Charles University in Prague
Faculty of Mathematics and Physics
&
University of Nancy 2
UFR Mathematics and Informatics

DIPLOMA THESIS

Domain Specific Information Extraction for Semantic Annotation

by

Zeeshan Ahmed

Supervisor 1 :	Dr.Yannick Toussaint
Supervisor 2 :	Dr. Martin Holub
Study program :	Computer Science
Study Specialization :	Computational Linguistics

European Masters Program in Language and Communication Technologies (LCT)

2009

Contents

Declaration

I hereby declare that this diploma thesis is my own work and where it draws on the work of others it is properly cited in the text. I understand that my thesis may be made available to the public.

Prague : 02/11/2009 **Zeeshan Ahmed**

Abstract

Semantic annotation is a helpful technique to understand the under laying semantics of the document. It provides additional information in the form of metadata which then makes documents to be processed in an intelligent way. The problem with semantic annotation is that these annotations are not universal e.g. semantic annotation for a document in particular domain might have different meaning in other domain. Therefore, domain specific knowledge is used for semantic annotation and this domain specific information is provided by ontologies.

The main problem with Semantic Annotation is availability of ontology for the domain. Ontology comprises of concept and relationships. In an ontology, a concept may be atomic or defined by a set of properties. This set of properties classifies the concept with other concept in ontology. In this thesis, we present an approach that deals with semantic annotation using properties of concept in an ontology rather than simple instance matching technique currently available. In this approach, the document is analyzed for the purpose of identifying these properties using ontology. If the properties found in document match with properties of any concept in ontology, the document is annotated with that concept. In this way, documents are indexed according to these properties.

The main target of this thesis is to present approaches of how these properties can be extracted from documents; both for the purpose of semantic annotation and ontology building. To achieve this target, we present two different approaches to information extraction for Semantic Annotation; "Rule Based Approach" and "Dependency Based Approach. We present the comparative analysis of effectiveness of these two approaches on a small corpus.

This kind of semantic annotation is useful for the efficient answering of search queries, clustering, text summarization etc. We apply this semantic annotation approach on the corpus of recipe documents. In our domain, these annotations are used for recipe adaptation purpose. In adaptation, the purpose is to intelligently replace some ingredient with other ones to make an adapted recipe. Apart from our main target of Information Extraction, we also propose an Ontology for our domain of recipe document as well as a process of semantic annotation.

Chapter 1

Introduction

1.1 Background

The growth of World Wide Web and corpus of huge documents has increased the necessity to bring up solutions that can intelligently manipulate documents. Documents constitute the valuable knowledge for particular domain. But due the unstructured nature of the documents, this knowledge cannot be efficiently exploited by machines for automation purpose. Thus the creation of semantic metadata related to document content seems to be a way to exploit this knowledge and extract implicit and explicit expertise.

Semantic Annotation (SA) is the approach proposed within the framework of semantic web for creating such metadata. SA refers to the process of indexing and retrieving useful knowledge from documents thus creating annotation or metadata on top of documents contents. These annotations or metadata are well defined for a particular domain using appropriate syntax and semantics. Therefore, the overall goal of SA is to create metadata that can be exploited by both humans and machines.

SA of documents hence is our pursued goals in this thesis. As proposed by (Berners-Lee T., Hendler J., and Lassila) [31], the semantic web can help us to reach this goal as it proposes the use of domain ontologies as semantic guide to annotate document content. Thus, semantic annotation is the process of mapping textual element found in the text with ontological concept. This mapping is performed on the basis of some criteria defined by concepts in ontology. The result of this mapping is well defined metadata that become the integral part of the document for its understanding and efficient processing.

In semantic web, domain ontology is a main resource for SA. Ontology is defined as formal and explicit specification of shared conceptualization (Gruber, 1993) [15]. It represents knowledge in structured form suitable for inference and reasoning over knowledge. However, the development of domain ontology is not a trivial task and consumes important resources in term of time and money. Thus, (semi) automatic generation of domain ontologies are used to reduce the cost of such an operation. Unfortunately, we also lack ontology for our domain. Therefore we also discuss how to automatically generate our domain ontology from recipe text. From this perspective the overall approach is depicted as follow

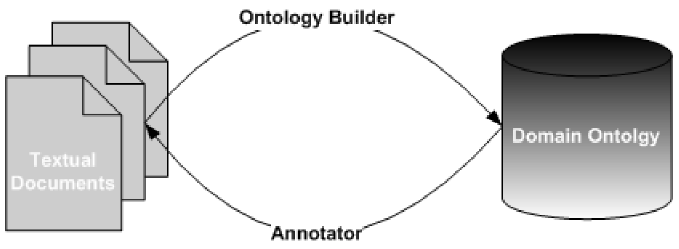


Figure 1.1: Text-to-Ontology and Anotation Cycle

1.2 Objective

This thesis is mainly concerned with SA as a guide to understand the document and how these annotations can be used for further processing. Similar to semantic web, we employ ontology as main resource to understand the textual information contained within the documents. Therefore the objective of this thesis is to define our new approach of SA, describe our domain ontology and define metadata structure created as a result of our SA approach. The big problem here is that our domain ontology is not available. We describe a method to create it from raw text. The Fig. 1.2 shows the overall architecture of the process.

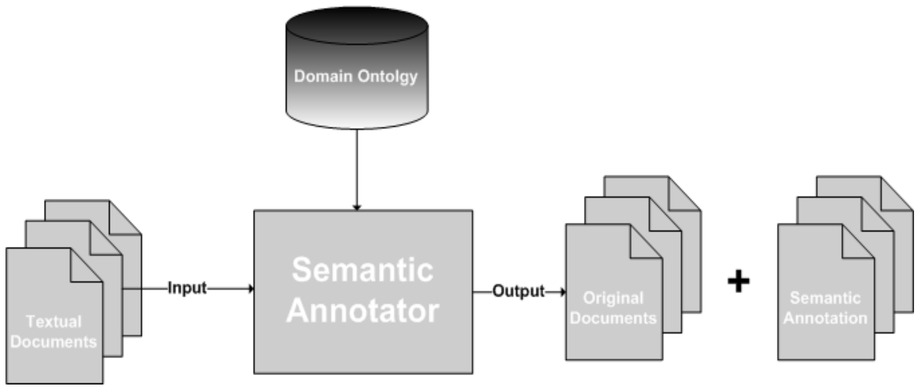


Figure 1.2: Semantic Annotation Process

There are different levels of granularity of SA i.e. annotation of complete document, paragraph, sentence, concept, term or word. But currently available systems like GATE [16], KIM [3], Melita [10], MnM [25], Magpie [11] etc. are only capable of annotating words or term. None of them provide annotation on granularity above word or term. These systems' technique is simply based on string matching between objects found in text and instances in ontology concept.

We define a different approach for SA of textual document in this thesis. In this approach our goal is to annotate documents according to concept it contains rather than just word or term. A concept can be atomic or define by the set of properties in an ontology. These properties of concept distinguish the concept from rest of the concepts in ontology. This makes the SA task more complex because this kind of annotation is not a simple annotation of word, term or relationship in document. Rather documents will have to be completely analyzed to find out properties.

We apply our SA approach on textual documents of cooking recipes. Our objective is to annotate the recipe according to the way ingredients are used. The different ways of preparation of ingredient are defined in an ontology. The concepts of ontology define different ways of preparation of an ingredient (or culinary action performed on ingredient) in recipe. We call these ontology concepts as ingredient preparation prototype. The task of our SA process is to extract out these actions for each ingredient from recipe text and annotate the recipe with appropriate prototype in ontology on the basis of these actions. For this purpose, we employ two different techniques for information extraction namely "Rule Based Approach" and "Dependency Based Approach". We also perform some experiment on corpus of manually annotated recipes using these techniques. We present the results that reflect the effectiveness of these approaches.

We use these annotations for recipe adaptation purpose. In recipe adaptation, we want to replace an ingredient in given recipe with another one. Our hypothesis is that simple ingredient replacement is not sufficient. Ingredient must be replaced in accordance with actions performed on it in the recipe. Let us consider following example. In this example, potato has the prototype "Potato (Peel, Cut, Boil, Drain, Mash, Beat)" because potato is being peeled, cut, boiled, mashed and beaten in this recipe.

> *"Peel potatoes and cut into large pieces. Boil in salted water for 15 to 20 minutes, or until tender. Drain potatoes. Mash potatoes in large bowl. Add milk in small amounts, beating after each addition, until desired consistency is reached. Add butter, 1/4 teaspoon salt, and pepper. Beating until mashed potatoes are light and fluffy."*

The problem here is the annotation of recipe according to above prototype. Currently available tool can only annotate all potato objects in the recipe as potato concept using some ontology. But none of them can associate actions to it like the prototype defined above. Moreover, none of current tool is able to find out if there is potato object in 2nd part of 1st sentence or in second sentence. These tools cannot provide us with ingredient prototype as "Potato (Peel, Cut, Boil, Drain, Mash, Beat)". In our approach it is possible to obtain such representation and we can use this prototype representation for adaption purpose. Let us suppose that we want to replace potato with "cabbage (Cut, Boil, mash)" prototype. Using our annotations, it is possible. Because these annotations guide us about the possible place in the recipe where to make changes. So, application of our approach results in following adapted recipe

> *" Cut cabbages. Boil in salted water for 15 to 20 minutes, or until tender. Mash cabbages in large bowl. Add milk in small amounts, beating after each addition, until desired consistency is reached. Add butter, 1/4 teaspoon salt, and pepper."*

According to the adaptation problem and corresponding example presented above, there are following four important tasks to semantic annotation process.

1. Robust Text analysis technique to extract out ingredient, actions and their mapping.
2. Creation of formal specification of conceptualizations i.e. ontology. In our case ontology describing different type of prototype of ingredient preparation.
3. Based on a defined ontology, an automatic process that tries to find the concept instances inside the target recipe documents and annotate it accordingly.
4. Formal representation of annotated documents.

In this thesis, we mainly concern with the information extraction component and show two different approaches to extracting information from recipe text. Apart from our main target of Information Extraction for semantic annotation, we also propose an ontology for our semantic annotation process. We call this ontology ingredient ontology that represents different ways of preparation of an ingredient or different actions performed on an ingredient in a recipe. We also propose our semantic annotation process and present examples to in support it.

1.3 Thesis Outline

This thesis has been organized as follows. Chapter 2 details the SA with formal definition. It also describes the state of the art techniques currently available. The chapter presents the SA model in which ontology is an integral resource. This chapter presents the detail description about ontology, its structure and representation. Chapter 3 describes methods of extracting actions, ingredients and their relationship from recipe text and issues faced during this process. This chapter is the core of this thesis in which we present our two approaches to information extraction namely "Rule Based" and "Dependency Based" approaches. Chapter 4 explains the automatic ontology construction from text. The current state of the art techniques available for this purpose are highlighted and the technique of Formal Concept Analysis (FCA) used for the development for ingredient ontology is explained. Chapter 5 then explains the annotation process and chapter 6 presents the conclusion.

Chapter 2

Semantic Annotation

Annotation is the process of adding additional information to any other information such as information in a book, document, online record, video etc. In linguistics, annotation is the process of adding additional linguistics information such as morphological, syntactic, semantic etc. to available linguistics forms to make these forms more descriptive.

