
PluralNoun
SingularNoun
SingularProperNoun
Noun
ProperNoun
PluralProperNoun
subClassOf

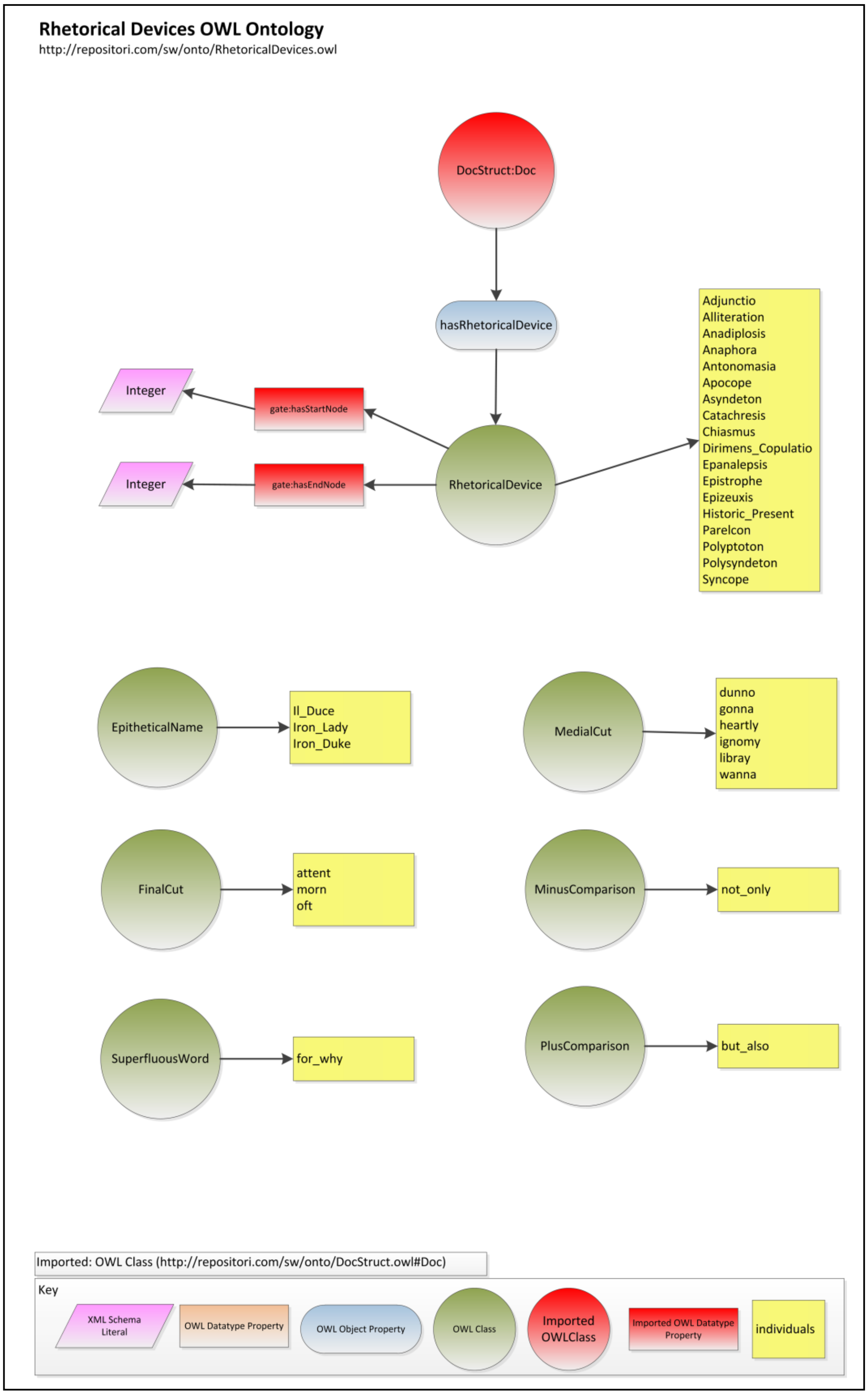
Appendix 3


GATE OWL Ontology
http://repositori.com/sw/onto/gate.owl
Integer
hasStartNode
Integer
hasEndNode
orth
allCaps
lowercase
mixedCaps
upperInitial
hasOrth
paragraph
Sentence
Split
Token
subClass
hasLength
hasString
Integer
String
kind
SubClass (Disjoint)
SpaceToken
space
punctuation
word
hasDependency
Key
XML Schema Literal
OWL Datatype Property
OWL Object Property
OWL Class
individuals

