Extracting Generic Statements for the Semantic Web

Sangweon Suh

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2006

Abstract

Much of the natural language text found on the web contains various kinds of common sense knowledge and such information is potentially an important supplement to more formal approaches to building knowledge bases for Semantic Web applications. The common sense knowledge is often expressed in the form of generic statements such as "*Elephants are mammals.*" The generic statement refers to a sentence that talks about *kinds* rather than individuals.

In this thesis, we develop methods of automatically extracting generic statements from unrestricted natural language text and mapping them into an appropriate logical formalism for Semantic Web. The extraction process utilises cascaded transduction rules for identifying generic sentences and extracting relations from them. NLP pipeline and a suite of XML tools designed for generic manipulation of XML were used to carry out the series of tasks. The Wikipedia XML corpus was adopted for development, as a rich source of generic statements, and we used existing annotations of the ACE 2005 corpus for testing the identification of generic terms.

For identifying generic terms, we use a set of morpho-syntactic features coded into definite transduction rules and apply the rules to the noun groups resulting from chunking. Next, relations are extracted with those identified terms as arguments. The semantic interpretation for the relation extraction was based on what can be called semantic chunking. Finally, we show how these extracted relations can be converted to RDF(S) statements as a knowledge representation for Semantic Web.

Acknowledgements

I should like to record my gratitude to my project supervisor, Prof. Ewan Klein who guided this work from the early stage and gave me valuable comments and advice.

I'm indeted to Mr. Harry Halpin in Language Technology Group whose initial work on this topic was inspiring and kindled my interest (and thanks for acting as judge for my RDF data evaluation).

Also, my thanks go to Mr. Richard Tobin and Dr. Maria Milosavljevic who helped me with solving technical problems I've had during the course of this work.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Sangweon Suh*)

Table of Contents

Chapter 1

Introduction

The Semantic Web's overarching goal, as stated by Tim Berners-Lee, is to enable large-scale creation of machine-readable representations of meaning that make the Web more intelligent and responsive(Tim Berners-Lee and Lassila, 2001). Ultimately, the Semantic Web aims at allowing machines sharing and exploiting the distributed knowledge in a scalable and adaptable manner but in a decentralised way like Web, that is, without central authority.

Availability of ontological knowledge is a crucial foundation for the Semantic Web, which provides common semantics for machine interpretation of resources. To date, efforts have been largely directed to formalizing domain specific ontologies. For example, the Gene Ontology (GO) project has been developed in an effort to address the need for consistent descriptions of gene products in different databases[1].

The use of URIs as identifiers plays a crucial role in making connections between these diverse ontologies. But in practice it is often difficult to merge ontologies in disparate domains. There are several reasons for this, but one of the most obvious is that most ontologies rely upon a large amount of *common sense* background knowledge that are not formalized and explicitly given in the ontology.

Lack of the explicit common-sense knowledge seriously hampers the merging or interrelation of disparate ontologies. For example, an ontology of food should mention wine, but in a restaurant ontology, it may not be stated that one drinks wine in a restaurant, or even that one can only drink things that are liquid and wine is a liquid.

On the other hand, there is also a serious bottleneck problem with manually devel-

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

oping conceptual information to build ontologies. Therefore, automatically building common sense knowledge bases in Web accessible format is thought to be an important issue in ontology development.

In this project, we address the question of whether it would be possible to bootstrap a repository of common-sense knowledge based on natural language resources on the Web such as Wikipedia. We develop methods of extracting a large class of common-sense generic statements from unrestricted text and to transform them into a suitable knowledge representation.

The generic statement refers to a sentence that talks about *kinds* or *classes* of individuals instead of individuals themselves. It is in fact a primary manner of expressing common-sense knowledge in a natural language sentence. For example, sentences like ”Elephants are mammals.” or ”Elephants are grey.” are, without a preceding article for the subject noun, recognized as generic statements about elephants in general.

1.1 Background

About Generic Statements

The generic statements or generics have long been recognized as one of the difficult problems to deal with in formal linguistics and are a subject of research in the linguistic philosophy. Especially, they should be distinguished from the universally quantified statements over sets of individuals; ”Dogs have four legs.” is generally true of dogs in normal condition but there are individual dogs that, for example, lost their legs and don’t have four legs. And yet, such dogs will be still regarded as dog. To put otherwise, we normally want a generic statement to remain true even if some exceptional cases are encountered.

There has been debate on the correct interpretation of generic sentences within computational linguistics and philosophy of language. For example, Cohen (2002) proposes that generics should be interpreted as probabilities while Carlson (1982) classifies them as more related to habitual actions or statements of taxonomic knowledge. The correct interpretation of generics can influence the choice of knowledge representation.

Common Sense Research

Much research has been carried out into commonsense knowledge representation given its importance for Semantic Web. Amongst important projects developed, the largest one is the Cyc project with its commonsense assertions collected over two decades (Lenat, 1990). In Cyc, the commonsense facts was stored in a custom knowledge representation language called CycL by knowledge engineers. A small part of Cyc known as OpenCyc has been made publicly available that contains 47,000 concepts from Cyc ontology.

The second largest is the *OpenMind Commonsense* project (Singh, 2002). While Cyc relied on trained knowledge engineers, OpenMind collected data via untrained general Web users in the form of natural language statements. The project gathered 1.6 million statements of which a subset of 700,000 facts can be accessed through the *ConceptNet* database. ConceptNet models common-sense via the spreading activation of a semantic network, but it does not support logical deduction, only allowing similarity judgements to be made.

RDF/RDFS

RDF is a data model for semantically describing resources. The RDF statements consist of (object, attribute, value) triples. For instance the following triple describes a web resource giving information about a movie:

```
( http://www.imdb.com/title/tt0120737,
  http://purl.org/dc/elements/1.1/title,
  "Lord of the Rings"                  )
```

We can think of this triple as a logical formula $P(x,y)$ where the binary predicate P(the second element of the triple) relates the object x to the object y. Here, the meaning of P is pre-defined and should be respected by all RDF processing software. The RDF document, machine-processable representation of these statements, follows XML syntax. RDF is domain independent without assumptions about a particular domain of use. Through RDF Schema(RDFS) user can define their own RDF vocabulary specifying which properties apply to which kinds of objects and what values they can take, and also relationships between objects.

OWL and Description Logic

Description Logic(DL) is regarded as a cornerstone of Semantic Web for its use in ontology design. Description Logics are a family of knowledge representation languages and form a decidable subset of First Order Logic, with more expressive power. In particular, DL allows defining classes explicitly using relations between classes. This is in contrast to RDFS where only superclass/subclass relationship can be defined. The following are examples of such definition of classes:

Class	Set of instances
$Elephant$	$\{x \mid Elephant(x)\}$
$Elephant \sqsubseteq Mammal$	$\forall x(Elephant(x) \Rightarrow Mammal(x))$
$Herbivore \doteq \forall eat.Plant$	$\{x \mid \forall y(eat(x,y) \Rightarrow Plant(y))\}$
$Elephant \sqsubseteq Herbivore$	$\forall x(Elephant(x) \Rightarrow Herbivore(x))$
$Elephant \sqcap Carnivore \sqsubseteq \bot$	$\emptyset$

Compared to RDF(S), DL can express disjointness and restriction of predicates as given in the table. DL sentences can be categorized into two distinct parts: TBox and ABox. The TBox consists of terminological statements such as those in the above example, whereas ABox consists of individual assertions(e.g. Jerome is an Elephant).

OWL DL is a sublanguage of OWL, an ontology language which has been developed in extension of RDF(S) [2]. But the extended requirements for an improved ontology language were not satisfiable all at once, and 3 species of OWL were defined by W3C, namely OWL LITE, OWL DL and OWL FULL. OWL DL was designed to meet the efficient reasoning support requirement and its language corresponds to DL.

1.2 Tasks

The project involves two different parts. For the first, we are concerned with identifying generic sentences through matching some structured expressions that indicate the presence of generic terms and sentences. The matching is applied to the parts of each sentence and matched sentences are sent through an NLP pipeline so as to render them suitable for subsequent knowledge representation. We'll use shallow parsing, also

[2] http://www.w3.org/TR/owl-ref/

known as *chunking* for the extraction of generic noun groups and predicate-relation structures contained in the sentence.

The second part concerns automatic mapping of the identified generic statements into a logical representation. We will use a generic XML processing method employing the XSLT stylesheet to implement this conversion. After the logical representation is done, evaluation by human judges will be performed to assess the effectiveness of the conversion and quality of the resulting RDF statements as representation for generic statements.

Chapter 2

A Review on the Generics and Generic Terms in English

2.1 Formal Semantics of Generics

The semantic interpretation of generics has received copious discussion in the linguistics and philosophy literature (Carlson and Pelletier, 1995; Cohen, 2002). Although relevant discussions tend to be abstract by nature, several salient points are reviewed here as they provide insight into the subject and can indeed have significant bearing on the practice of generics processing.

The primary challenge around generics is that they have particular characteristics of default statements, in the sense that generics admit exceptions without changing their truth condition. For example, the generic statement *Dogs bark.* is not rendered false by the fact that some dogs fail to bark. Conversely, the fact that a certain number of exemplars share a common property P does not warrant a generic statement that the kind as a whole has the property P; this is natural especially when exemplars of the kind are scarce.

Suppose we assume that a generic statement is equivalent to the corresponding universally quantified statement. We can formally express it as in the following logical formula(Carlson, 1982). That is, the proposition that the kind F denoted by a generic term has property G is equivalent to the proposition that all individuals of that kind F have property G. (Γ is a generic operator.)

$$G\Gamma xF(x) \leftrightarrow \forall x(F(x) \rightarrow G(x)) \tag{2.1}$$

Now, if we accept as premises the said characteristics of generics, then it amounts to saying that generics are *not* universally quantified statements, both in necessary and sufficient conditions. Also, a related issue is that in natural language there are predicates which are only meaningfully applied to kinds but not to individuals. These include *extinct*, *common*, *widespread*, and so on. However, the above formula 2.1 does not allow for this, hence another argument for its inappropriateness.

In fact, there are two varieties of genericity involved in generics. The first is reference to kind: that is, we have to consider whether the generic terms such as *dogs* can be taken as denoting something in the model theoretic view, or they should be treated as non-denoting terms such as *nobody*. Traditionally, philosophers have argued that generic terms cannot denote something because allowing them to do so can result in counterintuitive conclusions(Russell, 1905; Bacon, 1973).

One common way to get round the denoting problem was to attribute some implicit quantification to the subject generic term:

(All) dogs are mammals.

(Most) dogs eat meat.

(Some) dogs like chasing rabbits.

However, the difficulty is that it is often not clear which quantifier should be applied to get the appropriate truth condition of the statement. In fact, generic terms have characteristics of denoting terms in several ways. In particular, they can be antecedent of pronouns as in:

Dogs are highly social animals. *They* are also man's best friend.

Also, generic terms fail to interact with modal operators, quantifiers and negatives to produce scope ambiguity, a well-known problem in computational semantics(Blackburn and Bos, 2005), and they function similarly to proper names in this regard.

Every woman likes a few dogs.
Every woman likes several dogs.

Every woman likes *dogs*.
Every woman likes Peter.

So, in model theoretic framework, generic terms are taken as denoting kinds of things and the kinds are represented as basic entities of the model, along with other more normal entities. (But there are of course alternative suggestions.)

The second variety of genericity is concerned with predicates of sentence rather than noun phrases, when they report a kind of general property instead of presenting specific episodes or isolated facts.

Dogs bark.

As in the analysis of generic terms, a popular means of analysing sentences like this is to posit an implicit quantification over times: such as "*Dogs often bark.*" But this also has problems in specifying the exact nature of the implicit quantifiers, as they vary sensitively depending on the predicates of sentence. There are a bunch of other problems as well and this quantification approach to the truth conditions of such sentences is complicated.