SA is similarly defined as a process of adding semantic information to linguistic forms. But in the context of semantic web, Euzenat [12] formalized SA as mapping functions. From two set of objects, document and formal representation; two function can be defined. A function that maps documents to formal representation, called annotation and a function from formal representation to documents called indexing.

The formal representation is generally modeled in the form of comprehensive repositories like Ontologies. Ontology is defined as formal and explicit specification of shared conceptualization (Gruber, 1993) [15]. Ontology contains the description of type of objects and concepts, and their properties and relationship.

Therefore, SA of textual document is to identify the concept with the help of domain ontologies (*domain because semantic or concept analysis is mostly domain specific i.e. one thing defined for one domain may have difference sense or concept in another*). For this purpose, a robust text analysis technique is required which will be composed of identification of object in a text, identification of relationship between these objects and analysis on how these objects and their relationships combine to form a concept. Hence the objective of SA is to tag ontology class instances found in the text using text analysis process and map it into ontology classes as depicted in [22] by Fig 2.1.

From this perspective, the semantic annotation model is composed of following elements.

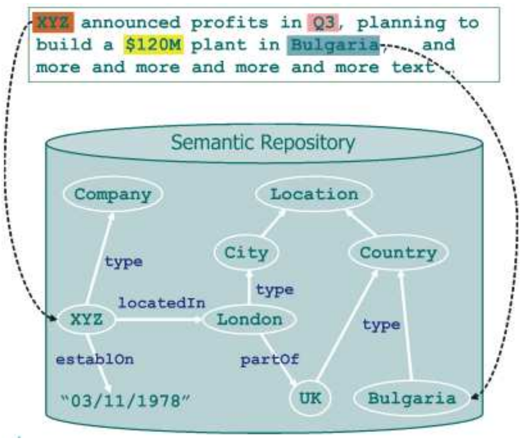

XYZ announced profits in Q3, planning to build a $120M plant in Bulgaria, and more and more and more and more text
Semantic Repository
Company
Location
City
Country
type
type
locatedIn
XYZ
London
type
establOn
partOf
"03/11/1978"
UK
Bulgaria

Figure 2.1: Semantic Annotation

1. An ontology describing the domain.
2. An annotation process or technique (mapping function as defined by Euzenat [12]) that links entities, objects or concepts in text with classes in ontology.
3. Representation or encoding of semantically annotated documents i.e. meta-data.

We define in following section current state of the art tools that are based on above model and then give a comprehensive discussion of the elements of this model in the following sections.

2.1 Tools for Semantic Annotation

There has been lot of work done in the field of semantic annotation using ontology as main guide but still there is no complete automatic semantic annotation tool available that has a good accuracy due to inherent ambiguity in Natural Languages. There are many tools available for semantic annotation of textual document like GATE [16], KIM [3], Melita [10], MnM [25], Magpie [11] etc. but none of the tools are totally automatic. Furthermore, these systems perform annotation on words and terminologies to indentify real world objects and their relationship in the text. None of them provide annotation above word level. Therefore these systems cannot be used in recipe annotation problem described in this thesis. In sequel, we present a brief overview of all of these tools.

GATE

GATE [11] (General Architecture for Text Engineering) is an infrastructure for development software components based on Human Languages. The GATE system provide many functionalities among them, it provides the functionality to annotate textual documents both manually and automatically. GATE uses JAPE [17] pattern matching engine for rule based Named Entity Recognition. JAPE is ontologically aware which can map the Named Entity to ontology classes during recognition. In GATE, the task of textual annotation is just defined more domain specific rules in addition to already available basic rules.

KIM

KIM [3] is another ontology base semantic annotation system that uses a special knowledge base(KIMO) which has been pre-populated with 200,000 entities. KIM uses GATE, SESAME and Lucene for many information extraction tasks. KIM also uses version of ANNIE for Named Entity Recognition. KIM has a feature of automatically adding new instances found in text to Ontology. It also performs disambiguation step because many instances can be added to different places in ontology.

Melita

Melita [10] provides the interface to semantically annotate the textual document using Adaptive Information Extraction technique. This technique reduces the burden of text annotation on user. It starts with manual annotation of text by user and as user keeps on annotating text the system learns the annotation process. Melita uses Amilcare [9] which runs in background learning how to reproduce the inserted annotation.

MnM

MnM [25] is another system based on supervise learning technique to annotate the text. it provides an environment to manually annotate a training corpus, and then feed the corpus into a wrapper induction system based on the Lazy-NLP (natural language processing) algorithm. The resulting output is a library of induced rules that can be used to extract information from corpus texts.

Magpie

Magpie [11] is browser enable ontology based semantic markup system that annotates the web document on fly. It uses ontology to annotate the document either using predefined lexicon in the ontology or using Named Entity recognition technique.

Some other system that are also used for semantic annotation are Onto-Mat [18] (work like MnM and Melita), AeroDML [23] (uses pattern based approach) etc.

2.2 Semantic Annotation Techniques

Currently available techniques for SA involve identification of objects and their relationships using ontology. Following is a brief review of these techniques.

2.2.1 Annotation of Object

The words or terms can be annotated using ontology in two different ways.

Lexicon based Annotation - In this approach, the each ontological class has set of lexicon associated with it. Annotation is simply a search through a lexicon list. If word or term is found, its appropriate class is assigned to extracted word or term in document.

Named-Entity Recognition - In Lexicon based approach, the problem is that it is difficult to build a lexicon for all the available vocabulary or terminologies. This is the case with open word classes of natural languages like noun, adjective, verbs etc. Nouns have big problem with Proper Nouns which are impossible to model through lexicon. Therefore, Named Entity recognition technique is applied to recognize the class of proper nouns such as, Name of Person, Name of Institute, and Name of Street etc. and then these classes are mapped to appropriate class of ontology.

2.2.2 Annotation of Relationship

Relationship can also be annotated using ontology. For example, Country and City are defined as having relationship of "partOf" in Ontology. Therefore, there is an appropriate need to annotate this relationship described by ontology. For example, in a text document, there is a fragment of text "Paris is capital of France". Using the Object annotation process, Paris and France can be annotated as City and Country and relationship annotation will define that Paris and France are related through "isCapitalOf" relation. This sort of annotation has been used in the Artequakt [2].

The relationship annotation can be performed using the shallow or deep syntactic analysis of sentences. The syntactic analysis provides us with structured representation of sentence where relationship between groups of words can be identified.

2.3 Representation Format for Semantic Annotation

To represent the metadata information created as result of SA process, there are various options available. The simplest of all is to directly modify document contents and add metadata in the same document. Another approach is to represent the metadata in separate document from the original one. In this way, document become independent of annotations. RDF (Resource Description Format) * is a standard format for this purpose. It has been specified by the World Wide Web Consortium (W3C) for representation of SA in the context of semantic web. RDF is XML based format. It encodes knowledge in sets of triples (also called statements), each triple representing the subject, predicate and object of an elementary sentence. For example to express the sentence "A knows B" in RDF, a triple with a subject denoting "A", a

*www.w3.org/RDF/

predicate denoting "knows" and an object denoting "B" can be formed. Apart from RDF, custom XML format can also be used.

2.4 Ontology

An ontology is a formal specification of a shared conceptualization (Gruber, 1993) [15]. A conceptualization can be understood as an abstract representation of the world or domain we want to model for a certain purpose.

A very formal definition of ontology is given by (Bozsak, 2002) [6]. According to him.

Definition 1. (Ontology) *An ontology is a structure $O := (C, \leq_C, R, \leq_R)$ consisting of (i) two disjoin set C and R called concept identifiers and relation identifiers respectively,(ii) a partial order $\leq_C$ on C called concept hierarchy or taxonomy, (iii) a function $\sigma : R \implies C \times C$ called signature and (iv) a partial order $\leq_R$on R called relation hierarchy.*

The Fig 2.2 dipicts an example ontology of foods.

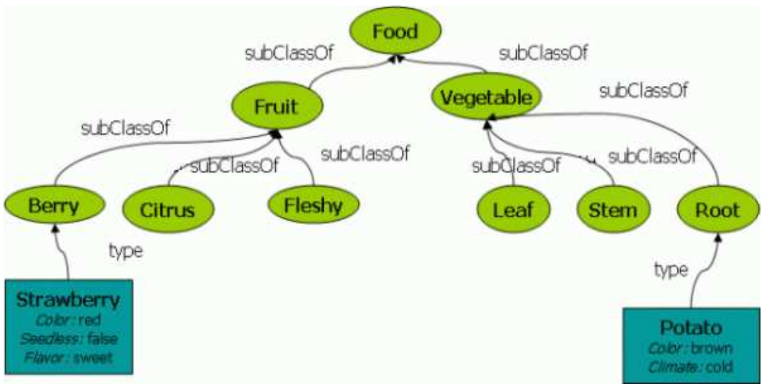


Figure 2.2: An Example Ontology of Foods

Ontology can be used for many different purposes, as described by [13]; to provide a controlled vocabulary, to customize and personalize search possibilities, to provide a structure that can be used for extracting document content, to perform word sense disambiguation or, as in our case, semantic annotation of textual documents.

To be exploited by any system, the ontology must be formally defined in term of its information structure and format of its representation. Let us briefly describe these two areas.

2.4.1 Structure and Component of Ontology

Ontologies describe individuals, classes, attributes, and relations. Individuals are the basic, "ground level" components of an ontology. The individuals in an ontology may include concrete objects such as people, animals, tables, automobiles, molecules, and planets, as well as abstract individuals such as numbers and words. Classes are the sets or collections of objects describe by the set of attributes. An object holds the membership of class if it possesses the same set of attributes as class. Classes may classify individuals with help of these attributes. Some examples of classes are Person, Vehicle, Car, Thing etc. Attributes are the aspects, properties, features, characteristics, or parameters that objects (and classes) can have. For example, a person class or object has the properties name, age, height etc which collectively differentiate it from other classes and objects. Relationships (also known as relations) between objects in an ontology specify how objects are related to other objects. Typically a relation is of a particular type (or class) that specifies in what sense the object is related to the other object in the ontology. For example in the ontology that contains the concept "Team" and the concept "Player" might be related by a relation of type "IsMemberof".

2.4.2 Ontology Representation

Since ontologies contains the knowledge according to the structure described above and in figure 2.2 , therefore this knowledge needed to represented in a format for efficient processing. Generally, the terminology used for ontology representation is ontology language. There are various type of ontology languages that can be used for representing ontology. (Maurizio Lenzerini, Diego Milano, and Antonella Poggi) [24] gives a comprehensive detail of the ontology languages like logic based, graph based, frame based etc.

The logic based representation uses language based on logic as formal representation for ontology. These kinds of languages have well defined semantics which can be efficiently used for inference on the ontology knowledge base. The well-known languages in this regard are prepositional logic, description logic and first order logic.

Description Logic (DL) [4] is one of the well-known knowledge representation frameworks. It offers means to structure knowledge in terms of *concepts, roles and individuals* [4]. DL languages allow expressions, or descriptions, to be composed out of other descriptions up to an arbitrary depth. A DL language is built on top of a collection of primitive concept and role names which denote the meaningful concepts and relations from a domain (e.g., Human, Male, Engineer, child, Mother, etc.), individual names (e.g., John) and constants ($\top$ and $\bot$).