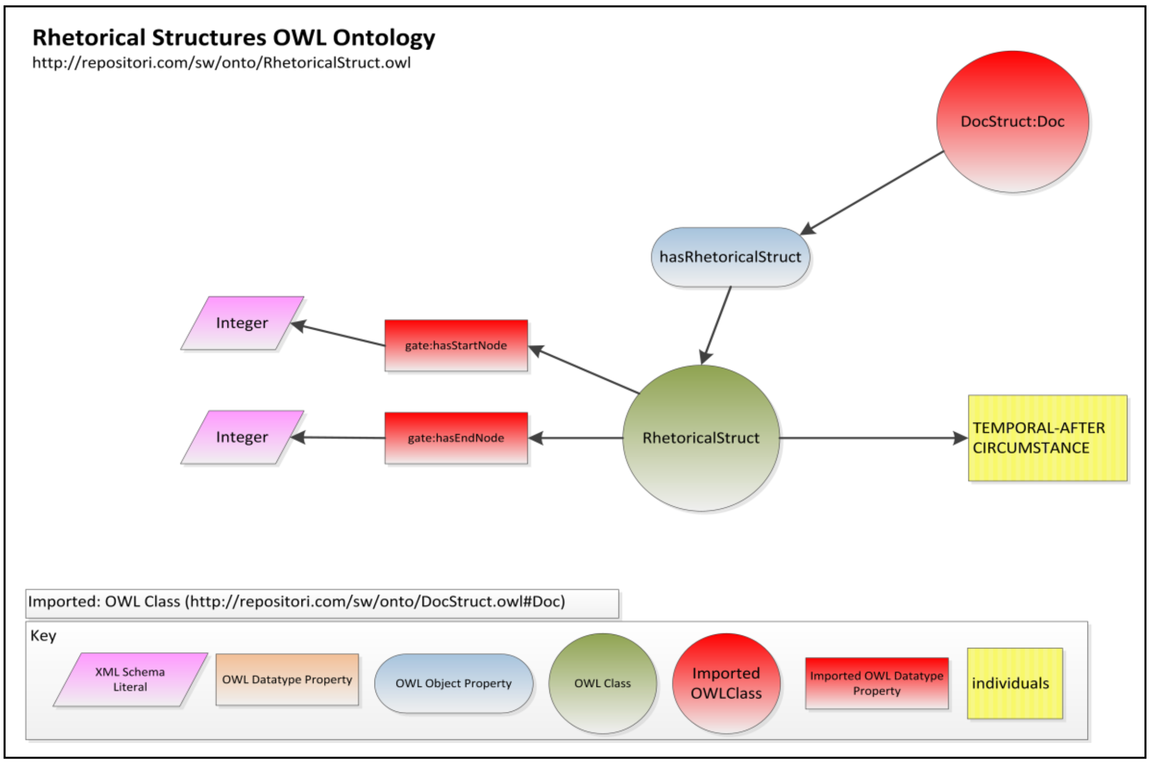
Appendix 4


VerbNounCombo OWL Ontology
http://repositori.com/sw/onto/VerbNounCombo.owl
UnusualVerbInCombo
UnusualNounInCombo
VerbNounCombo
Key
OWL Class