Carlson (1982) proposed the *habitual* analysis of generics, which removes the idea of any quantification over times. It rather addresses the dispositional nature of the predicates in question. Without consideration of quantifiers, one common way to deal with habitual sentences(or habituals) is to make them tenseless or timeless, but they should be regarded as tensed. Although habituals or generics[1] are tensed, they have different truth conditional properties with respect to time and space, compared to episodics. For one, habituals/generics are not spatially disjoint.

Proposals for representing habituals/generics include using quantifier-like element like *generally*, representation by a modal logic, and employing the concept of *individuals*. Delving a bit more into the philosophical research, the concept of individuals here is rather different from that discussed in Semantic Web ontology, i.e., individuals of a class. But they are discussed as a central concept that sheds light on the analysis of genericity. Individuals are regarded as things that transcend time and space and they are abstracted from a mass of associated facts.

Dogs barked.
Dogs bark.
Dogs will bark.

Each of the three sentences above can have two readings: episodic reading and generic(or non-episodic) reading, and the distinction between them is explained by way of the individual: here it is the kind 'dogs'. The generic reading is obtained by attributing the predicate 'bark' directly to the individual, i.e., the kind 'dogs', whereas

[1] when the arguments of the habitual predicate are generic terms

the episodic reading arises when the predicate is attributed to a time-space instance of the kind.

Moreover, such explanation can deal with more general class of sentences where properties and nominalised sentences may appear as subject argument; the set of generics become a subclass of it. Individuals include kinds, properties and nominalised phrases, and sentences can have two readings depending on whether the predicate is attributed to the individual or to the instances of it.

Redness appears only in visible objects. (non-episodic)
Redness suddenly appeared before John. (episodic)

To eat moderately is wise. (non-episodic)
To eat moderately was wise on that occasion. (episodic)

Terrorism is menacing to the public. (non-episodic)
Terrorism was menacing to the public at the time. (episodic)

In summary, we can say that generics are facts about kinds as individuals in the above sense, whereas the episodics are about time-space instances of such individuals.

In the section 5.3, there is some additional discussion related to the semantics of generics but in the following, we will only focus on the issue of modelling kinds as RDFS classes and individuals mean the RDF individuals. Chapter 3 deals with the first variety of genericity explained above, namely reference to kind and chapter 5 is relevant to the second variety of genericity, that is, genericity around the predicates of sentence.

2.2 Kinds and Countability in English Nouns

As we are working with word forms of head nouns, especially bare plural constructions, it would be informative for our purpose to consider the studies on countability in English nouns(Carlson, 1977; Croft, 2000). According to the viewpoint of construction grammar, three main English countability constructions are as follows. (NUM means a numeral construction.)

1. Counting construction: [a _], [NUM _(-s)]
2. Bare Plural construction: [_-s]
3. Bare Singular construction: [_]

Counting	**Bare Plural**	**Bare Singular**
individual	group	substance
(inflected/uninflected) variety	variety	group
(inflected/uninflected) kind	kind	kind
	dual object	gradient/scalar property

Table 2.1: *Senses construed from countability constructions in English nouns*

In the above, the counting construction construes the entity in question as discrete and internally heterogeneous, and the words occurring in the counting construction take singular or plural form. The bare plural construction construes the entity in question as a group or collection, that is, non-discrete but internally heterogeneous. Finally, bare singular construction construes the entity in question as non-discrete and internally homogeneous. Table 2.1 summarises the senses so construed of the three constructions respectively.

In that regard, the definite construction with definite article 'the' or possessive adjectives stands independently: all the three constructions above can alternate with the definite constructions, as shown in some of the examples below.

2.2.1 Counting construction

The Counting construction can be construed in four different ways. The first sense is a specific discrete individual or set of individuals.

- I bought *a best-seller* yesterday.
- There is *a cat* in the backyard.
- Would you like *an apple*?
- *Two flies* got into our kitchen.
- *A large mountain* was visible on the horizon.
- We ordered *three beers*.

Some nouns are not subject to inflection for number agreement but have the same individual sense:

- I caught *six salmon* in the river.
- I saw *many fish* in the pool.
- She picked *several clover*.

The second sense of the counting construction is the *variety* interpretation. In this interpretation, the individual unit is a variety of the kind labeled by the noun. The variety interpretation is most easily obtained with nouns denoting biological kinds, which tend to occur in varieties.

- There are over *a hundred butterflies* found in this country.
- *Many apples* are cultivated in Britain.
- *the fishes* of the North Atlantic
- *the soils* of Great Britain
- There are *three beans* in this salad.

The variety interpretation can also be found with nouns denoting foodstuffs and artifacts.

- Are you familiar with *the wines* produced in Spain?
- We manufacture *six cars* and *three light trucks*.
- We carry *three sheets*: single, queen size, and king size.
- This cabinet has *five different woods* in it.

Some of the same nouns above, without number inflection, have the variety interpretation likewise.

- *Six salmon* in the Pacific are already extinct.
- *A few shellfish* are almost extinct in the globe.

Also, in the counting construction, there are found certain nouns for which variety interpretation is the only interpretation available.

- *A dozen vegetables* are grown in my greenhouse in Highland.

Finally, the third sense of the counting construction is the *kind* interpretation, referring to the kind as a whole, not just an individual variety of the kind. The kind interpretation is possible with either a or the. Like the variety interpretation, the kind interpretation is easily available for nouns denoting biological kinds, but the interpretation is also available for other nouns denoting such entities as artifacts, geographical features and body parts.

- *A mouse* is a small furry mammal.
- *The penguin* can be found only in the Antarctica.
- *The car* is responsible for thousands of deaths every year.
- *A sheet* is normally made of cotton.
- *The human brain* has increased in size over the evolution of the species.
- *A mountain* can be the result of volcanic activity.

2.2.2 Bare Plural construction

Four senses can be obtained with the bare plural construction. The first one is reference to kinds, also known as the *generic bare plural*(Carlson, 1977). This is known to be the most common form of reference to kinds in English.

- *Dogs* are highly social animals like *humans*.
- *Elephants* are the largest land animals alive today.
- *Cheap earrings* can damage *your earlobes*.
- *Beetles* are one of the most diverse groups of *insects*.
- Do they make *good scissors* in Britain?
- *Women* are allowed to wear *trousers* to work here.

- *Computers* are extremely versatile.
- *Legs* tend to be longer than *arms*.
- *Birds* fly.
- *Dogs* bark.

But bare plurals also can lead to slightly different meaning of *group*, referring to a group of individuals of indeterminate cardinality. This sense is called the existential bare plural (Carlson, 1977). Examples pertaining to this sense are like the below.

- *Chairs* were scattered around the room.
- *Socks* were all over the floor.
- I threw out *the coffee dregs*.
- *Straws* stuck out of her hair.
- He wouldnt eat *his vegetables*, and we threw them away.
- *Women* are allowed to wear *trousers* to work here.
- *Computers* are extremely versatile.
- *Legs* tend to be longer than *arms*.
- *Birds* fly.
- *Dogs* bark.

As a special case of this sense of bare plurals, there is the dual object sense. The ordinary interpretation of the following examples is that they refer to a pair of the relevant objects, either attached or unattached but belonging together as a pair.

- Please pass me *the scissors*.
- She put on *her trousers*.
- He blinked *his eyes*.
- I had *my glasses* broken.

And with some nouns, bare plurals can refer to heterogeneous objects and thus varieties.

- *Tonight's leftovers* consist of chicken, string beans and wild rice.

2.2.3 Bare Singular construction

The bare singular construction has four senses. The first sense is substance. This is the common interpretation for a substantial range of nouns.

- There was *mud* on the carpet.
- I put *butter* on the bread.
- There is more plaster than *brick* in these walls.
- He left *some chicken* on the plate.
- This marzipan contains *pistachio* as well as *hazelnut*.

The second sense for the bare singular is group interpretation. Examples are like the following.

- *Spawning salmon* filled the stream.
- *Fish* circled around the sunken ship.
- *Clover* dotted the meadow.
- *Furniture* filled the garage from floor to ceiling.

Therefore, there are group senses in both the bare plural and the bare singular constructions. The grouped entities occurring in the bare singular construction tend to be smaller than the grouped entities occurring in the bare plural construction, although this is relative to experiential domain.

The bare singular also has a kind sense. The kind sense is found for nouns such as the following.

- *Wine* is an alcoholic beverage.
- *Sandy soil* is better for these plants.

- I like *pineapple*.
- I prefer *wood* to *plastic* in furniture.

The nouns which are uninflected in the individual-sense counting construction may be considered 'pseudo-bare singular' in their kind sense. Sometimes these nouns take a plural verb form, suggesting that they belong to an 'uninflected bare plural' kind sense.

- *Clover* is/are found at low elevations.
- *Fish* breathes/breathe through their gills.

The fourth sense of the bare singular is to a gradient or scalar property of the entity in question and it is fairly common in spoken English: in the sentence like *It's interesting because they have more house 'cause they're on the side.*, *house* means the surface area of the house.

Chapter 3

Extraction of Generic Terms

3.1 Generic Noun Group Labeling

3.1.1 Assumptions

In this project, our aim is to develop a practical system that can extract plain common sense statements useful for ontology building, so that we take a pragmatic approach for identifying them in the text. We don't attempt to fully address three countability constructions explained in 2.2 and made a few simplifying assumptions around generic terms and sentences.

For the identification of generic terms, we concentrate on the bare plurals plus singulars with *every* and *any* determiners that is, universally quantified terms. Definite constructions, that is, *the* or possessive adjectives plus a noun, were ruled out because that inclusion accompanies too much of false alarms in generic term labeling, given that our method relies on basic morpho-syntactic rules and does not use any means from discourse semantics such as presupposition resolution.

In a similar vein, counting constructions having an indefinite article and the kind interpretation, such as in "*A mouse is a small furry mammal.*", were not taken into consideration. Constructions with a specific head noun can have statistically significant preferences to a particular interpretation such as individual versus kind, as can be inferred from the discussion in 2.2.1. Thus, a sophisticated system to deal with these constructions may need to be backed by lexical semantics and data intensive methods.

3.1.2 Methodology

We base the generic term extraction on noun groups resulting from chunking, and code the identification rules into deterministic transduction rules used by an XML tool called lxtransduce. Information such as POS tags is made use of appropriately and the grammar of rules tries to match any head noun in bare plural form without an article or in singular form with preceding every/any determiners. The detection of plurals is actually based on the POS tag information(NNS and/or NNPS) so that noun groups with an uninflected head noun such as police, fish, etc. are also detected as bare plurals.

Whenever the match is found, the containing noun group is labeled as generic. The labeling can be done flexibly as allowed by XML. We use two ways as appropriate, either wrapping the noun group in a new element or adding an attribute of genericity to each word belonging to the noun group.

Some exception lexicons were set, as in the following list, to filter out noun groups with certain determiners, adjectives and head nouns. Also, a vocabulary of time, dates and measurement units were taken out of consideration as head noun.

- **articles**: a/an, the, no
- **indefinite determiners**: another, some, each, both, either, neither
- **indexical prenominal modifiers**: such, other, this/these, that/those, ...
- **quantifiers**: many, few, several, multiple, ...
- **numbers**: five, fifth(s), dozen(s), million(s), ...
- **head noun exceptions**: others, ones, one, ...

As a further filtering of candidate noun groups, we use results from Named Entity Recognition(Tjong Kim Sang and De Meulder, 2003), which is performed after the chunking in the processing pipeline (but before genericity labeling). Named entities include location and organisation names which are proper nouns.