Concepts in DL are interpreted as sets of individuals (their instances) and roles as sets of individual pairs. Further concepts and roles are defined by combining concept and role names, either primitive or already defined, via a set of constructors, e.g., conjunction ($\sqcap$), disjunction ($\sqcup$), negation ($\neg$). By definition, a role has a domain and a range concept and is inherited by the sub-concepts of the domain concept. It may be further restricted for every concept it applies to, for instance, by applying universal or existential quantifiers to the set of links. Thus, given a role r and a concept C, the following concept expressions can be composed: (i) $\forall r.C$ (value restriction), (ii)

$\exists$r.C (full existential quantification), and (iii) $\exists$r.$\top$ (limited existential quantification). These are basically the filter on the individuals: (i) collects those whose links of type r, if any, point exclusively to instances of the concept denoted by the expression C, (ii) those with at least one r link to such an instance, and (iii) those with at least one r link, regardless of the underlying concept. As an illustration, consider the expression of the concept of "Person all of whose children are either Doctors or have a child who is a Doctor" in DL:

$$\text{Person} \sqcap \forall\text{hasChild.}(\text{Doctor} \sqcup \exists\text{hasChild.Doctor})$$

The way DL represent information is widely suitable for the ontology representation. There are various standards defined that utilizes description logic as the underlying formalism. Web Ontology Language (OWL) * is well known standard defined by W3C consortium that is also based on DL. OWL has been widely accepted standard for representing and sharing knowledge in the Semantic Web context. OWL comes in three versions supporting different compromises between expressiveness and computational tractability: OWL-Lite only supports classification hierarchies and simple constraints, OWL-DL is more expressive but still computationally tractable, and OWL-Full is even more expressive but offers no computational guarantee. Particularly, OWL-Lite and OWL-DL belong to the description logics family, which are decidable fragments of first order logics.

DL has a precisely and formally defined semantics refering to the set theory. Some generic reasoning tools have been developed that leverage this semantics(Sattler, [30]). A DL reasoner performs various inferencing services, such as computing the inferred superclasses of a class, determining whether or not a class is consistent (a class is inconsistent if it cannot possibly have any instances), deciding whether or not one class is subsumed by another, etc. All this advantages of DL has made it widely accepted for ontology representation.

2.5 Conclusion

This chapter formally defined SA and major component of SA. It also presented an SA model which is composed of an ontology, an annotation process and representation format for the annotation. The chapter also highlighted current state of the art tools like GATE [16], KIM [3], Melita [10], MnM [25], Magpie [11] etc. that follow defined SA model. This chapter also addressed different techniques that are used by above tools for SA like annotation of objects and relationships. Different sort of representation format for represention of annotation were also described. At the end, ontology in general, its component & structure and DL based ontology language for representation of ontology was discussed.

*http://www.w3.org/2004/OWL/

Chapter 3

Information Extraction for Semantic Annotation

We apply our SA approach to cooking recipe documents corpus. One of the main components of SA is information extraction for analysis of text documents. The Fig. 3.1 present the overall schema of our semantic annotation process.

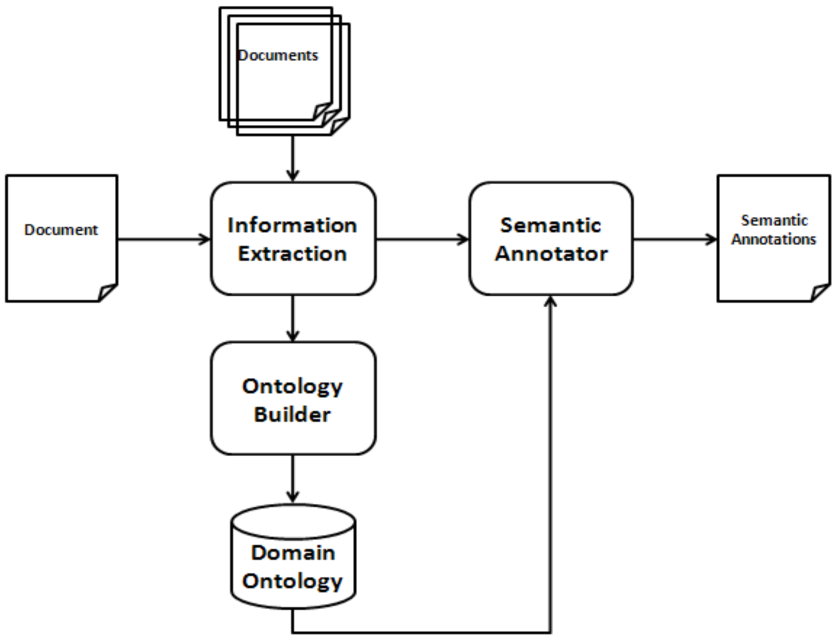


Figure 3.1: Semantic Annotation Process

As depicted in figure, there are three main components of our annotation process;

"Information Extraction"," Semantic Annotator" and "Domain Ontology". Since Domain Ontology is automatically created in our approach of SA, therefore, there is also an "Ontology Builder" process in this diagram. This process generates the ontology that is required by "Semantic Annotator" process. Note that all of our three processes i.e. "Information Extraction"," Semantic Annotator" and "Ontology Builder", are fully automatic.

In this chapter, we present Information Extraction component while Ontology Builder and Semantic Annotator components are presented in chapter 4 and chapter 5 respectively. we will present two different approaches to information extraction from cooking recipe documents namely "Rule Based" and "Dependency Based". The information extraction process is necessary not only for semantic annotation but also for ontology construction. The objective of our information extraction approaches is to extracting out ingredients, actions and their relationship information (i.e what actions are being performed on particular ingredient) from recipe. We also present at the end of the chapter some experimental results to highlight the effectiveness of the approaches and compare two approaches on the basis of results obtained.

3.1 Rule-Based Information Extraction

In rule based information extraction approach, we apply grammatical rule or pattern recognition technique to extract out ingredients, culinary actions and their relationships. For this purpose, on the basis of analysis of recipe text, few grammatical rules have been designed which capture the ingredient, action and their relation from the text. The Fig. 3.2 shows the schematic digram of the Rule Based Information Extraction process.

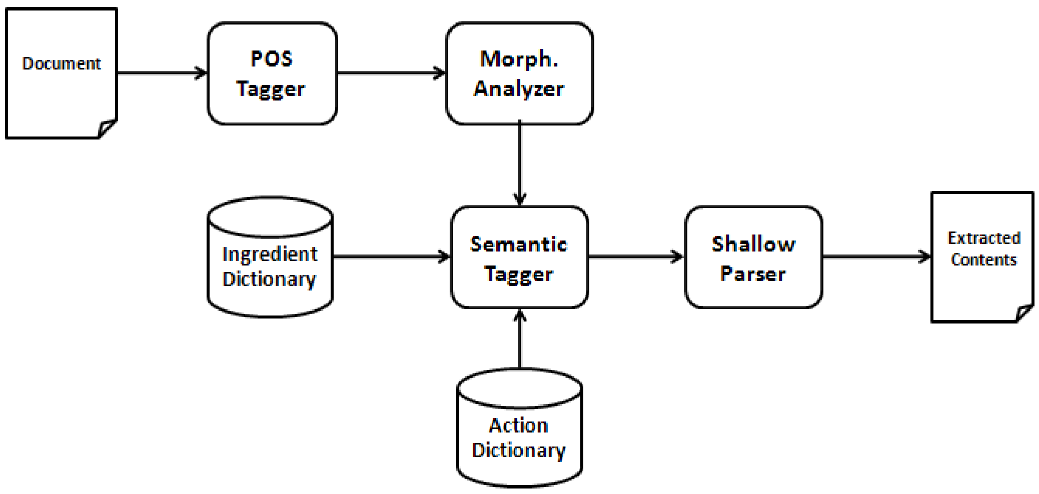


Figure 3.2: Rule Based Information Extraction

As depicted in figure, there are six major components of "Rule Based Information Extraction". There are four processes (POS tagger, Morph. Analyzer, Semantic Tagger and Shallow Parser) which are fully automatic while the two dictionaries are manually created from corpus. In sequel, we discuss following main steps that are applied in Rule Based information extraction from recipe text.

1. Domain Specific POS Tagging
2. Morphological Analysis
3. Identification of Ingredients and Actions
4. Identification of Ingredient-Action Relationships

3.1.1 Domain Specific POS-Tagging

In case of recipe text, culinary actions are mostly defined by the "verb" and sometime by "adjectives". The ingredients are identified as "nouns". Therefore, if these tags are primarily known then annotation process can be made quite robust because for searching for actions, only "verbs" and "adjectives" would have to be analyzed and similarly, for ingredients, "nouns" will be the focus of consideration. Furthermore, for extracting Ingreident-Action relationship, patterns of tags are used to search for the ingredients in the complement of action verb. These tasks make tagging essential for our work.

Natural language statements used in recipes are mostly imperative and instructive i.e. sentences start with "verbs". But most of the available POS tagger systems do not cater this problem. If used, these systems will tag "verbs" at the beginning of sentence as "nouns" which really degrade the information extraction process.

Therefore, corpus of recipes which contains around 82 different recipes has been tagged manually to custom train the Brill's tagger*. We use this tagger for POS tagging of recipe of documents.

3.1.2 Morphological Analysis

It is the process of analyzing internal structure of the word. As a result of morphological analysis, the word is broken into base form and affixes. Morphological analysis is necessary for our Information Extraction process because in this process we have maintained dictionaries for ingredients and actions. In these dictionaries, we only maintain base form of the words. Therefore, to search for any ingredient or action, first the corresponding word is analyzed for its base form and affixes, and then the base form is searched in dictionary. This process helps in reducing size of dictionaries because it prevents us to maintain all the different forms of the word.

*The Brown Corpus tag-set has been used here for tagging. http://en.wikipedia.org//wiki//Brown_Corpus.

3.1.3 Identification of Ingredients and Actions

For identification of ingredients and culinary actions, two types of dictionaries have been maintained. The ingredient dictionary contains ingredients and culinary action dictionary contains culinary actions. No other ingredients and culinary actions are considered apart from those present in dictionaries. These dictionaries have been manually created using the corpus that we have developed for the evaluation purpose for our approach. The corpus and dictionaries can be found in accompanied CD with this thesis.

Identification of Ingredient is trivial in the recipe. After POS tagging, all the nouns in the recipe text are searched in the list of ingredients in dictionary. Ingredients are appropriately marked if there is a match with dictionary.

Similar to identification of ingredients in the recipe, culinary actions are indentified using a list of culinary actions in dictionary. We consider only "verb" and "adjectives" for indentification of actions. After match is found, actions are appropriately marked. Table 3.1 list the semantic tags used for marking ingredient and actions in the recipe text.

Class	TAG
Ingredient	<ING>
Actions	<ACT> or <JJACT>

Table 3.1: Semantic TAGs

In these Tags, <ING> and <ACT> tags simply refer to ingredient and action while <JJACT> tag is a special tag. This tag is also used to indentify actions apart from <ACT> tag but it is only used for actions which are adjectives. If we find any adjective in a text with <JJ> pos tag and this adjective is listed in our dictionary, we mark it with <JJACT> tag.