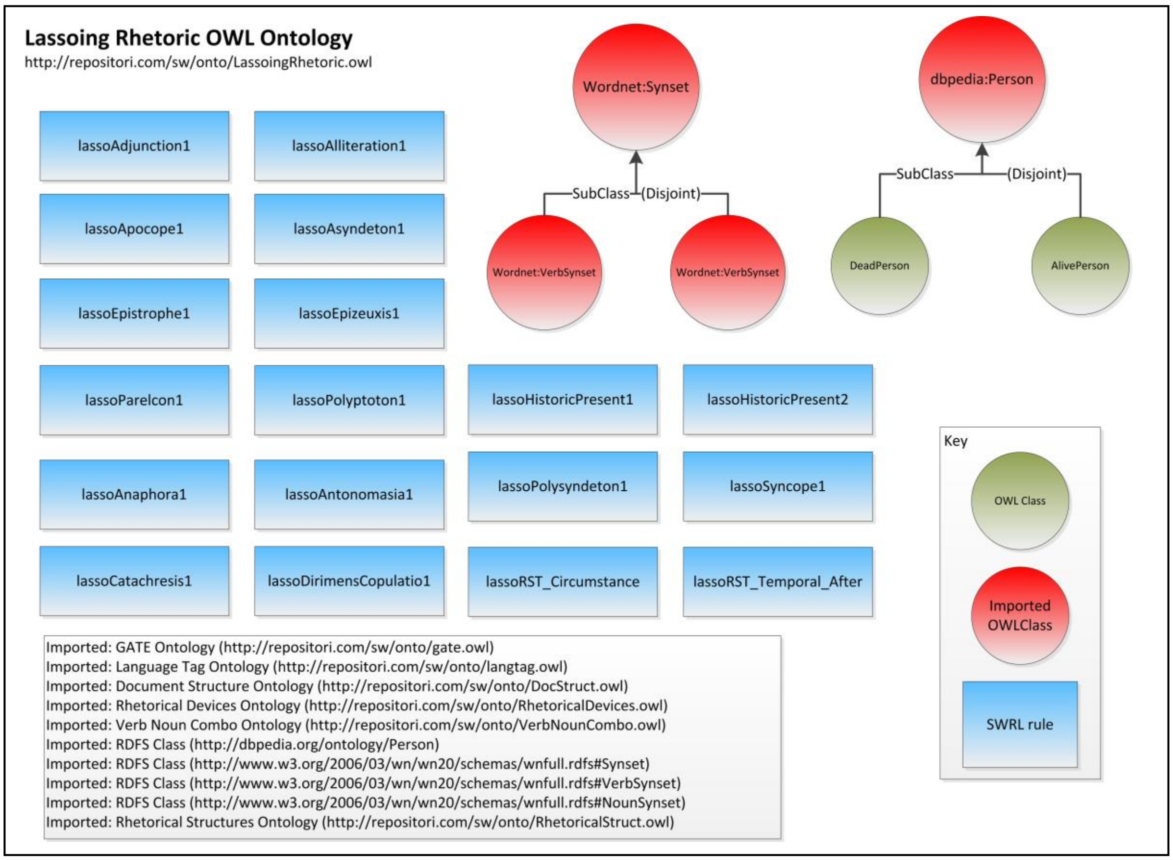
Appendix 5


Rhetorical Devices OWL Ontology
http://repositori.com/sw/onto/RhetoricalDevices.owl
DocStruct:Doc
hasRhetoricalDevice
RhetoricalDevice
gate:hasStartNode
gate:hasEndNode
Integer
Integer
Adjunctio
Alliteration
Anadiplosis
Anaphora
Antonomasia
Apocope
Asyndeton
Catachresis
Chiasmus
Dirimens_Copulatio
Epanalepsis
Epistrophe
Epizeuxis
Historic_Present
Parelcon
Polyptoton
Polysyndeton
Syncope
EpitheticalName
Il_Duce
Iron_Lady
Iron_Duke
MedialCut
dunno
gonna
heartly
ignomy
libray
wanna
FinalCut
attent
morn
oft
MinusComparison
not_only
SuperfluousWord
for_why
PlusComparison
but_also
Imported: OWL Class (http://repositori.com/sw/onto/DocStruct.owl#Doc)
Key
XML Schema Literal
OWL Datatype Property
OWL Object Property
OWL Class
Imported OWLClass
Imported OWL Datatype Property
individuals

Appendix 6


Rhetorical Structures OWL Ontology
http://repositori.com/sw/onto/RhetoricalStruct.owl
DocStruct:Doc
hasRhetoricalStruct
Integer
gate:hasStartNode
Integer
gate:hasEndNode
RhetoricalStruct
TEMPORAL-AFTER CIRCUMSTANCE
Imported: OWL Class (http://repositori.com/sw/onto/DocStruct.owl#Doc)
Key
XML Schema Literal
OWL Datatype Property
OWL Object Property
OWL Class
Imported OWLClass
Imported OWL Datatype Property
individuals