So having gone through the above steps of processing we get genericity labeled noun groups. The Figure 3.1 shows a returned sentence XML structure after the labeling is done. We can see the `gen` attribute attached to each word element belonging to the same noun group (as identified by `ng-id` attribute).

```
<s>
  <w ng-id="ng1" p="DT" phr="B-NP">A</w>
  <w ng-id="ng1" l="fox" p="NN" headn="yes" phr="I-NP">fox</w>
  <w vg-id="vg1" l="be" p="VBZ" headv="yes" phr="B-VP">is</w>
  <w ng-id="ng2" p="DT" phr="B-NP">a</w>
  <w ng-id="ng2" l="member" p="NN" headn="yes" phr="I-NP">member</w>
  <w p="IN">of</w>
  <w p="DT">any</w>
  <w p="IN">of</w>
  <w ng-id="ng3" p="CD" phr="B-NP">27</w>
  <w ng-id="ng3" l="species" p="NNS" headn="yes" phr="I-NP">species</w>
  <w p="IN">of</w>
  <w ng-id="ng4" p="JJ" phr="B-NP" gen="probably">small</w>
  <w ng-id="ng4" p="JJ" phr="I-NP" gen="probably">omnivorous</w>
  <w ng-id="ng4" l="canid" p="NNS" headn="yes" phr="I-NP" gen="probably">canids</w>
  <w p=".">.</w>
</s>
```

Figure 3.1: *Genericity labeling of noun groups*

3.1.3 Illustration with Wikipedia corpus

From the Wikipedia XML corpus (Denoyer and Gallinari, 2006), a random sample of 659 articles were selected for experiments and development of rules. Figure 3.2 displays an HTML screenshot of the labeling result for a sample document (ID: 11299.xml, from the *original* section of the corpus). An XSLT stylesheet was applied to produce the HTML for display after the the labeling module returns the XML with genericity annotated. The chunks marked with keys, `ng` and `vg` represent noun groups (not identified as generic) and verb groups, respectively and bare plurals and every/any quantified constructions are identified and marked with the key `ng-gen`. Also, Figure 3.3 shows the bare singulars identified as generic for the same document.

As POS tagging and chunking cannot be perfect and their results are occasionally erroneous for difficult sentence structures etc., apparent labeling errors were encountered in such cases. (In the result for "*These sounds grade into one and other and span five octaves ...*" *grade* is not a bare singular but it's a verb.)

Having processed all the noun groups, clauses with generic terms in subject or object position will be counted as generic statements in our framework, and relations inside them are extracted for logical representation in later processing.

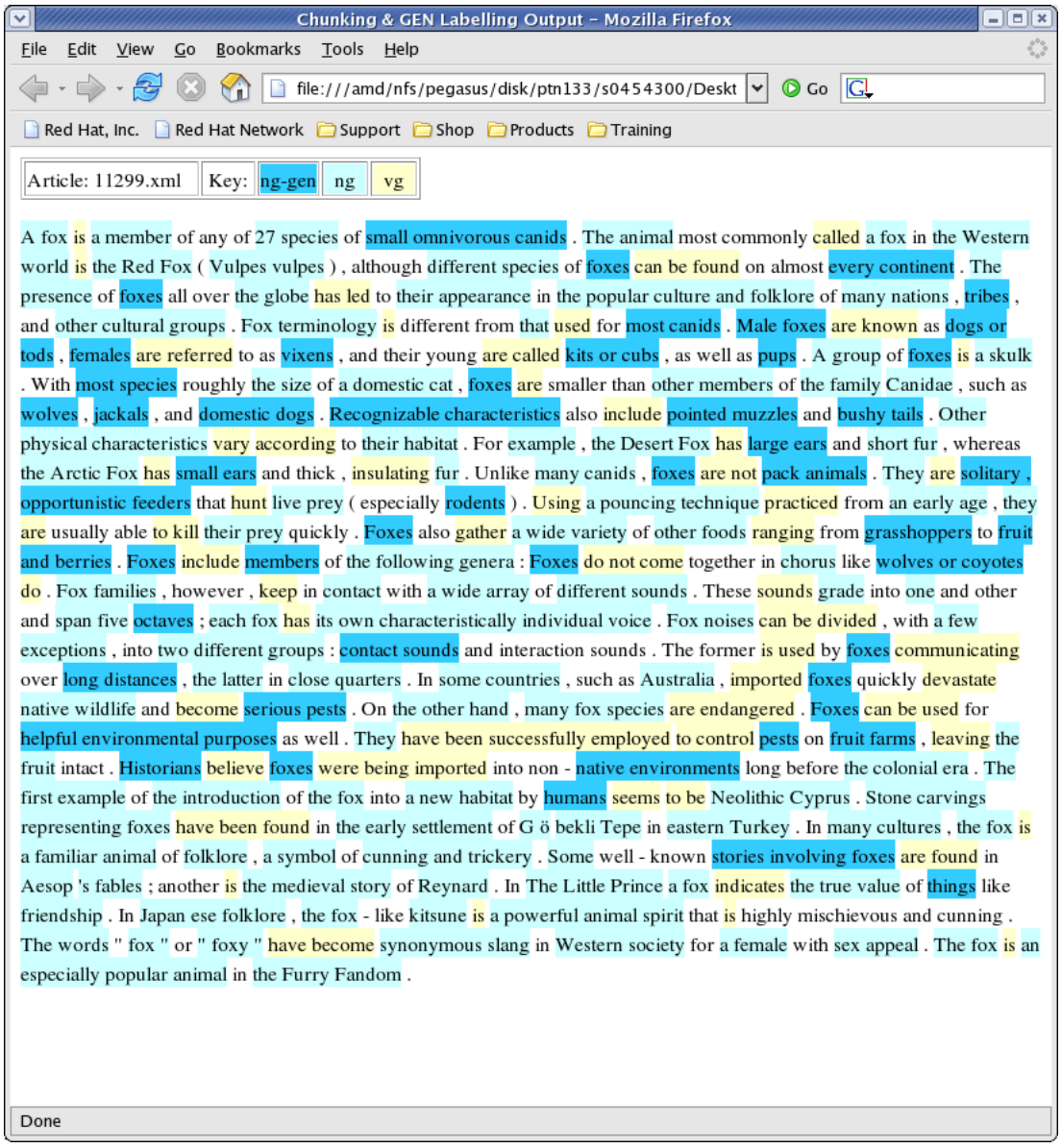

Figure 3.2: *Identification of chunks and generic terms*

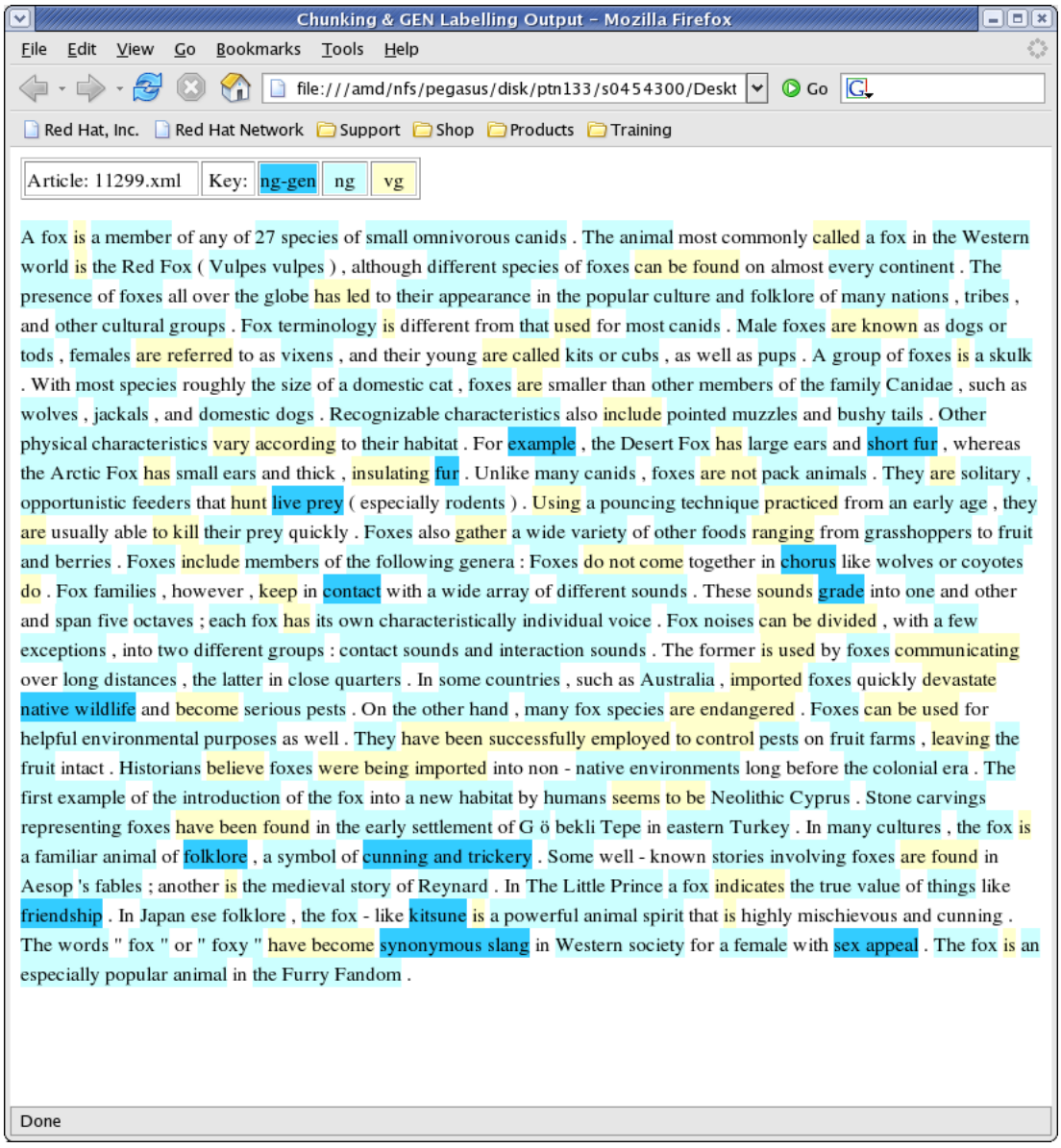

Figure 3.3: *Identification of chunks and generic terms: bare singulars*

3.2 Evaluation with ACE 2005 Corpus

3.2.1 ACE 2005 multilingual training corpus

In order to evaluate our developed method, we used ACE 2005 Multilingual Training Corpus as test data. The corpus includes annotations of generic entities and events. The ACE annotation scheme(ACE05, 2005) defines five classes of entities and one of them was assigned to each entity as XML attribute. They are negatively quantified (NEG), attributive (ATR), specific referential (SPC), generic referential (GEN) and under-specified referential (USP).

An entity is SPC when the entity being referred to is a particular, unique object (or set of objects): (e.g.) *[John's lawyer] won the case.* ATR indicates that the entity is only being used to attribute some property or attribute to some entity, such as in *John is [a lawyer].* Finally, the USP class is reserved for quantified and ambiguous NPs which the annotators could not confidently assign to the other four classes.

An entity in ACE is an object or set of objects in the world, and an entity mention is a reference to an entity, which may expressed in text by a common noun/noun phrase or by a pronoun. The corpus designates 7 types of entities and their subtypes, and only these types of entities are annotated. They are listed in the below. For each entity, the annotation records the type of the entity, subtype, class, and all the textual mentions of that entity.

1. Person(PER): a single individual or a group
2. Organization(ORG): corporations, agencies, and other groups of people defined by an established organizational structure
3. Geo-political Entity(GPE): geographical regions defined by political and/or social groups
4. Location(LOC): geographical areas and landmasses, bodies of water, geological formations, etc.
5. Facility(FAC): buildings and other permanent man-made structures and real estate improvements
6. Vehicle(VEH): a physical device primarily designed to move an object from one location to another

```
<entity ID="AFP_ENG_20030304.0250-E1"
                            TYPE="ORG" SUBTYPE="Medical-Science" CLASS="SPC">
  <entity_mention ID="AFP_ENG_20030304.0250-E1-3"
                                     TYPE="NAM" LDCTYPE="NAM" LDCATR="FALSE">
    <extent>
      <charseq START="493" END="516">The Davao Medical Center</charseq>
    </extent>
    <head>
      <charseq START="497" END="516">Davao Medical Center</charseq>
    </head>
  </entity_mention>
  <entity_mention ID="AFP_ENG_20030304.0250-E1-4"
                                       TYPE="NOM" LDCTYPE="NOM" LDCATR="TRUE">
    <extent>
      <charseq START="519" END="548">a regional government hospital</charseq>
    </extent>
    <head>
      <charseq START="541" END="548">hospital</charseq>
    </head>
  </entity_mention>
  <entity_attributes>
    <name NAME="Davao Medical Center">
      <charseq START="497" END="516">Davao Medical Center</charseq>
    </name>
  </entity_attributes>
</entity>
```

Figure 3.4: *Entity structure in ACE corpus*

7. Weapon(WEA): physical devices primarily used as instruments for physically harming or destroying other entities

One thing to note is that in this scheme the *entity* is an abstract object and different than any word tokens occurring in the text. One or more spans of text or character sequences are annotated that are judged to be expressions of such an entity. The figure 3.4 is an example of the XML entity structure employed in ACE. We can see that the `<entity>` element has two `<entity_mention>` elements as its children, both of which denote the same parent entity. The class of the entity is SPC as indicated by the `CLASS` attribute of the element, and the entity type is ORG.