3.1.4 Identification of Ingredient-Action Relationships

To extract the ingredient-action relationship information from recipe text, we have designed few grammatical rules that are given below. These grammatical rules indentify particular patterns in given piece of recipe text. If these patterns are successfully recognized in the text, the ingredients and actions in the recognized pattern are indentified to be related.

1 VP − > ACT NP (, NP)* (CC NP)?
2 VP − > ACT NP PP
3 VP − > ACT PP
4 VP − > JJACT ING | ING JJACT
5 VP − > ACT

6 NP − > DT? JJ* ING
7 PP − > IN NP (, NP)* (CC NP)?

The above rules are presented in Context Free Grammar (CFG) rules format. The following is brief description of the notations and symbols used in these rules.

Notation	Description
VP,NP,PP	Non-terminal symbols
ACT,JJACT, ING,CC,JJ,DT,IN,','	Terminal Symbols
()	Operator applies to every element in parentheses
*	Repeat one or more times
?	zero or one time
|	Logical "OR" Operator

Table 3.2: Rules Notation Description

Symbol	Description
ACT,JJACT,ING	Semantic Tags
CC	Coordinating conjunction
JJ	Adjectives
DT	Determiners
IN	Preposition

Table 3.3: Description of Non-Terminal Symbols

The rule 1 is used to extract relationship information when there is an action followed by one or list of ingredients e.g. "peel potatoes, onion and carrots." here "peel" is related with "potato","onion" and "carrot". As a result, we extract three relationships i.e. "Potato(peel)","Onion(peel)" and "Carrot(peel)".

The rule 2 is used to extract relationship information when there is an action that is being applied to two ingredients which are separated by preposition. This is the case with text like "Marinate chicken with yogurt". Currently, we have not been able to devise a way to say that either "Marinate" is being applied to chicken only or both chicken and yogurt. In this case, we extract two relationships from this text i.e. Chicken(marinate) and yogurt(marinate).

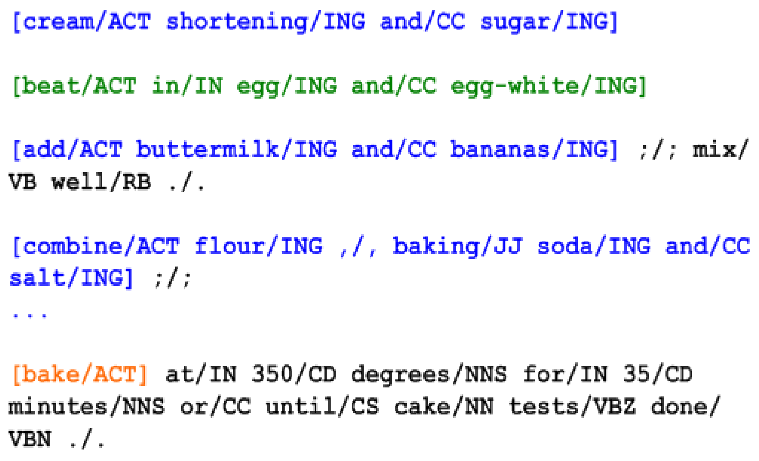
The rule 3 is applicable when there is an action that is being followed by a preposition and then a noun phrase which contain ingredient. "Beat in egg and egg white" is the piece of recipe text where this rule is applicable.

The rule 4 is a special kind of rule. In recipes, the text contains two type of information. One type of information is about ingredients and their quantities and other is the set of sequential steps. Sometimes when listing ingredient and their quantities the author of the recipe specifies the actions that must have been performed before applying further steps. Like " 1 kg of chopped potatoes", here protatoes must have been chopped before using in this recipe. Therefore, we extract "Potato(chop)" as relation from this text.

The rule 5 is also another kind of special rule. This rule is associated with discourse analysis. When none of the above rule is applicable and the piece of recipe text does not contains an ingredient then rule comes into play. The example of this rule is "Bake 350 F,30 mins.". In this text, there is no ingredient that has to be baked. Therefore its necessary to analyze this sentence with help of previous sentence to indentify which ingredient is related with bake. See discourse analysis section for further details.

The rule 6 & 7 are the sub-rules that are used in rule 1 to 5.

The Fig. 3.3 shows an example of how these patterns are matched in recipe text. The blue text shows that pattern 1 is matched with text, green color shows pattern 3 matches and brown color shows pattern 5 while text in black is not matched.



```
[cream/ACT shortening/ING and/CC sugar/ING]

[beat/ACT in/IN egg/ING and/CC egg-white/ING]

[add/ACT buttermilk/ING and/CC bananas/ING] ;/; mix/
VB well/RB ./.

[combine/ACT flour/ING ,/, baking/JJ soda/ING and/CC
salt/ING] ;/;
...

[bake/ACT] at/IN 350/CD degrees/NNS for/IN 35/CD
minutes/NNS or/CC until/CS cake/NN tests/VBZ done/
VBN ./.
```

Figure 3.3: Shallow Parsing of Recipe

The sample output of this example would be as "sugar(cream)" , "softening(cream)", "egg(beat)", "buttermilk(add)", "banana(add)", "flour(combine)", "soda(combine)", "salt(combine)" and at the end everything formed so far is baked i.e. "bake(allingredients)". The last information is extracted using discourse analysis as discussed in section 3.2.

3.1.5 Implementation

We have developed our suggested information extraction approach for extracting ingredients, actions and their relationships using a shallow parser. We have developed this parser in java language. All the program source code and relevant data for this parser is available in accompanied CD with this thesis.

The parser takes the sentences one by one. It first indentifies if there is any ingredient and action in the sentence. For this purpose, the sentence is first tokenized and tag with POS tag using our custom built POS tagger. For tokenization, we simply take into consideration white spaces and punctuations. The tokens in the tagged sentence are then searched for ingredients and actions using dictionaries. The identified ingredients and actions are appropriately marked with <ING>, <ACT> and <JJACT> semantic tags.

After tagging, the parser then applies the shallow parsing rules 1 to 5(presented in previous section) in priority order. If any of the rule is applicable to the chunk (a technical term used to refer piece of text to which shallow parsing rule is applied successfully) , the ingredient-action relationships are created for the ingredients and actions presented in the chunk. Our parser does not create the overlapping chunks means if one rule is applied on one chunk then the second chunk will start after the first one ends but not in between the first one.

3.2 Dependency Based Information Extraction

In this approach of information extraction, we are going to use well know syntax analysis technique called Dependency based parsing. In this technique the syntactic analysis of text is based on dependencies between words within a sentence. The syntactic structure of sentence is determined by the relation between a word (a head) and its dependents.

Dependency relations among the words of a sentence can be represented as a graph. More specifically, the dependency structure for a sentence $w = w_1 ... w_n$ is the directed graph on the set of positions of w that contains an edge $i- > j$ if and only if the word w_j depends on the word w_i. In this way, just like strings and parse trees, dependency structures can capture information about certain aspects of the linguistic structure of a sentence. As an example, consider Fig. 3.4. In this graph, the edge between the word "likes" and the word "John" encodes the syntactic information that "John" is the subject of "likes". When visualizing dependency structures, we represent (occurrences of) words by node, and dependencies among them by arrows: the source of an arrow marks the governor of the corresponding dependency, the target marks the dependent.

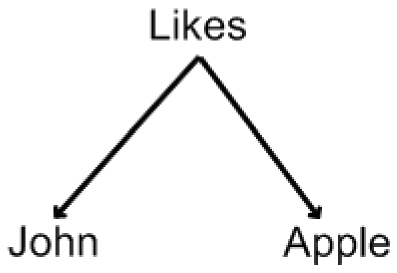


Figure 3.4: Dependency Tree

3.2.1 Dependency Based Parsing

There are various tools available for dependency base syntax analysis. For our experiment in this thesis, we use Stanford Lexicalized Dependency Parser[29]. The Stanford parser can output typed dependency between pair of words. Here typed dependency means each dependency is accordingly marked either as subject, object, modifier etc.

The Stanford typed dependencies representation provides a simple description of the grammatical relationships in a sentence that can easily be understood and effectively used by people without linguistic expertise who want to extract textual relations. It represents all sentence relationships uniformly as typed dependency relations between pairs of words, such as "the subject of distributes is Bell." This sort of representation is quite accessible to non-linguists thinking about tasks involving information extraction from text and is quite effective in relation extraction applications. Following is an example English sentence and it dependency relationship. This example has been taken from Stanford typed dependency manual[29] provided with the Stanford Parser.

> "Bell, based in Los Angeles, makes and distributes electronic, computer and building products."

The Stanford Dependencies (SD) representation is:

```
nsubj(makes-8, Bell-1)
nsubj(distributes-10, Bell-1)
partmod(Bell-1, based-3)
nn(Angeles-6, Los-5)
prep in(based-3, Angeles-6)
conj and(makes-8, distributes-10)
amod(products-16, electronic-11)
conj and(electronic-11, computer-13)
amod(products-16, computer-13)
conj and(electronic-11, building-15)
amod(products-16, building-15)
dobj(makes-8, products-16)
dobj(distributes-10, products-16)
```

This maps straightforwardly onto a directed graph representation, in which words in the sentence are nodes in the graph and grammatical relations are edge labels. Fig. 3.5 gives the graphical representation for the example sentence above.

The details for all the typed dependencies can be found in "Stanford Typed Dependency Manual"[29].

The Stanford parser provides different styles of dependency representation. Currently, there are four variant of type dependency representation. The representation format is same for all i.e abbreviated_relation_name(governor, dependent). The difference are that they range from a more surface oriented representation, where each token appear as the dependent in a tree, to a more semantically interpreted representation.

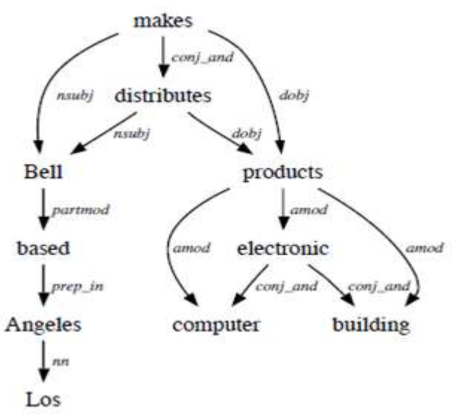


Figure 3.5: Dependency Tree

Basic

In the basic type dependency representation, each word (except for the head word) is related to(or dependent on) other word and. This typed dependency representation form a tree structure. For example for the sentence, "Bell, a company which is based in LA, makes and distributes computer products", the typed dependencies will be follow whereas the corresponding tree structure is shown in Fig. 3.6.

nsubj(makes-11, Bell-1)
det(company-4, a-3)
appos(Bell-1, company-4)
rel(based-7, which-5)
auxpass(based-7, is-6)
rcmod(company-4, based-7)
prep(based-7, in-8)
pobj(in-8, LA-9)
cc(makes-11, and-12)
conj(makes-11, distributes-13)
nn(products-15, computer-14)
dobj(makes-11, products-15)

Collapsed Dependencies

In the collapsed representation, additional dependencies are considered, even ones that break the tree structure (turning the dependency structure into a directed graph). So in the above example, the following relations will be added:

ref(company-4, which-5)
nsubjpass(based-7, which-5)

These relations do not appear in the basic representation since they create a cycle with the rcmod and rel relations. Relations that break the tree structure are the ones taking into account elements from relative clauses and their antecedents (as shown in this example), as well as the controlling (xsubj) relations.