Appendix 7


Lassoing Rhetoric OWL Ontology
http://repositori.com/sw/onto/LassoingRhetoric.owl
Wordnet:Synset
dbpedia:Person
SubClass
(Disjoint)
Wordnet:VerbSynset
Wordnet:VerbSynset
SubClass
(Disjoint)
DeadPerson
AlivePerson
lassoAdjunction1
lassoAlliteration1
lassoApocope1
lassoAsyndeton1
lassoEpistrophe1
lassoEpizeuxis1
lassoParelcon1
lassoPolyptoton1
lassoHistoricPresent1
lassoHistoricPresent2
lassoAnaphora1
lassoAntonomasia1
lassoPolysyndeton1
lassoSyncope1
lassoCatachresis1
lassoDirimensCopulatio1
lassoRST_Circumstance
lassoRST_Temporal_After
Imported: GATE Ontology (http://repositori.com/sw/onto/gate.owl)
Imported: Language Tag Ontology (http://repositori.com/sw/onto/langtag.owl)
Imported: Document Structure Ontology (http://repositori.com/sw/onto/DocStruct.owl)
Imported: Rhetorical Devices Ontology (http://repositori.com/sw/onto/RhetoricalDevices.owl)
Imported: Verb Noun Combo Ontology (http://repositori.com/sw/onto/VerbNounCombo.owl)
Imported: RDFS Class (http://dbpedia.org/ontology/Person)
Imported: RDFS Class (http://www.w3.org/2006/03/wn/wn20/schemas/wnfull.rdfs#Synset)
Imported: RDFS Class (http://www.w3.org/2006/03/wn/wn20/schemas/wnfull.rdfs#VerbSynset)
Imported: RDFS Class (http://www.w3.org/2006/03/wn/wn20/schemas/wnfull.rdfs#NounSynset)
Imported: Rhetorical Structures Ontology (http://repositori.com/sw/onto/RhetoricalStruct.owl)
Key
OWL Class
Imported OWLClass
SWRL rule

Appendix 8

<u>Adjunctio (SWRL rule name: lassoAdjunction1)</u>

DocStruct:Doc(?x) ∧ gate:paragraph(?y) ∧ gate:Sentence(?z) ∧
DocStruct:hasParagraph(?x, ?y) ∧ DocStruct:hasSentence(?y, ?z) ∧ gate:word(?a)
∧ DocStruct:hasFirstWord(?z, ?a) ∧ langtag:Verb(?a) ∧ gate:hasStartNode(?z, ?b)
∧ gate:hasEndNode(?z, ?c) → RhetDev:hasRhetoricalDevice(?x, RhetDev:Adjunctio) ∧
gate:hasStartNode(RhetDev:Adjunctio, ?b) ∧ gate:hasEndNode(RhetDev:Adjunctio, ?c)

<u>Alliteration (SWRL rule name: lassoAlliteration1)</u>

DocStruct:Doc(?x) ∧ gate:paragraph(?y) ∧ DocStruct:hasParagraph(?x, ?y) ∧
gate:Sentence(?z) ∧ DocStruct:hasSentence(?y, ?z) ∧ gate:word(?a) ∧
gate:word(?b) ∧ gate:word(?c) ∧ DocStruct:hasWord(?z, ?a) ∧
DocStruct:hasWord(?z, ?b) ∧ DocStruct:hasWord(?z, ?c) ∧
DocStruct:hasNextWord(?a, ?b) ∧ DocStruct:hasNextWord(?b, ?c) ∧
DocStruct:hasFirstCharacter(?a, ?d) ∧ DocStruct:hasFirstCharacter(?b, ?e) ∧
DocStruct:hasFirstCharacter(?c, ?f) ∧ swrlb:equal(?d, ?e) ∧ swrlb:equal(?e, ?f)
∧ gate:hasStartNode(?a, ?g) ∧ gate:hasEndNode(?c, ?h) →
RhetDev:hasRhetoricalDevice(?x, RhetDev:Alliteration) ∧
gate:hasStartNode(RhetDev:Alliteration, ?g) ∧
gate:hasEndNode(RhetDev:Alliteration, ?h)