Table 3.1: *Labeling precision against* ACE *annotations*

entity class	SPC	GEN	USP	NEG	ATR
ng-gen	34.2%	28.9%	36.8%	0.1%	0.0%

(total 1244 ng-gen terms)

3.2.2 Precision of Generic Term Labeling

The evaluation was performed to estimate how well the generic noun-group labels produced match Gold Standard labels. We used 412 files in their broadcast news, news wire and web log sections, which account for 55% of the whole corpus. Table 3.1 gives the results.

Table 3.1 indicates that among the noun groups identified as generic (labeled with ng-gen) 34.2% was annotated as SPC, 28.9% as GEN and so on, according to the gold standard labels of ACE. So, the value of ng-gen & GEN pair is the exact precision result and other entries can be understood as representing confusions of labeling: that is, terms in fact corresponding to non-generic entities(SPC, USP, NEG, ATR) were labeled as generic.

If we only accept exact matches with GEN class, the labeling precision is 28.9%. But since our our operational definition of generics, based on morpho-syntactic features also encompasses many terms that are labeled as USP in ACE, it would be more appropriate to consider matches with GEN and USP merged together.

Merging GEN and USP gives us the higher precision of 65.7%. As another point relevant to merging these two classes, we may consider that the primary objective in recognising generic terms would be to ensure that we can attain reasonable discrimination against specific-referential (SPC) entities.

Among the noun groups extracted, about one third were in fact classified as SPC by the ACE annotators. That is, even though those terms are not definite constructions, they were judged to be referring to specific entities in their discourse context. Error analysis reveals clearly the difficulties. As an illustration, consider the following mismatch case.

> Some 70 people were arrested Saturday as *demonstrators* clashed with *police* at the end of a major peace rally here. ...

In the above context, both *demonstrators* and *police* correspond to SPC entities accord-

entity class	SPC	GEN	USP	NEG	ATR
ng-gen(plural)	129(20.6%)	226(36.1%)	271(43.3%)	0(0.0%)	0(0.0%)
ng-gen(singular)	217(65.8%)	62(18.8%)	48(14.5%)	3(0.9%)	0(0.0%)

Table 3.2: *Comparison of bare plurals vs. bare singulars*

ing to ACE, as they refer to a particular set of people who actually participated in the stated event, but our method just labels them as generic.

However, in another context, such as the one below, the same terms would be correctly classed as generic by our method:

Demonstrators are often aggressive to *police*.

Among the mismatch results of table 3.1, most of the true SPC cases (i.e., an SPC entity wrongly recognised as GEN) can be viewed similarly.

Given difficulties of this kind, it is thought that an extraction method relying on a few types of constructions provides an evaluation baseline for generic term identification. In order to achieve significantly better and reliable results, we need to be backed by some means of discourse semantics.

3.2.3 A comparison of bare plurals vs. bare singulars

One interesting question was to see what kind of differences there are and how labeling precision varies between the bare plural and bare singular constructions identified as generic. Universally quantified constructions were removed for both singulars and plurals so that noun groups with every/any for singulars and all/any for plurals were excluded in labeling. And then, the precision of labeling was checked against the same test data set.

Table3.2 shows the labeling results for each case and Figure 3.5 displays them as a chart. It appears that we can draw quite a significant interpretation from this result. They show clear differences between singular and plrual cases. At first, the number of bare singulars captured as generic was 330, about 47% less than that of bare plurals which was 626, so that according to our labeling system, bare plural generic terms appeared about two times more frequently than bare singular generic terms. Overall precision of labeling was markedly lower for bare singulars, compared to bare plurals:

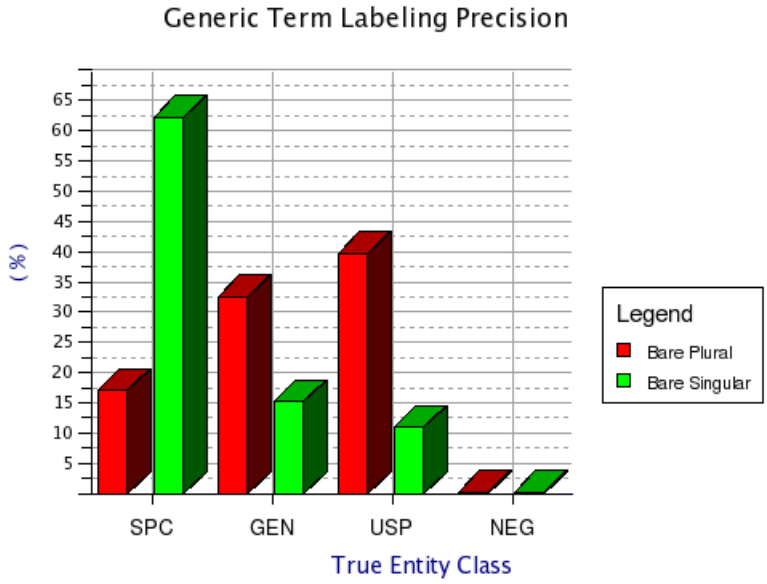

Figure 3.5: *Labeling Precision: Bare Plural vs. Bare Singular*

33.3% vs. 79.4%. The obvious reason is that the bare singulars were classifed into specific referentials(SPC) far more than the bare plurals.

A statistical inference was made to test if we have enough evidence to reject the hypothesis that the labeling result is statistically independent of the two types of constructions. A Chi-square test was applied to the 2×3 contingency table consisting of SPC, GEN and USP cases for each type of consturction with the negligible NEG and ATR cases ignored. (By the way, our transduction rules perform a filtering of NEGs, but a few negatively quantified bare singulars got through it. Of course the rules can be updated to additionally filter them out.)

The χ^2 test statistic (which follows $\chi^2(2)$ under null hypothesis) gives the value 197.28 with $p < 0.001$. So the test result indicates that the differences are highly significant and we may reject the null hypothesis of independence at very low significance level.

As shown in the table, 65.8% of identified bare singular GENs were SPCs in their true class, forming a major source of errors. An examination of such errors revealed a number of cases where bare singulars appear as modifier nominals for the noun group next to them which often consisted of named entities(NEs) such as company/person names. And they tended to be mixed up with GEN in labeling, although the correct

```
<s>
  <w ng-id="ng4" l="media" p="NNP" headn="yes" phr="B-NP" gen="probably">Media</w>
  <w ng-id="ng4" l="tycoon" p="NN" headn="yes" phr="I-NP" gen="probably">tycoon</w>
  <w ne="I-PER" ng-id="ng5" l="barry" p="NNP" headn="yes" phr="B-NP">Barry</w>
  <w ne="I-PER" ng-id="ng5" l="diller" p="NNP" headn="yes" phr="I-NP">Diller</w>
  <w p="IN">on</w>
  <w ne="I-DAT" ng-id="ng6" l="wednesday" p="NNP" headn="yes" phr="B-NP">Wednesday</w>
  <w vg-id="vg1" l="quit" p="VBD" headv="yes" phr="B-VP">quit</w>
  <w p="IN">as</w>
  <w ng-id="ng7" l="chief" p="NN" headn="yes" phr="B-NP" gen="probably">chief</w>
  <w p="IN">of</w>
  <w ne="I-ORG" ng-id="ng8" l="vivendus" p="NNP" headn="yes" phr="B-NP">Vivendi</w>
  <w ne="I-ORG" ng-id="ng8" l="universal" p="NNP" headn="yes" phr="I-NP">Universal</w>
  <w ne="I-ORG" ng-id="ng8" l="entertainment" p="NNP" headn="yes" phr="I-NP">
                                                              Entertainment</w>
  <w p=",">,</w>
   ...

</s>
```

Figure 3.6: *An example of bare singulars with genericity labeled*

annotation will be SPC because they denote the same entity as the named entities. The following is an illustration of such cases. A fragment of output from the pipeline's chunking module is shown for the sentence "*Media tycoon Barry Diller on Wednesday quit as chief of Vivendi Universal Entertainment, the entertainment unit of French giant Vivendi Universal whose future appears up for grabs.*" The *Media tycoon* and *chief* were spotted as generic but the correct class of entity associated with these is SPC.

In the gold standard annotation, these two terms actually correspond to one entity representing an individual person, Barry Diller. That is, they are both represented as <entity_mention> children of the same parent entity:

```
<entity CLASS="SPC" SUBTYPE="Individual" TYPE="PER"
                                         ID="AFP_ENG_20030319.0879-E4">
```

This kind of error can be considered not serious because the source of the problem is lack of performing a simple structual analysis. In this particular problem, the solution would be to check whether the noung group following such a modifying bare singluar has named entities and, if so, exclude the case from genericity labeling. We in fact excluded noun groups consisting of named entities in the labeling, but in this situation, the named entities belong to a noung group next to the current noun group so

they are independent in principle. As we are performing labeling on a chunk by chunk basis both for plurals and singulars, the same applies to both cases.

One thing to consider in this context would be that if the bare singular had a neighbouring noun group and assumed a modifier role for the noun group, it wouldn't automatically follow that the bare singular in question should be ruled out as non-generic. As was discussed in previous section 2.1, the generic term possesses characteristics of proper names. So, we could think of a sentence like the following:

> Man's amiable friend dogs are endangered.

Here, the noun group *dogs* of course denotes a kind and *amiable friend* has a modifying role. Granted that both of them denote the same entity, the bare singular *amiable friend* would be classed as GEN. (Strictly, this would be a counting construction rather than bare singular because *friend* represents a discrete entity, but it is just a singular noun without associated lexical information.)

On the other hand, successful identification of generic terms based on bare singulars included cases like these:

- at *home*
- in *prison*
- *offensive lineman*
- ...

Finally, as a general point, we may infer that the results obtained are in agreement with the assumption that bare plurals are the most common construction for denoting kinds in English(Cf. section 2.2.2). Although counting constructions with kind sense were not compared here, bare plurals are shown to be at least more likely to represent kinds than bare singulars, apart from the fact that they appeared more frequently in the text than bare singulars.

3.2.4 Discussion

The difficulties with generic term labeling were discussed in the above and we may understand they suggest us to do some discourse level processing to make systematic improvements. For example, in the case of

> Some 70 people were arrested Saturday as *demonstrators* clashed with *police* at the end of a major peace rally here.

it is seen that there appear some words indicating anchors in time and space. In our system, *Saturday* represents a separate noun group but it is independent of the noun groups *demonstrators* and *police*. Also the adverb *here* could be considered together in deciding the genericity of the two bare plural terms. Finally, taken together with these time-space anchor words, the verb group with head verb *clash* in past tense would indicate that the sentence describes an episodic event rather than a generic event. These points need to be made into preconditions that prevent the two terms *demonstrators* and *police* from being classed as GEN. Similarly, in the cases like

- Davao City international airport
- former chairman Alfred Taubman

the two neighbouring noun groups need to be checked together to see if either of them contains named entities and if so, both of them should be taken out of consideration. The processing in these lines would form a simple kind of context filtering in genericity labeling.

As a further discussion on the difficulties to overcome, there were certain words that often presented labeling errors. Among non-specific referential cases, such words as police, force, and everyone/anyone/everybody/anybody were labeled as USP or GEN, depending on the subtle discourse context. For example, in "... without really bothering *anybody*.", *anybody* is a USP and in "If *anybody* would like to debate the merits of ...", *anybody* is a GEN.