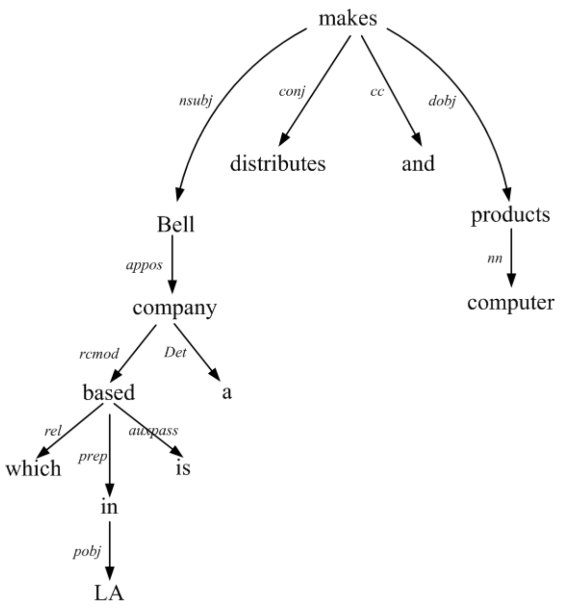


Figure 3.6: Dependency Tree for Basic Representation

Furthermore, dependencies involving prepositions, conjunct and information about the referent of the relative clauses are collapsed to get the direct dependencies between content words. This is very useful in relation extraction application. In the above example, the preposition "in" will collapse following two relations into one relation.

prep(based-7, in-8)
pobj(in-8, LA-9) $\Longrightarrow$ prep in(based-7, LA-9)

Similary for conjunction

cc(makes-11, and-12)
conj(makes-11, distributes-13) $\Longrightarrow$ conj_and(makes-11, distributes-13)

At the end, the following typed dependencies are obtain from the standford parser for our example sentence. The corresponding tree structure is shown in Fig. 3.7.

nsubj(makes-11, Bell-1)
det(company-4, a-3)
appos(Bell-1, company-4)
nsubjpass(based-7, company-4)
rel(based-7, which-5)
auxpass(based-7, is-6)
rcmod(company-4, based-7)
prep_in(based-7, LA-9)
conj_and(makes-11, distributes-13)
nn(products-15, computer-14)
dobj(makes-11, products-15)

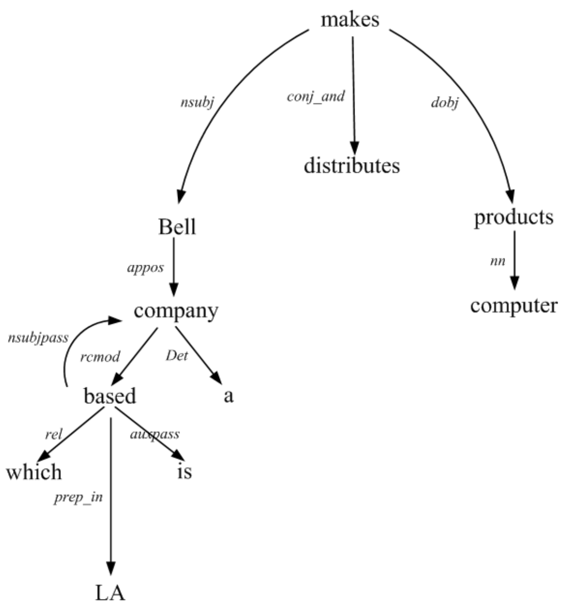


Figure 3.7: Dependency Tree for Collapsed Dependency Representation

Collapsed Dependencies with Propagation of Conjunct Dependencies

When there is a conjunction, we can also get propagation of the dependencies involving the conjuncts. In the sentence here, this propagation will add two dependencies to the collapsed representation; due to the conjunction between the verbs "makes" and "distributes", the subject and object relations that exist on the first conjunct ("makes") will be propagated to the second conjunct ("distributes"). The tree structure shown in Fig. 3.8 highlights this change.

nsubj(distributes-13, Bell-1)
dobj(distributes-13, products-15)

Since this representation is an extension of the collapsed dependencies, it does not guarantee a tree structure.

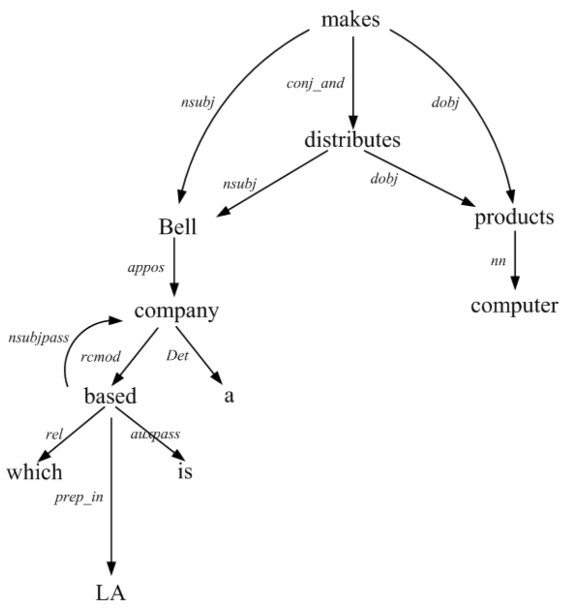


Figure 3.8: Dependency Tree for Conjunct Propagation Representation

Collapsed Dependencies Preserving a Tree Structure

In this representation, dependencies which do not preserve the tree structure are omitted. As explained above, this concerns relations between elements of a relative clause and its antecedent, as well as the controlling subject relation (xsubj). This also does not allow propagation of conjunct dependencies. In our example, the dependencies in this representation will be following and corresponding tree structure is shown in Fig. 3.9.

nsubj(makes-11, Bell-1)	rcmod(company-4, based-7)
det(company-4, a-3)	prep_in(based-7, LA-9)
appos(Bell-1, company-4)	conj_and(makes-11, distributes-13)
rel(based-7, which-5)	nn(products-15, computer-14)
auxpass(based-7, is-6)	dobj(makes-11, products-15)

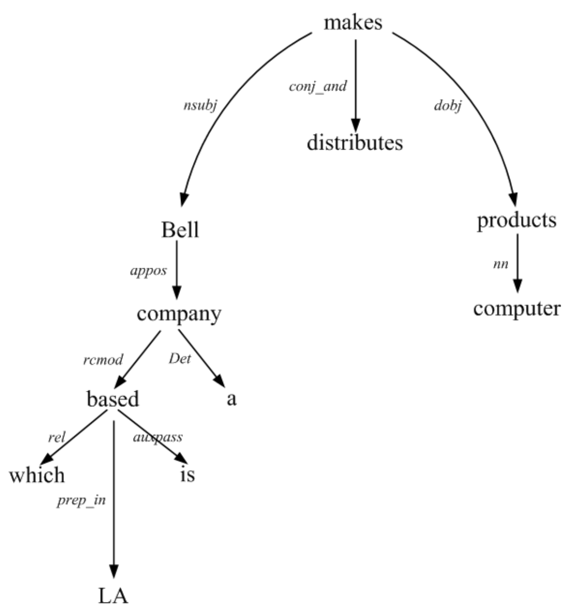


Figure 3.9: Collapsed Dependency Preserving a Tree Structure Representation

3.2.2 Ingredient, Action and their Relationship Extraction

We experiment Stanford typed dependency parser for extracting ingredient, action and their relationship. The following Fig. 3.10 depicts the overall schematic approach of Dependency Based Information Extraction process.

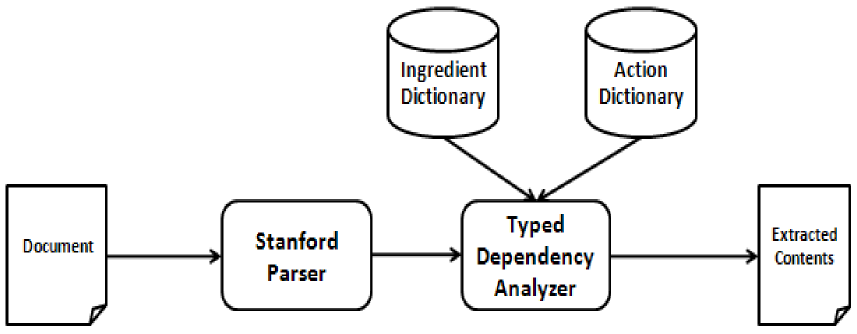


Figure 3.10: Dependency Based Information Extraction

As depicted in figure, there are four major components of "Dependency Based Information Extraction" process. There are two processes (Stanford Parser and Typed Dependency Analyzer) which are fully automatic while the two dictionaries (Ingredient and Action) are manually created from a corpus. As in the case of our Rule Based information extraction approach, the ingredients and actions are indentified using dictionaries. But one of the main advantages of using dependency parsing approach is that relationship extracting become trivial. There is no need to form grammatical rule to extract relation rather relationship information is provided as a part of output from parser. Similarly as in the case of Rule Base extraction, morphological analysis is necessary for searching of word in the dictionaries.

In our experiment, we work with "Collapsed dependencies with propagation of conjunct dependencies" representation format. This format is very useful in extraction of relationship information between actions and ingredients even if they are separated by many words. Collapsed dependencies representation format is also handy in dealing with preposition as we don't have to traverse through preposition dependency to look for dependent ingredient of action. For example, in the sentence "Beat in egg". The basic dependency format provide following representation.

```
prep(Beat-1, in-2)
pobj(in-2,egg-3)
```

In this output, if it is to find what ingredient is related to "beat" action then two relations "prep" and "pobj" would have to be analyzed. While collapsed dependency provides

```
prep_in(Beat-1, egg-3)
```

which is very easy to capture relationship information.

After parsing the sentence, we look for the relation in which governor is action and dependent is ingredient or governor is ingredient and dependent is action. When such relationships are indentified we call them Ingredient-Action relationship in our domain. For example consider following sentence.

"Scrub potatoes and pat dry."

After parsing the sentence with Stanford parser, we get following dependencies.

```
nn(potatoes-2, Scrub-1)
dep(dry-5, potatoes-2)
conj_and(potatoes-2, pat-4)
dep(dry-5, pat-4)
```

Suppose "potato" is present in our ingredient dictionary and "scrub" and "dry" are present in action dictionary then following Ingredient-Action relationship are extracted.

```
potato(scrub,dry)
```

"scrub" and "potato" are related because there is a dependency relation between these two as shown in example above. In this relation "potato" is governor and "scrub" is dependent. Similarly, "dry" and "potato" are also related because there is a dependency relation between them. We don't take into consideration dependency type through which words are related. Furthermore, we don't also take into consideration the validity of relation i.e. normally in dependency parsing verbs are governor but here in our example above "scrub" is a verb but it's a dependent of potato which is noun.