<u>Anaphora (SWRL rule name: lassoAnaphora1)</u>

DocStruct:Doc(?h) ∧ DocStruct:hasParagraph(?h, ?i) ∧
DocStruct:hasSentence(?i, ?z) ∧ gate:word(?x) ∧ DocStruct:hasNextWord(?x, ?y) ∧
gate:Sentence(?z) ∧ DocStruct:hasFirstWord(?z, ?x) ∧ gate:word(?a) ∧
DocStruct:hasNextWord(?a, ?b) ∧ gate:Sentence(?c) ∧
DocStruct:hasFirstWord(?c, ?a) ∧ DocStruct:hasNextSentence(?z, ?c) ∧
gate:hasString(?x, ?d) ∧ gate:hasString(?y, ?e) ∧ gate:hasString(?a, ?f) ∧
gate:hasString(?b, ?g) ∧ swrlb:equal(?d, ?f) ∧ swrlb:equal(?e, ?g) ∧
gate:hasStartNode(?x, ?j) ∧ gate:hasEndNode(?b, ?k) →
RhetDev:hasRhetoricalDevice(?h, RhetDev:Anaphora) ∧
gate:hasStartNode(RhetDev:Anaphora, ?j) ∧ gate:hasEndNode(RhetDev:Anaphora, ?k)

<u>Antonomasia (SWRL rule name: lassoAntonomasia1)</u>

DocStruct:Doc(?x) ∧ gate:word(?y) ∧ DocStruct:hasNextWord(?y, ?z) ∧
RhetDev:EpitheticalName(?y) ∧ RhetDev:EpitheticalName(?z) ∧
gate:hasStartNode(?y, ?a) ∧ gate:hasEndNode(?z, ?b) →
RhetDev:hasRhetoricalDevice(?x, RhetDev:Antonomasia) ∧
gate:hasStartNode(RhetDev:Antonomasia, ?a) ∧
gate:hasEndNode(RhetDev:Antonomasia, ?b)

Apocope (SWRL rule name: lassoApocope1)

DocStruct:Doc(?b) ∧ gate:word(?x) ∧ RhetDev:FinalCut(?x) ∧
gate:hasStartNode(?x, ?y) ∧ gate:hasEndNode(?x, ?z) →
RhetDev:hasRhetoricalDevice(?b, RhetDev:Apocope) ∧
gate:hasStartNode(RhetDev:Apocope, ?y) ∧ gate:hasEndNode(RhetDev:Apocope, ?z)

Asyndeton (SWRL rule name: lassoAsyndeton1)

DocStruct:Doc(?x) ∧ gate:word(?y) ∧ DocStruct:hasNextWord(?y, ?z) ∧
DocStruct:hasNextWord(?z, ?a) ∧ langtag:HyphenatedAdjective(?y) ∧
langtag:HyphenatedAdjective(?z) ∧ langtag:HyphenatedAdjective(?a) ∧
gate:hasStartNode(?y, ?b) ∧ gate:hasEndNode(?a, ?c) →
RhetDev:hasRhetoricalDevice(?x, RhetDev:Asyndeton) ∧
gate:hasStartNode(RhetDev:Asyndeton, ?b) ∧
gate:hasEndNode(RhetDev:Asyndeton, ?c)

Catachresis (SWRL rule name: lassoCatachresis1)

DocStruct:Doc(?x) ∧ gate:paragraph(?y) ∧ DocStruct:hasParagraph(?x, ?y) ∧
gate:Sentence(?z) ∧ DocStruct:hasSentence(?y, ?z) ∧ gate:word(?a) ∧
gate:word(?b) ∧ DocStruct:hasWord(?z, ?a) ∧ DocStruct:hasWord(?z, ?b) ∧
VNC:UnusualVerbInCombo(?a) ∧ VNC:UnusualNounInCombo(?b) ∧
DocStruct:hasNextWord(?a, ?b) ∧ gate:hasStartNode(?a, ?c) ∧
gate:hasEndNode(?b, ?d) →
RhetDev:hasRhetoricalDevice(?x, RhetDev:Catachresis) ∧
gate:hasStartNode(RhetDev:Catachresis, ?c) ∧
gate:hasEndNode(RhetDev:Catachresis, ?d)

Dirimens Copulatio (SWRL rule name: lassoDirimensCopulatio1)

DocStruct:Doc(?x) ∧ gate:Sentence(?y) ∧ DocStruct:hasWord(?y, ?z) ∧
DocStruct:hasWord(?y, ?a) ∧ RhetDev:MinusComparison(?z) ∧
RhetDev:PlusComparison(?a) ∧ gate:hasStartNode(?y, ?b) ∧
gate:hasEndNode(?y, ?c) →
RhetDev:hasRhetoricalDevice(?x, RhetDev:Dirimens_Copulatio) ∧
gate:hasStartNode(RhetDev:Dirimens_Copulatio, ?b) ∧
gate:hasEndNode(RhetDev:Dirimens_Copulatio, ?c)