Finally, the gold standard annotations themselves seem to vary subtly and in some cases the distinctions are not so clear-cut, even considering the contexts involved. For example, the bare plurals 'officials' and 'police sources' are both labeled as generic by our method but in ACE, they were assigned different classes as USP and SPC, in the following contexts. The text fragments are shown below. So for the former, 'officials' is a USP and for the latter, 'police sources' is a SPC term. But it appears two contexts are very similar and the distinction between the two classes seem rather ambiguous. There were a few more cases like this that were encountered.

Text A: AFP_ENG_20030304.0250.sgm

DAVAO, Philippines, March 4 (AFP)

At least 19 people were killed and 114 people were wounded in Tuesday's southern Philippines airport blast, *officials* said, but reports said the death toll could climb to 30.

...

Text B: AFP_ENG_20030323.0020.sgm

NEW YORK, March 23 (AFP)

...

For the most part the marches went off peacefully, but in New York a small group of protesters were arrested after they refused to go home at the end of their rally, *police sources* said.

...

Chapter 4

Relation Extraction for Semantic Representation

4.1 The NLP Pipeline

The method we have adopted for extracting relations from text can be called 'semantic chunking'. Firstly, a syntactic chunker is used to identify noun groups and verb groups (i.e. non-recursive clusters of related words with a noun or verb head respectively). Second, we use a cascade of deterministic transduction rules to map from this shallow syntactic structure into first-order clauses; this cascade is conceptually very similar to the chunking method pioneered by Abney's Cass chunker(Abney, 1996).

The text processing framework we have used draws heavily on a suite of XML tools developed for generic XML manipulation (LTXML (Thompson et al., 1997)) as well as NLP-specific XML tools (LT-TTT (Grover and Tobin, 2006), LT-CHUNK (Finch and Mikheev, 1997)). We used recently upgraded and improved versions of the tools, most notably the program *lxtransduce* which performs rule-based transductions of XML structures. We have used *lxtransduce* both for the syntactic chunking and for construction of semantic clauses.

The main steps in the processing pipeline are as follows, including the final step of transforming semantic clauses into RDF(S) statements:

1. Words and sentences are tokenized.

2. The words are tagged for their part of speech using the CandC tagger(Clark and Curran, 2004) and the Penn Treebank tagset(Santorini, 1990).

3. The words are then reduced to their morphological stem (lemma) using Morpha(Minnen and Pearce, 2001).

4. The *lxtransduce* program is used to chunk the sentence into verb groups and noun groups

5. In an optional step, words are tagged as Named Entities, using the statistical CandC tagger trained on MUC data.

6. Generic and specific noun groups are identified, using a variety of morphological and lexical cues.

7. The output of the previous step is selectively mapped into semantic clauses in a series of steps, described in more detail below.

8. The XML representation of the clauses is converted using an XSLT stylesheet into RDF and RDFS statements.

4.1.1 The sequence of tasks performed by pipeline

The output of the syntactic processing is an XML structure containing word elements which are heavily annotated with attributes including POS tag and word lemma. Following CoNLL BIO notation(Sang and Buchholz, 2000), chunk information is recorded at the word level, through a chunk element flattening process. And then a tagging of semantic elements is performed as a preparatory step for semantic interpretation.

In the skeletal form of semantic tagging module, only the head words of chunks are retained, together with some function words (e.g., prepositions, certain adverbs, negation, quantifiers). Most of the XML mark-up is removed and replaced by tags that are intended to indicate coarse-grained semantic function (e.g., predicates versus arguments). Heads of noun groups and verb groups are assigned tags such as `<arg>` and `<rel>` respectively. In addition, other semantically relevant forms such as conjunction, negation, and prepositions are also tagged. Most other input and syntactic information is discarded at this stage. However, we maintain a record through shared indices of which terms belong to the same chunks. This is used, for instance, to build coordinated arguments and complex nominal arguments.

The next step is semantic interpretation. Regular expressions over the semantically tagged elements are used to compose clauses, using the heuristic that an <arg> immediately preceding a <rel> is the subject of the clause, while <arg>s following the <rel> are complements. As an option, since the heads of verb groups are annotated for voice, we can treat passive clauses appropriately, yielding a representation that is equivalent to the active congener.

4.2 An Extension for the Semantic Interpretation Module

4.2.1 Building an extended argument structure

An extension for argument structure was tried in order to deal with noun group in subject position with PP attachments. <prep-arg> structure was previously only treated as a complement of predicate <rel> element but a new argument structure was introduced to address the missing gap in the interpretation. This structure was used for both subject and object arguments.

Previously, pipeline's interpretation module primarily dealt with composing the argument from the nominals within a noun group chunk, or the complex argument consisting of arguments from different noun groups. Postmodifiers such as PP are grouped independently and interpreted as complement of a predicate <rel>, and they are wrapped up into <property> element.

However, a noun group followed by postmodifiers such as PP attachments couldn't be captured as a subject argument so as to form an <event> together with the <property>. For instance, in a sentence like

Flights from L.A. to New York have been cancelled due to fog.

The VP *have been cancelled due to fog* is captured as a property but the NP *Flights from L.A. to New York* does not form a single argument structure.

In order to group together such arguments with PP attachments, <arg-prep-arg> structure was introduced. If we loosely think of it in terms of context-free grammar, in addition to nominals expressed by

$$Nominal \rightarrow Noun$$
$$Nominal \rightarrow Noun \quad Nominal$$

complex nominals from the following rule is now represented as one argument.

$$Nominal \rightarrow Nominal\ PP$$

In general, the nominal rule is of course recursive in nature with PP involving nominals too, but a finite state grammar approximation to the context-free grammar is possible, for example, if we limit the number of expansions for the inside nominals(Jurafsky and Martin, 2000).

For our present purpose, only iterative expansion of nominal into nominal with PP is represented by the regular expression, without any nominal expansions inside PP. However, it's not something like a complete parsing result and there is room for reinterpreting the composed argument structure at later stages.

Syntactically, it may be said that this type of `<arg-prep-arg>` structure captures NP with PP attachment by default. In case they are complements of some predicate `<rel>`, the structure will interpret NP + PP as NP with low attachment of PP. This is an adhoc way to deal with complex argument structures but there is room for flexible reinterpretation afterwards. That is, for example, we could take into account lexical information on specific verb frames and reinterpret it as high attachment of PP when appropriate.

Also adjectives qualifying head nouns were given a semantic tag as `<mod>` and were added into argument structure so that they can be used later as additional data for logical representation. Accordingly, an extended set of regular expressions was implemented to compose semantic clauses with them.

This is necessary because generic terms and corresponding noun groups encompass both modifiers and head nouns inside them and the information on modifiers should be kept for proper semantic representation in later stages. As qualification implies some sort of specialisation of information, loss of such information would result in partial semantic representation of properties which doesn't generally hold for the qualifier omitted terms. The following are a few illustrative examples which show the interpreted output of the pipeline for each test sentence.

```
Wine from Spain tasted good.
<s>
  <event>
    <arg-prep-arg>
      <arg>
```

```
        <arg0 ng-id="ng1">wine</arg0>
      </arg>
      <prep-arg>
        <prep>from</prep>
        <arg>
          <arg0 ng-id="ng2" type="LOC">Spain</arg0>
        </arg>
      </prep-arg>
    </arg-prep-arg>
    <property voice="act">
      <rel voice="act" vg-id="vg1">taste</rel>
      <mod>good</mod>
    </property>
  </event>
  <delim />
</s>

The agents of General Noriega in America did the operations.
<s>
  <event>
    <arg-prep-arg>
      <arg>
        <arg0 ng-id="ng1">agent</arg0>
      </arg>
      <prep-arg>
        <prep>of</prep>
        <arg>
          <arg0 ng-id="ng2" type="ORG">General_Noriega</arg0>
        </arg>
      </prep-arg>
      <prep-arg>
        <prep>in</prep>
        <arg>
          <arg0 ng-id="ng3" type="LOC">America</arg0>
        </arg>
      </prep-arg>
    </arg-prep-arg>
    <property voice="act">
      <rel voice="act" vg-id="vg1">do</rel>
      <arg>
        <arg0 ng-id="ng4">operation</arg0>
      </arg>
    </property>
  </event>
  <delim />
```

```
</s>

Former Panamanian leader General Manuel Antonio Noriega's defence
against drug trafficking charges in Miami gained ground this week.
<s>
  <event>
    <arg>
      <mod>Former</mod>
      <mod>Panamanian</mod>
      <arg0 ng-id="ng1">leader</arg0>
    </arg>
    <arg-prep-arg>
      <arg>
        <arg1>
          <arg0 ng-id="ng3" type="ORG" poss="yes" my-head="">
          General_Manuel_Antonio_Noriega</arg0>
          <arg0 ng-id="ng2">defence</arg0>
        </arg1>
      </arg>
      <prep-arg>
        <prep>against</prep>
        <arg>
          <arg0 ng-id="ng4">drug_trafficking_charges</arg0>
        </arg>
      </prep-arg>
      <prep-arg>
        <prep>in</prep>
        <arg>
          <arg0 ng-id="ng5" type="LOC">Miami</arg0>
        </arg>
      </prep-arg>
    </arg-prep-arg>
    <property voice="act">
      <rel voice="act" vg-id="vg1">gain</rel>
      <arg>
        <arg0 ng-id="ng6">ground</arg0>
      </arg>
      <arg>
        <arg0 ng-id="ng7" type="DAT">this_week</arg0>
      </arg>
    </property>
  </event>
  <delim />
</s>
```

4.2.2 Embedded relation extraction

Embedded relations was meant for those relations that can be extracted from any post-nominal modifiers with a predicate component such as VBN and VBG, as well as those inside a relative clause. They can be identified as head verb inside a verb group chunk, with their voice information annotated. It was considered that these relations had better be considered separately from the predicate of main clause and the predicates associated with embedded relations were labeled as `<argrel>` since their scope is local to the embedded relation.

Additionally it will be necessary to duplicate arguments that had an `<argrel>` predicate related to them so that those arguments are not missed in the representation of main clause. Likewise, the argument in object position that has an `<argrel>` needs to be duplicated so that one is captured as main clause's argument and the other as argument associated with the `<argrel>`.

Some heuristic rules in *lxtransduce* grammar were developed to copy the subject argument having an `<argrel>` into the immediate subject position of the main clause predicate, skipping in-between elements. Also, rules were implemented for copying the object argument into neighbouring position so that an `<event>` can be assigned with the copied argument and related `<argrel>` predicate.

```
The boy called his parents standing behind.
<s>
  <event>
    <arg>
      <arg0 ng-id="ng1">boy</arg0>
    </arg>
    <property voice="act">
      <rel vg-id="vg1" voice="act">call</rel>
      <arg role="obj">
        <arg0 ng-id="ng2">parent</arg0>
      </arg>
    </property>
  </event>
  <event>
    <arg role="obj">
      <arg0 ng-id="ng2">parent</arg0>
    </arg>
    <property voice="act">
      <argrel voice="act" vg-id="vg2">stand</argrel>
      <pred>behind</pred>
```

```
    </property>
  </event>
  <delim />
</s>
```

Using these heuristics, simple control cases can be dealt with as a special case, like the following.

```
The textbooks have been reported to be very popular among scientists.
<s>
  <event>
    <arg role="subj">
      <arg0 ng-id="ng1">textbook</arg0>
    </arg>
    <property voice="pass">
      <rel vg-id="vg1" voice="pass">report</rel>
    </property>
  </event>
  <inf>to</inf>
  <event>
    <arg role="subj">
      <arg0 ng-id="ng1">textbook</arg0>
    </arg>
    <property voice="act">
      <rel voice="act" vg-id="vg2">be</rel>
      <pred>very</pred>
      <mod>popular</mod>
      <prep-arg>
        <prep>among</prep>
        <arg>
          <arg0 ng-id="ng2">scientist</arg0>
        </arg>
      </prep-arg>
    </property>
  </event>
  <delim />
</s>
```

Chapter 5

RDF(S) Representation of Generics

5.1 RDF(S) Ontology Language for Semantic Web

The RDF(Resource Description Framework) provides a foundation for representing and processing metadata in Semantic Web(Antoniou and Harmelen, 2004; Passin, 2004). While XML is a universal metalanguage for defining markup, it does not provide any means of specifying meaning or semantics of data. So in XML, equivalent descriptions of a resource can be made in different syntactic structures, for example, using different nestings of element tags.

```
<movie title="The Lord of the Rings(I)">
  <director>Peter Jackson</director>
</movie>
<director name="Peter Jackson">
  <movie_directed> The Lord of the Rings(I)</movie_directed>
</director>
<movie_item>
  <title>The Lord of the Rings(I)</title>
  <director>Peter Jackson</director>
</movie_item>
```

In the above, we can notice that three different XML elements contain essentially same information about a movie and its director but there are no semantic differences imposed by XML with respect to different element nestings. On the other hand, RDF as an abstract data model provides a standardised semantics for resource description. RDF has been given a syntax in XML and inherits the benefits associated with XML, although

XML-based syntax is not a necessary requirement and other syntactic representations are possible.