3.3 Discourse Analysis for Information Extraction

Discourse referents are the linguistic terms that stand in place of other linguistics terms in the text. One of the famous examples of discourse referent is "Anaphora". Anaphora is the "pronouns" that stand in place of some other "nouns" mostly proper nouns. In the recipe text, we come across with anaphora various times. So there is a need to resolve anaphora before processing.

Another sort of discourse referent present in recipe text is the "Ellipsis". "Ellipsis" refers to the omission from a clause of one or more words that would otherwise be required by the remaining elements. The omitted information can be derived from the context.

There are various theories and techniques to handle discourse referent in text such as Discourse Representation Theory (DRT) [21]. But we will apply simple approach to discourse handling in this thesis. Let us consider the following two sentences that are the excerpt from the recipe.

"sprinkle over potato mixture. Bake."

In second sentence, it has not been specified explicitly what is to be baked but from the discourse or information available from the first sentence, it can be derived that it is the potato mixture that needs to be baked. To cater this problem, the semantics of each step has to be analyzed. Since it can be observed from recipe text that there are some inputs and outputs of each step, the input can be any ingredient defined in the recipe or mixture that has been formed till now and output is the action performed on them. Input can be missing or present; if it is present then it is referred by its name but if missing then it is assumed that last step output has been referred here.

For this purpose, a list is maintained for the output of each step and this list is presented to text analyzer for the next step analysis. Here in above example, the "Bake" action does not define any direct object to be baked; therefore it is assumed that the output of last step is bake and that is potato mixture. Similarly, anaphoras are resolved with output of last step.

3.4 Evaluation

We apply our Rule Based and Dependency Based information extraction approaches on recipe corpus of 43 recipes which were randomly selected from internet. We have developed this corpus for the purpose of evaluating our approaches. The corpus only contains evaluation data. This data is totally independent form the data that was used for the purpose of analysis of rules for our Rule Based approach. Whereas, the dependency parser that we have used for Dependency Analysis is already trained. So, our evaluation corpus is completely unseen for both approaches. This corpus contains 50 ingredients and 43 actions. Each recipe in the corpus has corresponding annotations that describe what information should be extracted from that recipe. The recipes have been manually annotated for the purpose of automatically evaluating our information extraction approaches. To evaluate our both information extraction approaches, we are interested in calculating the accuracy using the standard measures of "Precision" and "Recall".

We have implemented both of our Information Extraction approaches in JAVA language. A parser has been developed for our rule based approach that utilizes the rules presented in rule base information extraction section. The detailed information about this parser has been presented in section 3.1.5. The parser is accompanied by custom trained brill's part of speech tagger. This tagger works in python using NLTK tool kit and has been custom trained on manually annotated(at Part-of-Speech level) corpus of recipes. Note that this corpus is different from our evaluation corpus described above. The tagger and corpus were created by students of Orpailleur Group at Loria, University of Nancy 2, France. Furthermore, for morphological analysis, Stanford morphological analysis engine has been used that is provided as a part of Stanford POS tagger [32]. For dependency base parsing, we have used Stanford Dependency Parser [29]. All the program source code and relevant data is provided in accompanying CD with this thesis.

3.4.1 Precision & Recall

In Information Extraction domain, Precision describes the quantity of relevant information extracted in total amount of information extracted. It is represented in term of mathematical formula as

Precision = Relevant Information / Total Information.

On the other hand, Recall is the measure of how much relevant information extracted in overall relevant information. i.e.

Recall = Relevant Information extracted / Total Relevant Information

Let us consider an example of how precision and recall are measured in our experiment. We consider the following recipe from our experiment. The Table. 3.4 shows the contents of the recipe and the information that should be extracted from this recipe.

Recipe	Manually Extracted Information

6 To 8 potatoes
2 tb To 4 butter of lard
Salt & pepper
1 Onion chopped
Peel and slice potatoes.
Heat fat in a large skillet until hot, add potatoes.
Fry well, turning occasionally, until nice and brown.
Add onion and fry a few minutes more.
Season with salt and pepper.
	Onion(chop,fry)
	Potato(peel,slice,fry,brown)

Table 3.4: Recipe and Relevant Extracted Information

In manually extracted information, there are 6 ingredient-action relations. We consider *onion*(*chop*) as single relation. There are two ingredients *onion* and *potato* that are of interest. *Onion* has two actions related to it and *potato* has four actions related to it.

When we applied our Rule Based and Dependency Based Information Extraction approaches the results shown in Table. 3.5 were obtained.

Rule Based Information Extraction	Rule Based Information Extraction

Onion(chop)
Potato(peel,slice)
	Onion(chop,fry)
	Potato(peel,slice)

Table 3.5: Rule Based and Dependency Based Extracted Information

We now calculate the precision & recall measure for the two approaches.

Rule Based Approach

The following are the statistic obtained from the results.

Total relation(s) from Manually Extracted Information	= 6
Total relation(s) from Rule Based Information Extraction	= 3
Correct relations from Rule Based Information Extraction	= 3

Precision = (No. of Correct rel. *100/ Total rel.) %
= 3*100/3 = 100%

Recall = (Total correct rel. *100 / Total rel. in text) %
= 3*100/6 = 50%

The precision for Rule based approach is 100% because of the fact that whatever relations have been extracted are correct whereas recall is 50% because this approach could not extract 50% of the relations that should have been extracted.

Dependency Based Approach

The following are the statistic obtained from the results.

Total relation(s) from Manually Extracted Information	= 6
Total relation(s) from Dependency Based Information Extraction	= 4
Correct relations from Dependency Based Information Extraction	= 4

Precision = (No. of Correct rel. *100/ Total rel.) %
= 4*100/4 = 100%

Recall = (Total correct rel. *100 / Total rel. in text) %
= 4*100/6 = 66.66%

The precision for Dependency based approach is 100% because of the fact that whatever relations have been extracted are correct whereas recall is 66.66% because this approach could only extract 66.66% of the relations from the text.

3.4.2 Rule Based Approach

We applied our rule base information extraction approach on the data using all the settings explained previously. The following table presents the "Precision" and "Recall" measure of our rule base approach on manually annotated recipe corpus.

Method	Precision	Recall
Rule Based	97.39414%	51.54639%

Table 3.6: Rule Based Information Extraction Results

The results show that precision measure for the rule base approach is very satisfactory but the recall measure is quite disappointing. This shows that our Rule Based approach is able to extract information which is mostly relevant i.e. it is not extracting noise from the corpus. But Rule Based approach is not able to collect all the information from recipe corpus. That's why recall measure is low.

Error in Precision

The precision measure for the Rule Based approach is around 97%. But our intension was to develop rules that give us 100% accuracy. To find out the reason for error in precision, we looked more closely into our data. It was found out that the main reason of error in precision for Rule Based approach is due to the discourse analysis.

Since we are not performing complete analysis of sentence, sometimes our parser does not analyze the ingredient-action relations effectively.

For example, consider the following two consecutive sentences in a recipe.

"Wash and dice chicken. Boil broth in a big pan."

The information extracted by our Rule Based approach is "Chicken(wash,dice,boil)". The "Boil" action should not have been extracted as it is not associated with chicken. Keep in mind that we don't take into consideration "broth" as ingredient and it is also not present in our ingredient dictionary.

When we deeply analyzed our parser, it turned out that during the analysis of second sentence by the parser, none of our top 4 rules presented in section 3.1.4 could be successful. The rule 5 is then only choice and it is applied to the second sentence successfully and identifies "Boil" as an action. Since none of our rules could detect that there is an object associated to "Boil" action, therefore our parser analyzes it as action alone. In this situation, discourse analysis comes into play and associates output of last sentence as possible input to the second sentence. Therefore, our parser detects as "Boil" is being applied to "Chicken". This situation can be avoided if we are able to analyze sentence completely.

Error in Recall

The reason for low recall measure is limited set of rules and very simple handling of discourse. As a human being, we are intelligent enough to capture discourse of running text very efficiently which is reflected by our annotated corpus. But our discourse analysis technique described in previous section is not efficient enough to handle discourse like human beings. In recipe, discourse is span over complete text of recipe while our discourse analysis is only span over two sentences.

Role of Discourse Analysis

For the purpose of analyzing the behavior of our discourse analysis technique in recipe text, we perform another experiment in which we don't make use of our discourse analysis technique. The results of experiment are as follow.

Method	**Precision**	**Recall**
Rule Based	98.75776%	32.37113%

Table 3.7: Rule Based IE Results without Discourse Analysis

As shown by the results, our precision measure is getting little bit higher while recall measure notably decreased. It shows that even though we are applying simple technique for discourse analysis, it gives 19% improvement in recall.

Comments

Hence we conclude that rules developed for information extraction from recipe text are good enough to give better results provided that all the possible rules are available. Furthermore, these rules should be accompanied by good discourse analysis technique.

3.4.3 Dependency Based Approach

We applied our Dependency Based approach on the same data using same setting as in the case of Rule Base approach. The following results were obtained which highlight the "Precision" and "Recall" measure.

Method	Precision	Recall
Dependency Based	95.39877%	64.12371%

Table 3.8: Dependency Based Information Extraction Results

Similar to rule based approach, the precision measure is reflecting satisfactory accuracy while the recall measure is better than Rule Based Approach but still not appreciating. It shows that dependency base approach is also able to extract correct information but is not capable of extracting all the correct information present in the corpus.

Error in Precision

There are two main reasons of error in precision for Dependency Based approach. One is Discourse Analysis technique and other is over-parsing by Stanford Dependency parser. The discourse problem is same as discussed in "Rule Based approach" while over-parsing problem is discussed in section 3.4.4.

Error in Recall

The reason for this low recall measure is the handling of discourse in a simple manner.

Role of Discourse Analysis

To analyze the effect of discourse in Dependency Base approach, we perform another experiment in which we eliminate our discourse analysis technique. As a result, following precision and recall measure is obtained.

Method	Precision	Recall
Dependency Based	95.71428%	41.44330%

Table 3.9: Dependency Based Information Extraction Results without Discourse Analysis

As shown by the results, this time precision measure is more or less the same while recall measure notably decreased. The recall has droped from 64% to 41%. This shows that our discourse analysis technique is somehow reasonable.

Comments

Hence we conclude that Dependency Based approach is also good enough to give better results if error related to over-parsing could be eliminated. If it is accompanied by good discourse analysis technique, the recall can also be further improved.

3.4.4 Comparison of Results

The following table presents the comparative analysis of the Rule based and Dependency Based approaches.

Method	**Precision**	**Recall**
Rule Based	97.39414%	51.54639%
Dependency Based	95.39877%	64.12371%

Table 3.10: Comparison of Dependency Based and Rule Based Information Extraction Results

Comparision of Precision

It is interesting to note that precision measure for Rule Based approach is higher than Dependency Base approach. It shows that rules developed for Rule Based approach are only extracting information for which these are meant for. While Dependency Based approach is extracting little erroneous information.