Epistrophe (SWRL rule name: lassoEpistrophe1)

DocStruct:Doc(?h) ∧ DocStruct:hasParagraph(?h, ?i) ∧
DocStruct:hasSentence(?i, ?z) ∧ gate:word(?x) ∧
DocStruct:hasPreviousWord(?x, ?y) ∧ gate:Sentence(?z) ∧
DocStruct:hasLastWord(?z, ?x) ∧ gate:word(?a) ∧
DocStruct:hasPreviousWord(?a, ?b) ∧ gate:Sentence(?c) ∧
DocStruct:hasLastWord(?c, ?a) ∧ DocStruct:hasNextSentence(?z, ?c) ∧
gate:hasString(?x, ?d) ∧ gate:hasString(?y, ?e) ∧ gate:hasString(?a, ?f) ∧
gate:hasString(?b, ?g) ∧ swrlb:equal(?d, ?f) ∧ swrlb:equal(?e, ?g) ∧
gate:hasStartNode(?y, ?j) ∧ gate:hasEndNode(?c, ?k) →
RhetDev:hasRhetoricalDevice(?h, RhetDev:Epistrophe) ∧
gate:hasStartNode(RhetDev:Epistrophe, ?j) ∧
gate:hasEndNode(RhetDev:Epistrophe, ?k)

Epizeuxis (SWRL rule name: lassoEpizeuxis)

DocStruct:Doc(?d) ∧ DocStruct:hasNextWord(?x, ?y) ∧
DocStruct:hasNextWord(?y, ?z) ∧ gate:hasString(?x, ?a) ∧ gate:hasString(?y, ?b)
∧ gate:hasString(?z, ?c) ∧ swrlb:equal(?a, ?b) ∧ swrlb:equal(?a, ?c) ∧
gate:hasStartNode(?x, ?e) ∧ gate:hasEndNode(?z, ?f) →
RhetDev:hasRhetoricalDevice(?d, RhetDev:Epizeuxis) ∧
gate:hasStartNode(RhetDev:Epizeuxis, ?e) ∧
gate:hasEndNode(RhetDev:Epizeuxis, ?f)

Historic Present (SWRL rule name: lassoHistoricPresent1)

DocStruct:Doc(?x) ∧ gate:paragraph(?y) ∧ DocStruct:hasParagraph(?x, ?y) ∧
gate:Sentence(?z) ∧ DocStruct:hasSentence(?y, ?z) ∧ gate:word(?a) ∧
DocStruct:hasWord(?z, ?a) ∧ gate:word(?b) ∧ DocStruct:hasWord(?z, ?b) ∧
DeadPerson(?a) ∧ langtag:ThirdPersonSingularPresentVerb(?b) ∧
gate:hasDependency(?b, ?a) ∧ gate:hasStartNode(?a, ?c) ∧
gate:hasEndNode(?a, ?d) →
RhetDev:hasRhetoricalDevice(?x, RhetDev:Historic_Present) ∧
gate:hasStartNode(RhetDev:Historic_Present, ?c) ∧
gate:hasEndNode(RhetDev:Historic_Present, ?d)

Historic Present (SWRL rule name: lassoHistoricPresent2)

DocStruct:Doc(?x) ∧ gate:paragraph(?y) ∧ DocStruct:hasParagraph(?x, ?y) ∧
gate:Sentence(?z) ∧ DocStruct:hasSentence(?y, ?z) ∧ gate:word(?a) ∧
DocStruct:hasWord(?z, ?a) ∧ gate:word(?b) ∧ DocStruct:hasWord(?z, ?b) ∧
DeadPerson(?a) ∧ langtag:NonThirdPersonSingularPresentVerb(?b) ∧
gate:hasDependency(?b, ?a) ∧ gate:hasStartNode(?a, ?c) ∧
gate:hasEndNode(?a, ?d) →
RhetDev:hasRhetoricalDevice(?x, RhetDev:Historic_Present) ∧
gate:hasStartNode(RhetDev:Historic_Present, ?c) ∧
gate:hasEndNode(RhetDev:Historic_Present, ?d)