The basic concepts for RDF are resources, properties and statements. A resource is a *thing* we talk about in the statement; they are objects like movie, director, actors, etc. Every resource is supposed to have a URI(Uniform Resource Identifier) which can be a URL or some other kind of unique identifier(W3C URI Interest Group, 2001).

The RDF properties describe relations between resources as explained in 1.1. They are basically a binary predicate giving truth values of a relation between two resources. As RDF Schema makes it clearer, everything in RDF can be thought of as a resource, and properties are also a kind of resource.

The RDF statements make assertions about the properties of resources. They can be viewed in three different ways: as triples, as graph, or as XML document in RDF-XML syntax.

RDF is domain independent and does not make any constraints so that one can state anything about anything. RDF Schema(RDFS) is designed for specifying constraints for RDFstatements so that they can make sense and it thus enables more proper ontological description of resources.

RDFS introduces *classes*. A class is a set of individual objects called instances of that class. We can talk about individuals belonging to a particular domain, say dogs: (e.g.) the dog of my neighbour. (This can be done based on RDF.) But now we can equally talk about groups of such individuals: (e.g.) Labrador, Greyhound, Harrier, etc. The class can be importantly used in RDFS to impose the desirable constraints on what can be stated in an RDF document, which is analogous to restricting data types for specific operators in general programming languages. In RDF(S), we can restrict the range and the domain of any property.

Along with the class, the class hierarchy is defined so that the class Dog would be a subclass of Animal class, as every dog is an animal. This represents a universally quantified statement that every individual of Dog class also belongs to their superclass Animal. Also, it effectively leads to the *inheritance* mechanism regarding the property domain/range: that is, if a property makes a domain restriction on the superclass, the property can also be applied to all individuals of the subclass.

Similarly, the hierarchical relationship between properties can be made also. P is a subproperty of Q if $P(x,y)$ implies $Q(x,y)$. As RDFS creates and implements this se-

mantics about class hierarchy and inheritance, it forms a primitive *ontology language*.

5.2 RDF Representation of the Extracted Relations

5.2.1 Experimental results with Wikipedia corpus

The relations returned by our semantic interpretation module can be transformed into RDF statements by simply mapping the relation into the predicate of an RDF triple. Figure 5.1 shows a list of relations captured as 4 events. They were extracted from 4 corresponding sentences of a sample article in this section. As can be seen, they contain at least one genericity labeled argument so that they are recognised as generic sentence and selected for further RDF representation.

For converting these events to RDF triples, XSLT stylesheets were implemented which takes the events as input and transforms them to an intermediate format for inspection or a set of triples, and finally to the RDF-XML document itself. One of the stylesheets is included in the Appendix B.

Using the XPATH language[1], the stylesheet can do a wonderful job of accessing parts of XML structure and creating new documents using their content by applying a set of templates. In our context, the complex `<arg>` structure should be addressed effectively so that such query as "complex nominal or simple nominal argument whose last `<arg0>` bears genericity attribute" needs to be implemented as a compact XPATH query.

In the RDF transformation, there are problems with selecting suitable subject/object arguments because the arguments in the relations are often composite in structure and associated with prepositional phrases. The argument preceding a predicate is turned into the subject of an RDF triple and the argument immediately following the predicate into the object. As we are focussing on generic sentences, we filter out clauses where none of the arguments are identified as generic.

After suitable arguments are determined, RDF statements can be composed using suitable RDF vocabulary. The statements can be made up based on two different layers, namely RDF and RDFS layers. In the RDF layer, we can make statements about individuals where the participants should be instances rather than classes. On the other hand, if arguments are classes, they should involve RDFS. For instance, *Dogs chase*

[1] http://www.w3.org/TR/xpath

```
<event>
  <arg><arg0 ng-id="ng58" gen="probably">rodent</arg0></arg>
  <property voice="act">
    <rel vg-id="vg24" voice="act">have</rel>
    <arg><arg0 ng-id="ng59">incisor</arg0></arg>
  </property>
</event>

<event>
  <arg>
    <mod>early</mod>
    <arg0 ng-id="ng111">rodent</arg0>
  </arg>
  <property voice="act">
    <rel vg-id="vg48" voice="act">resemble</rel>
    <arg><arg0 ng-id="ng112" gen="probably">squirrel</arg0></arg>
  </property>
</event>

<event>
  <arg><arg0 ng-id="ng78" gen="probably">rodent</arg0></arg>
  <property voice="act">
    <rel vg-id="vg36" voice="act">lack</rel>
    <arg><arg0 ng-id="ng79" gen="probably">canine</arg0></arg>
  </property>
</event>

<event>
  <arg><arg0 gen="probably" ng-id="ng137">rodent_fossils</arg0></arg>
  <property voice="act">
    <rel vg-id="vg59" voice="act">appear</rel>
    <prep-arg>
      <prep>in</prep>
      <arg><arg0 type="LOC" ng-id="ng138">Australia</arg0></arg>
    </prep-arg>
  </property>
</event>
```

Figure 5.1: *Relations captured as* XML *event structure*

foxes. would be mapped into something like this.

```
(ex:dog  rdf:type  rdfs:Class)
(ex:fox  rdf:type  rdfs:Class)
(ex:chase  rdf:type  rdf:Property)
(ex:chase  rdfs:domain  ex:dog)
(ex:chase  rdfs:range  ex:fox)
```

In the following, we illustrate our approach with a set of RDFS statements extracted from a sample Wikipedia article. For simplicity, we map a head noun of the argument into an RDFS class (an instance of `rdfs:Class`) and the predicate of relation into an RDF property (an instance of `rdf:Property`). For class names, we use the word's lemma as the argument content, which may be attached to any contingent modifier words returned by our semantic interpretation module.

```
(wiki:rodent  rdf:type  rdfs:Class)
(wiki:incisor rdf:type  rdfs:Class)
(wiki:rodent_early  rdf:type  rdfs:Class)
(wiki:squirrel  rdf:type  rdfs:Class)
(wiki:rodent_fossils  rdf:type  rdfs:Class)

(wiki:lack  rdf:type  rdf:Property)
(wiki:have  rdf:type  rdf:Property)
(wiki:resemble  rdf:type  rdf:Property)
(wiki:appear_in  rdf:type  rdf:Property)

(wiki:rodent  wiki:lack  wiki:canine)
(wiki:rodent  wiki:have  wiki:incisor)
(wiki:rodent_early  wiki:resemble  wiki:squirrel)
(wiki:rodent  wiki:appear_in  wiki:rodent_fossils)
(wiki:rodent_fossils  wiki:appear_in  wiki:Australia)
```

Although omitted in the above, we normally add domain/range restrictions for properties, as part of the transformation.

```
(wiki:lack  rdfs:domain  wiki:rodent)
(wiki:lack  rdfs:range  wiki:canine)
(wiki:have  rdfs:domain  wiki:rodent)
```

```
(wiki:have  rdfs:range  wiki:incisor)
                ...
```

Besides the conventional triple notation, W3C offers another language called Notation 3 or N3[2] for describing RDF triples. N3 is a language which was designed as a compact and readable alternative to RDF's XML syntax. At the same time, N3 is more expressive. The above representation can be more compactly reexpressed using N3 as was done in Figure 5.3. (The `a` is an abbreviation for `rdf:type` and the semicolon `;` introduces another property of the same subject.)

Terms like `rodent_early` and `rodent_fossils` are constructed at different stages in the processing pipeline: complex nominals (*rodent fossils*) are combined into semantic primitives (`rodent_fossils`) once the noun group chunking results are available, while the attachment of adjectival modifiers (*early rodents*) is carried out in the final stylesheet processing for RDF transformation. Figure 5.2 displays a screenshot of the RDF graph corresponding to the above RDF triples, after the RDF/XML has been parsed on W3C RDF Validation Service site[3].

Finally, the source RDF/XML document, also produced by the stylesheet, can be seen in Figure 5.4. For the document, RDF namespace prefixes should be given properly: they are specified inside the `rdf:RDF` element, as well as declared as entities for use in the value of `rdf:about` attribute. The `rdf`, `rdfs` and `dc` are well-known RDF, RDFSand Dublin Core namespaces respectively, and `wiki` is a local namespace newly defined for the Wikipedia vocabulary created from our processing.

The statements are extracted from sentences that have the generic noun groups either as subject or object argument, and we can see that they contain plausible knowledge about the domain of rodents. The next goal will be then to scale-up this extraction to process a much larger set of sentences, leading to connections between the triples which will achieve the effect of knowledge integration, one of the ends of RDF.

5.2.2 Quality evaluation of the extracted RDF triples

The transformed RDF triples are supposed to describe generic and common sense statements. However, due to lack of existing methods for objective assessment, human judges should be involved in evaluating the quality of these triples.

[2]http://www.w3.org/2000/10/swap/Primer.html
[3]http://www.w3.org/RDF/Validator/

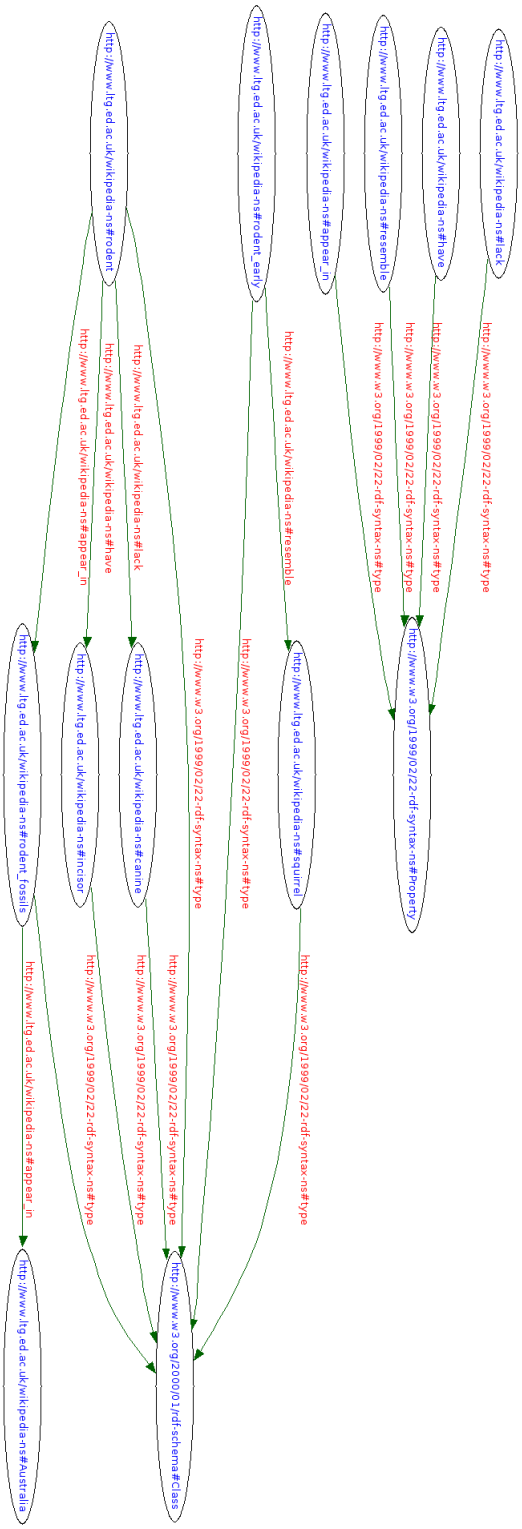

Figure 5.2: *RDF(S) statements as a graph*

```
wiki:rodent      a           rdfs:Class ;
             wiki:lack       wiki:canine ;
             wiki:have       wiki:incisor ;
             wiki:appear_in  wiki:rodent_fossils .

wiki:incisor a  rdfs:Class .

wiki:rodent_early      a          rdfs:Class ;
                   wiki:resemble  wiki:squirrel .

wiki:squirrel  a  rdfs:Class .

wiki:rodent_fossils      a           rdfs:Class ;
                     wiki:appear_in  wiki:Australia .

wiki:lack        a     rdf:Property .
wiki:have        a     rdf:Property .
wiki:resemble    a     rdf:Property .
wiki:appear_in   a     rdf:Property .
```

Figure 5.3: RDF *triples in N3 format*

For this purpose, two human judges took part in the evaluation performed. Each went through a set of RDF triples derived as common sense triples, marking those as 'YES' or 'NO' for the degree of making sense. Table 5.1 summarises the result. Total 585 triples from 200 random Wikipedia articles were evaluated by two judges. We obtained the kappa statistic as a measure of agreement.