After the close analysis of our recipe corpus, it is found that the main error in precision in Dependency Based approach is the presence of sentence like

> *"Combine beaten eggs with vinegar, sugar, and pepper; mix to blend well."*

Which gives following dependency representation.

nsubj(beaten-2, Combine-1)
dobj(beaten-2, eggs-3)
nn(pepper-10, vinegar-5)
prep_with(beaten-2, sugar-7)
conj_and(pepper-10, sugar-7)
prep_with(beaten-2, pepper-10)
dep(pepper-10, mix-12)
aux(blend-14, to-13)
infmod(mix-12, blend-14)
advmod(blend-14, well-15)

Here the dependencies in red color should not be extracted. But due to preposition propagation applied during parsing, the result is in some incorrect dependencies.

Comparision of Recall

In case of recall, Dependency Based approach out-performs Rule Based approach. Note that the discourse analysis (the major technique that is responsible for increase in recall) is same for both approaches. But this accuracy in recall is due to the fact that Dependency Based parsing is capable of handling natural dependencies effectively especially long distance dependencies which Rule Based approach is not able to cater.

After analyzing the recipe text present in our corpus on which the experiments were performed, it is found that Dependency Based parser is capable of extracting ingredient-action information in the following type of sentences as show in table 3.11 while there is no rule to handle such sentences in Rule Based approach. Moreover, inherent error in POS tagging is also contributing in low recall in Rule Based approach.

Sentence	Dependency Parse	Ingredient-Action

Add onions and fry until golden.	dobj(Add-1, onions-2)
	dobj(Add-1, fry-4)
	conj_and(onions-2, fry-4)
	prep_until(Add-1, golden-6)
		Onion(Fry)

Peel potatoes and cut into large pieces.	dobj(Peel-1, potatoes-2)
	dobj(cut-4, potatoes-2)
	conj_and(Peel-1, cut-4)
	amod(pieces-7, large-6)
	prep_into(cut-4, pieces-7)
		Potato(Peel,Cut)

Table 3.11: Sentences where Dependency Bassed approach works while Rule Based approach does not.

Comments

According to experiments performed, we would suggest that "Dependency Based Parsing" approach for information extraction could be better for scenarios discuss in this text. It is giving good recall due to the better analysis of dependency structure in the text. Furthermore, recall can be further increased if more effective discourse analysis technique could be used.

3.5 Conclusion

This chapter addressed the textual domain of work of this thesis i.e. textual document of cooking recipes. The chapter discussed one of the main tasks of our annotation process i.e. Information Extraction (IE) from recipe document. Two main techniques have been discussed namely "Rule Based Approach" and "Dependency Based Approach".

In Rule Based Approach, many issues pertaining to IE from recipe document were highlighted such as POS-Tagging, Morphological Analysis, how ingredients, actions and their relationship information is extracted. The chapter then presented the techniques to handle all of these issues. POS- tagging problem was solved by custom training a tagger on manually annotated recipe corpus. Dictionaries were used to indentify ingredient and actions in the text. Grammatical rule were developed to extract ingredient-action relationships.

In Dependency Based Approach, Stanford Dependency Parser was used for dependency parsing of recipe text. The dependency parsing indentifies the potential relationship between words in the text. These relationships are further analyzed for ingredient-action relationship. At the end of chapter, experiments and results were presented to describe the effectiveness of two techniques.

Chapter 4

Automatic Ontology Construction from Text

The semantic annotation model that we are following in this thesis requires ontology as main resource. But the big problem for the accomplishment of our SA objective is the unavailability of ontology that represent prototype for ingredient. In this chapter, we will propose an ingredient ontology for our SA process and method to built it automatically from recipe corpus.

4.1 Tools and Techniques for Ontology construction

The automatic ontology construction is not a trivial task and still require lots of human intervention between some stages of ontology construction. There are various approaches and tools available for automatic construction of ontology from natural language text. In [7], the authors present an interesting overview of the ontology generation layers. According to them, it consists of six extraction layers of growing complexity: terms, synonyms, concepts, taxonomy, relations and rules. A number of systems have been proposed for ontology learning from text. These systems combine one or more of the six layers mentioned above. Examples of systems are InfoSleuth [20], Text-To-Onto [26], Ontolearn [27], OntoLT [8] and GlossOnt [28]. Most of these systems exploit linguistic analysis and machine learning algorithms to find interesting concepts and relationships.

Apart from these system, Formal Concept Analysis (FCA) [5] approach has also been used for ontology generation. FCA based approach is based on application of Natural Language Processing (NLP) and Formal Concept Analysis (FCA) [14]. In this approach the NLP techniques are applied for extraction of factual information from textual documents and represented in structured form. Then FCA is applied that results in graphical representation of concepts in the form of concept lattice. In this thesis, the FCA approach is applied to build ingredient ontology automatically. This approach is useful for automatic detection of conceptual information and their relations. Following is the detail description of what is the ingredient ontology and its requirement, and how the FCA can be used for construction of such ontology.

4.2 Building Ingredient Ontology

In this thesis, for the purpose of semantic annotation of textual documents of cooking recipes, an ontology is required that describes the recipe according to preparation prototypes of ingredients. Here, preparation prototype means what set of culinary actions can be performed on an ingredient in a recipe. In our hypothesis, this kind of information about ingredient could be useful for replacing an ingredient in a recipe with any other ingredient. In support of our hypothesis, we present the complete annotation process using this kind of ingredient ontology in chapter 5.

Unfortunately, this ontology is not available, therefore we build ingredient ontology using formal concept analysis approach. We apply formal concept analysis on relation schema. The objects of schema are the recipes containing the ingredient and attributes are the culinary actions performed on these ingredient. The FCA then produces the conceptual grouping of recipes of ingredients according to culinary actions. This sort of conceptual grouping is useful in searching the data as it indexes the recipes which speed up searching process. Otherwise searching in relational schema will be quite expensive. Furthermore, relational schema does not support queries to find out subsumption relationships between set of objects. All these scenarios impelled us to use FCA for creating ontology and Description Logic for reasoning over the ontology knowledge base.

In this section, first we present what FCA is. Then, we describe the process of creating ingredient ontology using FCA. We also discuss Description Logic(DL) for the representation of conceptual information contained in Concept Lattice for reasoning over ontology knowledge base.

4.2.1 Formal Concept Analysis

Formal Concept Analysis (FCA) is an approach to identify structure present in domains. Introduced by Wille (see [14] for an overview), FCA is based on a complete lattice of all formal concepts in a domain. A concept in this formalism is an ordered pair of sets, one a set of attributes or descriptors of the concept, the other a set of object indices denoting all instances of the concept in the domain. The set of descriptors of a concept is the maximal set common to all the instances of the concept. These concepts form a partial order from which a concept lattice is constructed. A detailed coverage of Formal Concept Analysis (FCA) is in [14]. The following is the brief description of FCA.

Definition 1. (Formal context) *A formal context is a triple $\prec \mathcal{G}, \mathcal{M}, \mathcal{I} \succ$. $\mathcal{G}$ is a set of objects, $\mathcal{M}$ is a set of descriptors, and I is a binary relation such that $\mathcal{I} \subseteq \mathcal{G} \times \mathcal{M}$*

The notation $\prec y, x \succ \in \mathcal{I}$ or alternatively $y\mathcal{I}x$ is used to express the fact that an object $y \in \mathcal{G}$ has an attribute or descriptor $x \in \mathcal{M}$.

Definition 2. (Formal concept) *A formal concept is a pair of sets $\prec X, Y \succ$. where $Y \subseteq \mathcal{G}$ and $X \subseteq \mathcal{M}$. Each pair must be complete with respect to $\mathcal{I}$, which means that $Y' = X and X' = Y$, where $Y' = \{x \in \mathcal{M} | \forall y \in Y, y\mathcal{I}x\}$ and $X' = \{y \in \mathcal{G} | \forall x \in X, y\mathcal{I}x\}$*

The set of descriptors of a formal concept is called its intent, while the set of objects of a formal concept is called its extent. The formal conept is normally represented in the form $\prec X, Y \succ$. For a set of descriptors $X \subseteq M$, X is the intent of a formal concept if and only if $X'' = X$. A dual condition holds for the extent of a formal concept. This means that any formal concept can be uniquely identified by either its intent or its extent alone. Intuitively, the intent corresponds to a kind of maximally specific description of all the objects in the extent.

The correspondence between intent and extent of complete concepts is a Galois connection between the power set $\mathcal{P}(\mathcal{M})$ of the set of descriptors and the power set $\mathcal{P}(\mathcal{G})$ of the set of objects. The Galois lattice $\mathcal{L}$ for the binary relation is the set of all complete pairs of intents and extents, with the following partial order.

Definition 3. (Concept Order) *Given two concepts $N_1 = \prec X_1, Y_1 \succ$ and $N_2 = \prec X_2, Y_2 \succ$,$N_1 \leq N_2 \iff X_1 \supset X_2$The dual nature of the Galois connection means we have the equivalent relationship $N_1 \leq N_2 \iff Y_1 \subset Y_2$*

For a concept N, I(N) denotes its intent and E(N) denotes its extent.

4.2.2 FCA for building Ingredient Ontology

Formal Concept Analysis(FCA) is a principled way of automatically deriving an ontology from a collection of objects and their properties. It is basically a theory of data analysis which identifies conceptual structures among data sets. A strong feature of FCA is its capability of producing graphical visualizations of the inherent structures among data. This graphical representation then can be converted to other formalism for reasoning over conceptual information derived from FCA.

In this thesis, ontology is also constructed using FCA approach where formal context are converted to concept lattice. Let us suppose that we have following formal context as shown in table 4.1. This information has been extracted from recipe text using the information extraction technique defined in chapter 3. In this example, the objects represent the recipes involving potatoes and properties represent action taken on potatoes in the recipe.

After the FCA process the concept lattice shown in Fig. 4.1 is obtained. Each node in the concept lattice corresponds to single concept and in each concept 'I' stands for intension (properties of concept) and 'E' stands for extension (objects in

Recipe/Potatoes	Peel	Bake	Mash	Boil
1	X	X	X	X
2			X	X
3	X	X		X
4		X	X	
5	X		X	

Table 4.1: Formal Context of recipes

concept). The concept "0" and concept "9" are most general and most specific concept respectively. This classification hierarchy generated using FCA is referred to as an ontology. For our case, it is called "Ingredient Ontology".

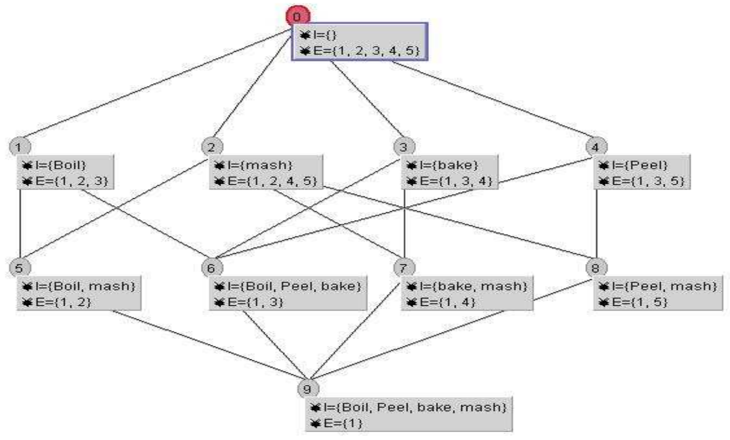


Figure 4.1: Concept Lattice

4.2.3 Ingredient Ontology Representation

Concept lattice is a graphical visualization of the structure in the data. This lattice cannot be used for further reasoning over conceptual knowledge base. For this purpose, the lattice needed to be translated into a formalism suitable for efficient reasoning. Description Logic (DL) is one of the well defined formalism for this purpose. An approach has been outlined in [19] for conversion from concept lattice to DL.