<u>Parelcon (SWRL rule name: lassoParelcon1)</u>

DocStruct:Doc(?x) ∧ gate:word(?y) ∧ DocStruct:hasNextWord(?y, ?z) ∧
RhetDev:SuperfluousWord(?y) ∧ RhetDev:SuperfluousWord(?z) ∧
gate:hasStartNode(?y, ?a) ∧ gate:hasEndNode(?z, ?b) →
RhetDev:hasRhetoricalDevice(?x, RhetDev:Parelcon) ∧
gate:hasStartNode(RhetDev:Parelcon, ?a) ∧ gate:hasEndNode(RhetDev:Parelcon, ?b)

<u>Syncope (SWRL rule name: lassoSyncope1)</u>

DocStruct:Doc(?b) ∧ gate:word(?x) ∧ RhetDev:MedialCut(?x) ∧
gate:hasStartNode(?x, ?y) ∧ gate:hasEndNode(?x, ?z) →
RhetDev:hasRhetoricalDevice(?b, RhetDev:Syncope) ∧
gate:hasStartNode(RhetDev:Syncope, ?y) ∧ gate:hasEndNode(RhetDev:Syncope, ?z)

<u>RST Circumstance (SWRL rule name: lassoRST_Circumstance)</u>

DocStruct:Doc(?x) ∧ gate:paragraph(?y) ∧ DocStruct:hasParagraph(?x, ?y) ∧
gate:Sentence(?z) ∧ DocStruct:hasSentence(?y, ?z) ∧ gate:word(?a) ∧
DocStruct:hasWord(?z, ?a) ∧ gate:word(?b) ∧ DocStruct:hasWord(?z, ?b) ∧
gate:hasDependency(?a, ?b) ∧ langtag:Noun(?b) ∧ gate:hasStartNode(?a, ?c) ∧
gate:hasEndNode(?a, ?d) →
RhetStruct:hasRhetoricalStruct(?x, RhetStruct:Circumstance) ∧
gate:hasStartNode(RhetStruct:Circumstance, ?c) ∧
gate:hasEndNode(RhetStruct:Circumstance, ?d)

RST Temporal-After (SWRL rule name: lassoRST_TemporalAfter)

DocStruct:Doc(?x) ∧ gate:paragraph(?y) ∧ DocStruct:hasParagraph(?x, ?y) ∧
gate:Sentence(?z) ∧ DocStruct:hasSentence(?y, ?z) ∧ gate:word(?a) ∧
DocStruct:hasWord(?z, ?a) ∧ gate:word(?b) ∧ DocStruct:hasWord(?z, ?b) ∧
gate:hasDependency(?a, ?b) ∧ langtag:Verb(?b) ∧ gate:hasStartNode(?a, ?c) ∧
gate:hasEndNode(?a, ?d) →
RhetStruct:hasRhetoricalStruct(?x, RhetStruct:TemporalAfter) ∧
gate:hasStartNode(RhetStruct:TemporalAfter, ?c) ∧
gate:hasEndNode(RhetStruct:TemporalAfter, ?d)

Appendix 9

Rhetorical form	**Test text string**
Adjunctio	Lassoing rhetoric with OWL and SWRL.
Alliteration	Perhaps Penelope picked plums today?
Anaphora	We shall fight in France, we shall fight on the seas and oceans, we shall fight with growing confidence and growing strength in the air, we shall defend our island, whatever the cost may be. We shall fight on the beaches, we shall fight on the landing grounds, we shall fight in the fields and in the streets, we shall fight in the hills; we shall never surrender.
Antonomasia	Will the Iron Lady meet the Iron Duke for tea with Il Duce?
Apocope	Will it occur oft or not at all?
Asyndeton	The result was an unhappy, miserable, unfair, unfixed, great mix
Catachresis	Lassoing rhetoric with OWL and SWRL.
Dirimens Copulatio	Not only was this project undertaken in the UK, but also during the summer of 2010.
Epistrophe	When I was a child, I spake as a child. I understood as child; I thought as a child.
Epizeuxis	It's all football, football, football with you!
Historic Present	When Aristotle says he thinks that correlatives are commonly held to come into existence together he means for the most part.
Parelcon	For why should he not think that?
Syncope	Dunno we think.
RST Circumstance	After its previous mayor committed suicide last year, an investigation disclosed that town officials regularly voted on their own projects.
RST Temporal-After	The dollar finished lower yesterday after tracking another rollercoaster session on Wall Street.