The kappa statistic(Cook, 1998) assesses the degree of agreement between subjects after eliminating chance agreement component from the observed agreement. The two judges had 70% observed agreement value and the κ value was 0.41 which indicates 'moderate agreement' between judges. (κ in the range of 0.40-0.59 receives the moderate agreement interpretation.)

Marking Result	**Yes**	**No**
Yes	209	126
No	48	202

Table 5.1: *Quality assessment of RDF triples by two judges*

The distribution of the number of triples per each article was also of interest. Ap-

```
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE rdf:RDF [
    <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
    <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
    <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
    <!ENTITY wiki 'http://www.ltg.ed.ac.uk/wikipedia-ns#'>
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:wiki="http://www.ltg.ed.ac.uk/wikipedia-ns#">

<rdfs:Class rdf:about="&wiki;rodent"/>
<rdf:Property rdf:about="&wiki;lack"/>
<rdfs:Class rdf:about="&wiki;canine"/>
<rdf:Description rdf:about="&wiki;rodent">
  <wiki:lack rdf:resource="&wiki;canine"/>
</rdf:Description>

<rdf:Property rdf:about="&wiki;have"/>
<rdfs:Class rdf:about="&wiki;incisor"/>
<rdf:Description rdf:about="&wiki;rodent">
  <wiki:have rdf:resource="&wiki;incisor"/>
</rdf:Description>

<rdfs:Class rdf:about="&wiki;rodent_early"/>
<rdf:Property rdf:about="&wiki;resemble"/>
<rdfs:Class rdf:about="&wiki;squirrel"/>
<rdf:Description rdf:about="&wiki;rodent_early">
  <wiki:resemble rdf:resource="&wiki;squirrel"/>
</rdf:Description>

<rdf:Property rdf:about="&wiki;appear_in"/>
<rdfs:Class rdf:about="&wiki;rodent_fossils"/>
<rdf:Description rdf:about="&wiki;rodent">
  <wiki:appear_in rdf:resource="&wiki;rodent_fossils"/>
</rdf:Description>

<rdf:Description rdf:about="&wiki;rodent_fossils">
  <wiki:appear_in rdf:resource="&wiki;Australia"/>
</rdf:Description>
</rdf:RDF>
```

Figure 5.4: RDF/XML *document produced by the* XSLT *stylesheet*

No. of triples	Frequency
0	1629
1	548
2	274
3	183
4	117
5	77
6	58
7	41
8	20
9	18
10	14

Table 5.2: *Frequency of number of triples of a document*

parently RDF triples were concentrated on relatively few articles and there often occurred articles from which no RDF triple was derived. So the distribution seemed highly skewed to the right. To see the distribution more closely, a random set of 3,272 articles were processed to produce the generic statement RDF triples in the same manner. The maximum number of triples within a document was 103 so that the range for the distribution was 0–103. Certain types of documents were taken out of consideration and these were articles producing repetitive sets of triples. For example, the text of an article introducing a city runs like:

> C is a city located in L. It has a total population of P. The city has a total area of A. There are X households out of which 35.4% have children under the age of 18, 53.0% are married couples living together, ...

Then, they derived such sets of stereotyped triples, which were thought not useful for our purpose. For the remaining 3,080 files, the number of articles that produced n RDF triples are shown for each $n \leq 10$ in Table 5.2. The average number of triples per document was 1.89. Except for the decaying tail part with some fluctuation ($10 < n \leq 103$), it is interesting to see that the empirical distribution has a nice decrease like a parametric distribution such as geometric distribution, but the rate of decrease is slower than exponential at the beginning.

5.3 Discussion on Other Logical Frameworks for Generics

5.3.1 Exceptions, Default Reasoning and RDF(S)

Not surprisingly, it is difficult to represent the semantics of generics in as simple a framework as RDFS. As discussed in 2.1, a generic statement has the characteristics of *default statement*, in the sense that generics admit exceptions without changing their truth condition. This is entirely lacking in RDF(S) logical framework.

In view of general logical reasoning framework, acceptance of such exceptions should be done using concepts and methods of common sense reasoning(Mueller, 2006). For example, in common sense reasoning, a *qualification* is a condition that prevents an event from having its intended effects or that prevents an event from occurring. Using *preconditions* provides a partial solution to the qualification problem. An event whose precondition is not satisfied may occur but the event is not supposed to have the intended effect.

The *default reasoning* can be used to provide a way of specifying qualifications, where abnormality predicates are introduced and cancelation axioms are added whenever one wishes to add a qualification. Thus, the default reasoning can be used to represent default properties that are true or false by default, with some qualifications given as cancellation axioms.

For example, "*Elephants are grey.*" may be treated as a default property as follows:

$$\neg Abnormal(a,t) \Rightarrow HoldsAt(Grey(a),t) \tag{5.1}$$

$Abnormal(a,t)$ is the abnormality predicate and $HoldsAt(Grey(a),t)$ is a (event calculus) predicate which simply means the property Grey(a) holds at time t; $Grey(a)$ is a term of *fluent* sort, in terms of sorted first-order logic(Blackburn and Bos, 2005).

But if we'd take into account the existence of White elephants, we should add a cancelation axiom representing when an elephant is not grey.

$$HoldsAt(WhiteElephant(a),t) \Rightarrow Abnormal(a,t) \tag{5.2}$$

Here, variable a is for the elephant sort and the formula means that if a is an white elephant at time t then a is abnormal in a way preventing a conclusion that it is grey. So, equipped with this mechanism of logical reasoning, the generic statement "*Elephants*

are grey." could be reinterpreted as "*Elephants are grey (except for ...).*" now having a means to deal with abnormal cases.

5.3.2 OWL DL

As is well known, there are some other limitations in using RDF(S). First, semantics of RDF does not assign any logical role to negation, so we cannot adequately express statements like *Foxes are not pack animals*: *not pack animals* here had better be expressed as a complementary class of *pack animals*. In other words, it's not possible to deal with negation of a predicate by a logical construct. Second, we cannot define a class like `rodent_fossils` as the intersection of the classes `rodent` and `fossils`.

In RDF(S), Boolean operators(and, or, not) cannot be expressed by the language. Therefore, using RDF(S) we only have to code a negated relation separately. These shortcomings provide a motivation for using OWL DL as a representation framework in place of RDF(S). In OWL DL (as well as in OWL FULL), Boolean class expressions are allowed using the vocabulary of owl:intersectionOf, owl:unionOf and owl:complementOf.

5.3.3 OWL Full

In the above, generic statement as a default statement was discussed. But even using the default reasoning, it may be difficult to specify arbitrary exceptions of individuals belonging to some class. Another possible way of accommodating exceptions would be to represent generic statement as predication over classes rather than individuals, where classes correspond to the kinds denoted by generic terms. In such a way, their truth condition will become independent of individual exceptions.

RDF counterpart for implementing it will be then a generic statement being modelled by a triple in which predicate `P` is only defined over some RDFS classes rather than RDF individuals. In the triple notation, that would lead to the following statements.

```
(P  rdf:type  rdf:Property)
(P  rdfs:domain  rdfs:Class)
```

That is, the property `P` takes only an RDFS class as its subject argument. `rdfs:Class` represents the class of all classes and any RDFS class is an instance of `rdfs:Class`. As `rdfs:Class` is a *class* anyway, it's an instance of `rdfs:Class` itself.

However, in some cases it would be desirable to impose some further restriction on the domain of `P`. For example, in the sentence "Dogs *chase* foxes." *chase* would be turned into a predicate `P` in our representation. But it would make sense only when the subject argument for the verb *chase* belongs to a class of living things like animals or humans. If we add a domain restriction like

```
(wiki:chase  rdf:type  rdf:Property)
(wiki:chase  rdfs:domain wiki:fox)
```

then it implies that we cannot but let `wiki:chase` be a predicate over individuals, namely those of `wiki:fox`. There can be no such thing like class of all animal classes as a subclass of `rdfs:Class`.

If we aim to implement it, we should turn to the OWL FULL ontology language(Dean and Schreiber, 2004), going beyond the OWL DL language. OWL FULLincorporates semantic extension of RDF(S) and enables constructs for *class as an instance*. So a class can be treated as an individual of some (meta)class.

One relevant point here would be in connection with the nature of natural language predicates discussed before. It is known that all natural language predicates which can be meaningfully applied to individuals are equally well applied to kinds but not vice versa. There do exist a number of predicates that are exclusively used for kinds. They are *extinct*, *common*, *widespread*, etc., as mentioned in section 2.1. A sentence like "*My neighbour's dog is extinct.*" does not make sense as long as *my neighbour's dog* has the individual sense.

So these predicates would naturally correspond to the predicates over classes only, not over individuals. But they could be considered rather exceptional and for the vast majority of predicates, such a logical representation may not be an elegant idea. However, differentiating the two contexts of applying a predicate and exploring logical relationships between them may be a subject worth paying attention to in the representation of generics.

Chapter 6

Conclusions

Through this project, we developed a prototype system that extracts generic statements from unrestricted natural language text and convert them to logical statements in RDF(S) in a fully automatic manner.

Although some simplifying assumptions were made in the generic term identification, the implementation was done in a general framework. The generic term labeling and RDF transformation parts were integrated into the overall process of pipeline to produce the target logical representations starting from raw text inputs.

The developed modules would be readily fit into extended implementations which will utilise more of the processing results from the pipeline at each stage. As items of the next extension, we may include the following.

- Indefinite noun groups in the generic term labeling: especially we can focus on those noun groups after the verb 'be' as likely kind-denoting terms: (e.g.) *a city* in "*London is a city.*" According to the ACE annotation, these are attributives(ATR) but they may as well be treated as a generic term so as to yield representations of class membership or subclass relationship between classes.

- Mapping the extracted relations into a few conceptual categories of relation such as subclass, possession, location and purpose relations.

This research is still in its early stages, but we believe it offers an innovative way of contributing to the construction of ontologies for the Semantic Web. Our approach has a virtue of its linguistic focus on generics, since this class of sentences would map naturally into the types of information to be embodied in ontologies. Using this

methodology to help generate semantic annotation for the Semantic Wikipedia project will be both exciting and useful.

In future, this work should be combined into a layer of tasks which develop more standardised representation of semantic relations such as Dynamic Ontology Refinement(Fiona McNeill, 2003). Another future avenue to explore is application of the Active Learning(AL) techniques(Thompson et al., 1999) in order to investigate how the quality of representation can be improved upon through the combination of human validation and automatic semantic annotation.

There is a continuum of other possible approaches for carrying out the kind of task we are exploring in this work. At one extreme are very shallow techniques for extracting rather impoverished semantic information, while at the other, one can use deep statistical parsers(Bos et al., 2004) to build much richer semantic structures. Since all wide-coverage techniques are error-prone, we believe that the most urgent task is to find the right balance between accuracy and volume in knowledge extraction techniques, and develop semantically oriented techniques for filtering and cleaning noisy semantic data.

In addition, as already discussed in the above, we believe it would be helpful to go beyond RDF(S) by mapping directly to an OWL ontology. By using human language technologies to semantically annotate Wikipedia, we show that utilizing the collective intelligence of ordinary users of the Web can provide a way to bootstrap tremendous amounts of common sense data that otherwise would take decades to engineer.

In concert with human validation to filter out errors and give feedback for refinements, this line of research would make the dream of a fully-annotated Semantic Wikipedia more feasible. An annotated Semantic Wikipedia would lead in turn to a greater network effect for the Semantic Web, so the Semantic Web would be used not only as a tool to reason about ontologies in specialised domains, but also to reason about the myriad of facts and vagaries that can be found in everyday life.

Appendix A

Deterministic Transduction Rules for Generic Term Labeling

```
<!DOCTYPE rules SYSTEM "lxtransduce.dtd">

<rules apply="ng-gentag" type="xml">

<lexicon name="exceptions" href="exceptions.lex"/> <lexicon
name="detsing" href="detsing.lex"/> <lexicon name="exceptions_headn"
href="exceptions_headn.lex"/> <lexicon name="numbers"
href="numbers.lex"/>

<rule name="ng-exclude">
  <query match="w">
    <constraint test="not(@c~'ord|pcent|(.?quote)|dots|what|abbr|sym')"/>
    <constraint test="not(@ne)"/>
    <constraint test="not(@l~'^[a-z]$')"/>
  </query>
</rule>

<rule name="ng-begin">
  <and>
    <query match="w[@phr='B-NP']"/>
    <query match="w[not(@p~'CD|(PRP.?)|PDT|POS|JJR|RBR')]"/>
    <query match="w[not(exceptions()|numbers())]"/>
    <ref name="ng-exclude"/>
  </and>
</rule>

<rule name="ng-begin-detsing">
  <query match="w">
    <constraint test="detsing()[cat='determiner' and cat='singular']"/>
  </query>
</rule>

<rule name="ng-begin-final">
  <and>
    <query match="w[@phr='B-NP' and @p~'NNS|NNPS' and @headn='yes']"/>
```

```
    <query match="w[not(exceptions_headn()|numbers())]"/>
    <ref name="ng-exclude"/>
  </and>
</rule>

<rule name="ng-inside">
  <and>
    <query match="w[@phr='I-NP']"/>
    <query match="w[not(@p~'CD|(PRP.?)|PDT|POS|JJR|RBR')]"/>
    <query match="w[not(exceptions()|numbers())]"/>
    <ref name="ng-exclude"/>
  </and>
</rule>

<rule name="ng-inside-final">
  <and>
    <query match="w[@phr='I-NP' and @p~'NNS|NNPS' and @headn='yes']"/>
    <query match="w[not(exceptions_headn()|numbers())]"/>
    <ref name="ng-exclude"/>
  </and>
</rule>

<rule name="ng-inside-final-sing">
  <and>
    <query match="w[@phr='I-NP' and @p~'NN|NNP' and @headn='yes']"/>
    <query match="w[not(exceptions_headn()|numbers())]"/>
    <ref name="ng-exclude"/>
  </and>
</rule>

<rule name="ng-outside">
  <query match="w[@phr!='I-NP' or not(@phr)]"/>
</rule>

<rule name="ng-gentag" attrs="gen='probably'">
  <first>
    <seq>
      <ref name="ng-begin-final"/>
      <ref name="ng-outside" suppress="true"/>
    </seq>

    <seq>
      <ref name="ng-begin"/>

      <repeat-until name="ng-inside">
     <ref name="ng-inside-final"/>
      </repeat-until>
      <ref name="ng-inside-final"/>

      <repeat-until name="ng-inside-final">
     <ref name="ng-outside"/>
      </repeat-until>
    </seq>

    <seq>
      <ref name="ng-begin-detsing"/>
```

```
      <repeat-until name="ng-inside">
     <ref name="ng-inside-final-sing"/>
      </repeat-until>
      <ref name="ng-inside-final-sing"/>

      <repeat-until name="ng-inside-final-sing">
     <ref name="ng-outside"/>
      </repeat-until>
    </seq>
  </first>
</rule>

</rules>
```

Appendix B

A Stylesheet for RDF Transformation of Relations

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="text"/>

<xsl:template match="/">

  <xsl:text>-- FILE_ID: </xsl:text>
  <xsl:value-of select="s/@file_ID"/>
  <xsl:text>&#10;</xsl:text>

  <xsl:apply-templates select="//event"/>
  <xsl:text>&#10;</xsl:text>

</xsl:template>

<xsl:template match="arg">
  <xsl:copy>
    <xsl:for-each select=".//arg0[1]|.//arg0[preceding-sibling::arg0][1][@role!='mod']">
     <xsl:choose>

      <xsl:when test="@role='mod'">
        <xsl:value-of select="normalize-space(following-sibling::arg0)"/>
        <xsl:text>_</xsl:text>
        <xsl:value-of select="normalize-space(.)"/>
      </xsl:when>

      <xsl:otherwise>
        <xsl:value-of select="normalize-space(.)"/>

    <xsl:if test="preceding-sibling::mod">
      <xsl:text>_</xsl:text>

      <xsl:for-each select="preceding-sibling::mod">
        <xsl:value-of select="normalize-space(.)"/>
        <xsl:if test="following-sibling::*[name()='mod']">
```

```
          <xsl:text>_</xsl:text>
        </xsl:if>
      </xsl:for-each>
    </xsl:if>
    <xsl:if test="following-sibling::arg0 or following-sibling::arg1">_</xsl:if>
      </xsl:otherwise>

     </xsl:choose>
    </xsl:for-each>
  </xsl:copy>
</xsl:template>

<xsl:template match="prep-arg">
  <xsl:copy>
    <xsl:value-of select="concat(prep,'_')"/>
    <xsl:apply-templates select="arg"/>
    <xsl:if test="following-sibling::prep-arg">_</xsl:if>
  </xsl:copy>
</xsl:template>

<xsl:template match="event">

  <xsl:if test="arg[following-sibling::property]//arg0[not(.='')]
                |arg-prep-arg[following-sibling::property]/arg//arg0[not(.='')] or
                property/arg//arg0[not(.='')]|property/arg-prep-arg/arg//arg0[not(.='')]
                |property/prep-arg/arg//arg0[not(.='')]">

    <xsl:text>( wiki:</xsl:text>
    <xsl:apply-templates select="arg|arg-prep-arg/arg"/>
    <xsl:text>  </xsl:text>

    <xsl:text>wiki:</xsl:text>
      <xsl:for-each select="property/*[self::rel or self::argrel]">
        <xsl:if test="./neg"><![CDATA[not_]]></xsl:if>
        <xsl:value-of select="normalize-space(.)"/>
        <xsl:if test="following-sibling::pred">
          <xsl:text>_</xsl:text>
          <xsl:value-of select="following-sibling::pred"/>
        </xsl:if>
        <xsl:if test="@voice='pass'"><![CDATA[_PASS]]></xsl:if>
      </xsl:for-each>
    <xsl:text>  </xsl:text>

    <xsl:text>wiki:</xsl:text>
    <xsl:apply-templates select="property/arg|property/arg-prep-arg/arg"/>

    <xsl:for-each select="property/*[self::mod]">
      <xsl:value-of select="normalize-space(.)"/>
      <xsl:if test="following-sibling::mod">_</xsl:if>
      <xsl:if test="following-sibling::prep-arg">_</xsl:if>
    </xsl:for-each>

    <xsl:apply-templates select="property/prep-arg"/>

    <xsl:text> )&#10;</xsl:text>
```

```
  </xsl:if>

</xsl:template>

<!-- identity transform --> <xsl:template match="node()|@*">
  <xsl:copy>
    <xsl:apply-templates select="node()|@*"/>
  </xsl:copy>
</xsl:template>

</xsl:stylesheet>
```

Bibliography

Abney, S. (1996). Partial parsing via finite-state cascades. *Natural Language Engineering*, 2(4):337–344.

ACE05 (2005). Ace (automatic content extraction) english annotation guidelines for entities version 5.6.1 (2005.05.23).

Antoniou, G. and Harmelen, F. (2004). *A Semantic Web Primer*. MIT Press.

Bacon, J. (1973). Do generic descriptions denote? *Mind*, (82):331–347.

Blackburn, P. and Bos, J. (2005). *Representation and Inference for Natural Language*. CSLI Publications.

Bos, J., Clark, S., Steedman, M., Curran, J., and Hockenmaier, J. (2004). Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, pages 1240–1246, Geneva, Switzerland.

Carlson, G. N. (1977). A unified analysis of the english bare plural. *Linguistics and Philosophy*, 1:413–457.

Carlson, G. N. (1982). Generic terms and generic sentences. *Journal of Philosophical Logic*, 11(2):145–181.

Carlson, G. N. and Pelletier, F. J., editors (1995). *The Generic Book*. University of Chicago Press.

Clark, S. and Curran, J. (2004). Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 104–111, Barcelona, Spain.

Cohen, A. (2002). Think generic! the meaning and use of generic sentences. *Language*, 78(1).

Cook, R. J. (1998). Kappa. In Armitage, T. P., editor, *The Encyclopedia of Biostatistics*, pages 2160–2166. Wiley, New York.

Croft, W. (2000). Countability in English nouns denoting physical entities: a radical construction grammar analysis. Unpublished manuscript.

Dean, M. and Schreiber, G. (2004). OWL web ontology language reference. W3C Recommendation.

Denoyer, L. and Gallinari, P. (2006). The Wikipedia XML Corpus. *SIGIR Forum*.

Finch, S. and Mikheev, A. (1997). A workbench for finding structure in texts. In Daelemans, W. and Osborne, M., editors, *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*. Washington D.C.

Fiona McNeill, Alan Bundy, M. S. (2003). Dynamic ontology refinement. In *Proceedings of 13th International Conference on Automated Planning and Scheduling*, Trento, Italy.

Grover, C. and Tobin, R. (2006). Rule-based chunking and reusability. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy.

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing*. Prentice Hall.

Lenat, D. (1990). Cyc: Towards programs with common sense. *Communications of the ACM*, 8(33):30–49.

Minnen, G., J. C. and Pearce, D. (2001). Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–203.

Mueller, E. T. (2006). *Commonsense Reasoning*. Morgan Kaufmann Publishers.

Passin, T. B. (2004). *Explorer's Guide to the Semantic Web*. Manning.

Russell, B. (1905). On denoting. *Mind*, (14):479–493.

Sang, E. F. T. K. and Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the Conference on Natural Language Learning (CoNLL-2000)*. Lisbon, Portugal.

Santorini, B. (1990). Part-of-speech tagging guidelines for the penn treebank project. 3rd Revision, 2nd printing.

Singh, P. (2002). The public acquisition of commonsense knowledge. In *Proceedings of AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access*, Palo Alto, CA.

Thompson, C., Califf, M., and Mooney, R. (1999). Active learning for natural language parsing and information extraction. In *Proceedings of the 16th International-Machine Learning Conference (ICML 1999)*, pages 406–414, Bled, Slovenia.

Thompson, H., Tobin, R., McKelvie, D., and Brew, C. (1997). LT XML. software API and toolkit for XML processing.

Tim Berners-Lee, J. H. and Lassila, O. (2001). The Semantic Web. *Scientific American*.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Daelemans, W. and Osborne, M., editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

W3C URI Interest Group (2001). URIs, URLs, and URNs: Clarifications and recommendations 1.0. http://www.w3.org/TR/uri-clarification.