The concept lattice depicted in Fig. 4.1 can be converted to DL representation by using the technique defined in [19]. This paper also discuss Relational Concept Analysis RCA in addition to FCA. Since relational information is not considered in our topic, it won't be discussed.

To convert the lattice, each property or intension (in our lattice property corresponds to cooking or culinary actions) are defined as primitive concept in DL. Then each Concept (Node) in concept lattice is then defined as conjunction ($\sqcap$) of these primitive concepts. This forms the TBox constructs in DL. Each instance in extension of the concept in concept lattice is placed in ABox in corresponding concept of DL. The final DL representation of concept lattice is shown in table 4.2

Concept	TBox	Abox
C1	Boil	1,2,3
C2	Mash	1,2,4,5
C3	Bake	1,3,4
C4	Peel	1,3,5
C5	C1$\sqcap$C2	1,2
C6	C1$\sqcap$C3$\sqcap$C4	1,3
C7	C2$\sqcap$C3	1,4
C8	C2$\sqcap$C4	1,5
C9	C1$\sqcap$C2$\sqcap$C3$\sqcap$C4	1

Table 4.2: DL Knowledge base for Lattice in figure 4.1

4.2.4 Structure of Ingredient Ontology

There are two ways in which we can construct our ingredient ontology. First is one in which we consider the entire ingredient and construct a single ontology. The other is one in which we consider one type of ingredient only e.g. potato, and construct ontology for that type of ingredient. In the second case, there will be a separate ontology for each ingredient.

For the purpose of analyzing the usefulness of the ontologies, we performed experiment on following set of data.

Total Recipe	765	Total Ingredients	358	Total actions	63

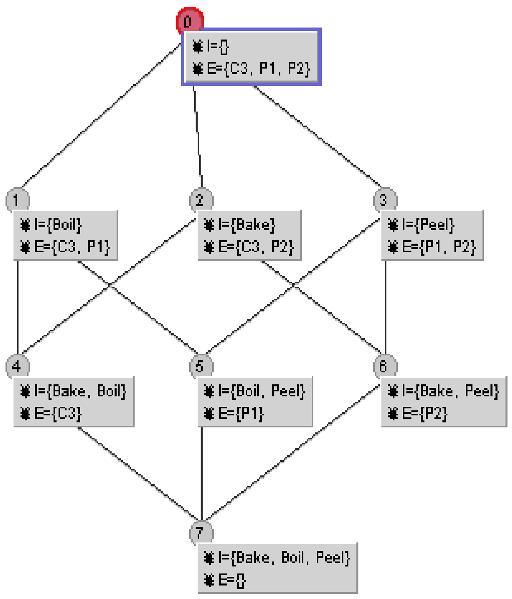
For experiment, we collected ingredient-action relationship from above recipe data using the approaches defined in chapter 3. We then apply FCA to construct the ontologies. The script for creating lattice from extracted information is available in accompanied CD with this thesis. The script creates the file that can be loaded in Galicia [1] tool for the purpose of generating ontologies using FCA approach. The following are the statistics of the two ontologies obtained as result of above experiments.

Ontology Type	**Recipes**	**Ingredients**	**Actions**	**Concepts**
Combined Ingredient Ontology	765	2654	63	162
Single Ingredient Ontology	366	366	21	47

Table 4.3: Ingredient Ontology

The two ontologies are not directly comparable as there are different numbers of recipes used. But the close analysis of our data and resultant concept lattice structure encouraged us to conclude that combined ingredient ontology is not reasonable because it might lose ingredient prototype information for some ingredient. For example, suppose there are two ingredients, potato in recipe 1 & 2 and carrot in recipe 3, as shown in table 4.4.

Ingredient	Peel	Boil	Bake
P1	X	X	
P2	X		X
C3		X	X



Ingredient	Peel	Boil	Bake
P1	X	X	
P2	X		X

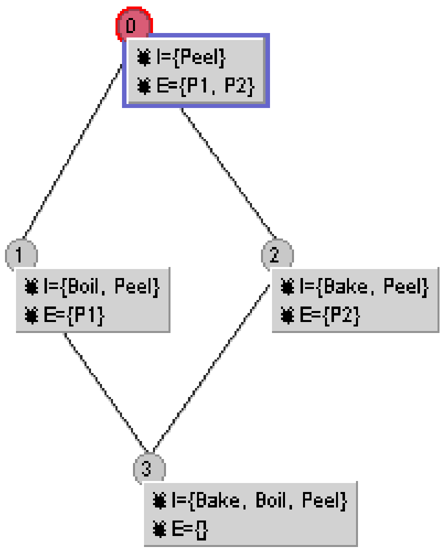


Table 4.4: Combined Potato & Carrot Ontology on left side and Individual Potato Ontology on right side

If we don't consider C3, then potato has preparation prototype according to ontology on right side of table 4.4. There are only two prototypes defined by concept 1 and 2. In both concepts, "Peel "action is necessary. While with C3, potato prototypes increases to 5 as shown in left side of table 4.4. In this ontology, the information that "Peel" action is mandatory for potato is also lost. Hence we can say that combined ontology causes increase in prototypes and cannot present prototype information efficiently. We recommend here single/individual ingredient ontology for our annotation process.

4.3 Conclusion

This chapter discussed automatic ontology construction from text. Automatic ontology construction is not a trivial task. This chapter highlighted different method or techniques in this regard e.g. InfoSleuth [20], Text-To-Onto [26], Ontolearn [27], OntoLT [8] and GlossOnt [28]. The chapter also gave a brief review of the Formal Concept Analysis(FCA) based ontology construction process which is the technique that is followed in building ingredient ontology in this thesis. The chapter showed that how ingredient ontology can be built. For this purpose, two different ontology experiments (one in which all the ingredients were included in ontology and other one in which ontology representing single ingredient) were conducted to figure out the best option. It was found that single ingredient ontology will be reasonable for SA in our work and ultimately for recipe adaptation.

Chapter 5

Annotation Process

There are three step involved in our annotation process. This annotation process uses the ingredient ontology described previously. The process involves extraction of actions, ingredients, their mapping from the recipe text and representing this information in a format suitable for ontology lookup, second step is ontology lookup i.e. searching the appropriate concept in the ontology corresponding to ingredient-action information extracted, and third is representing these annotations. Note that we do not provide any evaluation for our annotation process. The reason for not providing evaluation is the unavailability of standard techniques for evaluation and subsequent difficulty in development of corpus for evaluation.

In the following sections, these steps are described in more detail.

5.1 Ingredients-Actions Extraction

Let us formally describe what kind of information is required for annotation process. Let us suppose that there is a recipe "R" with ingredients $I_1, I_2, I_3...I_n$. In recipe "R", there are some set of culinary actions that are performed on ingredients. We want to extract information about ingredient "I" in this recipe with corresponding culinary actions and represent this information in a format suitable for ontology lookup. Let us suppose that ingredient "I" has action $A_1, A_2, A_3...A_n$ performed on it in the recipe "R". This information is extracted according to process defined in section 3.3 in chapter 3. This information is then represented in DL format as follow

$$I_R = A_1 \sqcap A_2 \sqcap A_3 \sqcap ... \sqcap A_n$$

Suppose that we have the following recipe of potato. We follow here the step by step algorithm defined in chapter 3 for extracting information from this example.

"...Boil potatoes................................ Bake potatoes 220C in oven................"

POS Tagging

After POS-tagging the recipe looks as follow.

> *"...Boil/VB potatoes/NNS................................ Bake/VB potatoes/NNS 220C/NN in/IN oven/NN................"*

Tagging Ingredients and Actions

Using dictionary, the ingredient (Nouns) and actions (Verbs) are searched and are tagged with "ING"and "ACT" respectively.

> *"...Boil/ACT potatoes/ING................................ Bake/ACT potatoes/ING 220C/NN in/IN oven/NN................"*

Ingredient-Actions Mapping

Using our grammar rules defined in chapter 3, two patterns are matched. ACT ING (Boil potatoes, Bake Potatoes). At the end, the new ingredient prototype that is extracted from the recipe is represented in DL as

$$Potato = Boil \sqcap Bake$$

5.2 Ontology Lookup

Now the task here is to look up the extracted prototype in ontology. There are two types of ontology lookup

1. Look for equivalent concept in ontology i.e. $I_R \equiv C_i$ where I_R is the ingredient "I" in recipe "R" and C_i is a concept in ontology.
2. If equivalence relation does not hold, look for subsumption relation i.e. $I_R \sqsubset C_i$.

This can be done using well defined inference algorithm for Description Logic like Structural Subsumption and Tableaux. We will only discuss how structural subsumption can be used for this task.

In structural subsumption technique, the syntactic structures of normalized concepts are compared to check the consistency of the knowledge. In this reasoning technique, the DL has following form which belongs to $\mathcal{FL}0$ family of DL

$$\begin{array}{lll} \text{C,D} \implies & \text{A} & |\text{(atomic concept)} \\ & \text{C} \sqcap \text{D} & |\text{(intersection)} \\ & \forall \text{ R.C} & |\text{(value restriction)} \end{array}$$

A concept is said to be in normalized form, if it has a following form

$$A_1 \sqcap ... \sqcap A_m \sqcap \forall R_1.C_1 \sqcap ... \sqcap \forall R_n.C_n$$

Let us suppose that we have following two concepts 'C' and 'D' in normal form

Normal form of C: $A_1 \sqcap ... \sqcap A_m \sqcap \forall R_1.C_1 \sqcap ... \sqcap \forall R_n.C_n$
Normal form of D: $B_1 \sqcap ... \sqcap B_k \sqcap \forall S_1.D_1 \sqcap ... \sqcap \forall S_l.D_l$

Then C $\sqsubset$ D iff:

- for all $i, 1 \leq i \leq k$, there exists $j, 1 \leq j \leq m$ such that $B_i = A_j$
- for all $i, 1 \leq i \leq l$, there exists $j, 1 \leq j \leq n$ such that $S_i = R_j$ and $C_j \sqsubset D_i$

Equivalence relation between concept "C" and "D" is defined in term of subsumption relationship as

$$R \equiv C \iff (R \sqsubset C \sqcap C \sqsubset R)$$

The structural subsumption algorithm is simply the comparison of all the primitive concepts, sub concept and relations one by one. Since in our case, we don't have any relational information, therefore our DL syntax is quite simple. This syntax is more or less similar to propositional logic. Therefore, the structural subsumption algorithm will compare only primitive and sub concepts.

We classify extracted potato prototype($Potato = Boil \sqcap Bake$) in our DL knowledge base presented in table 4.2 to see which ontological prototype this new prototype refers to. First it is check for equivalence relationship. Since there is no prototype in equivalence relationship with potato prototype therefore "Potato" in given recipe cannot be annotated with any prototype using equivalence relationship.

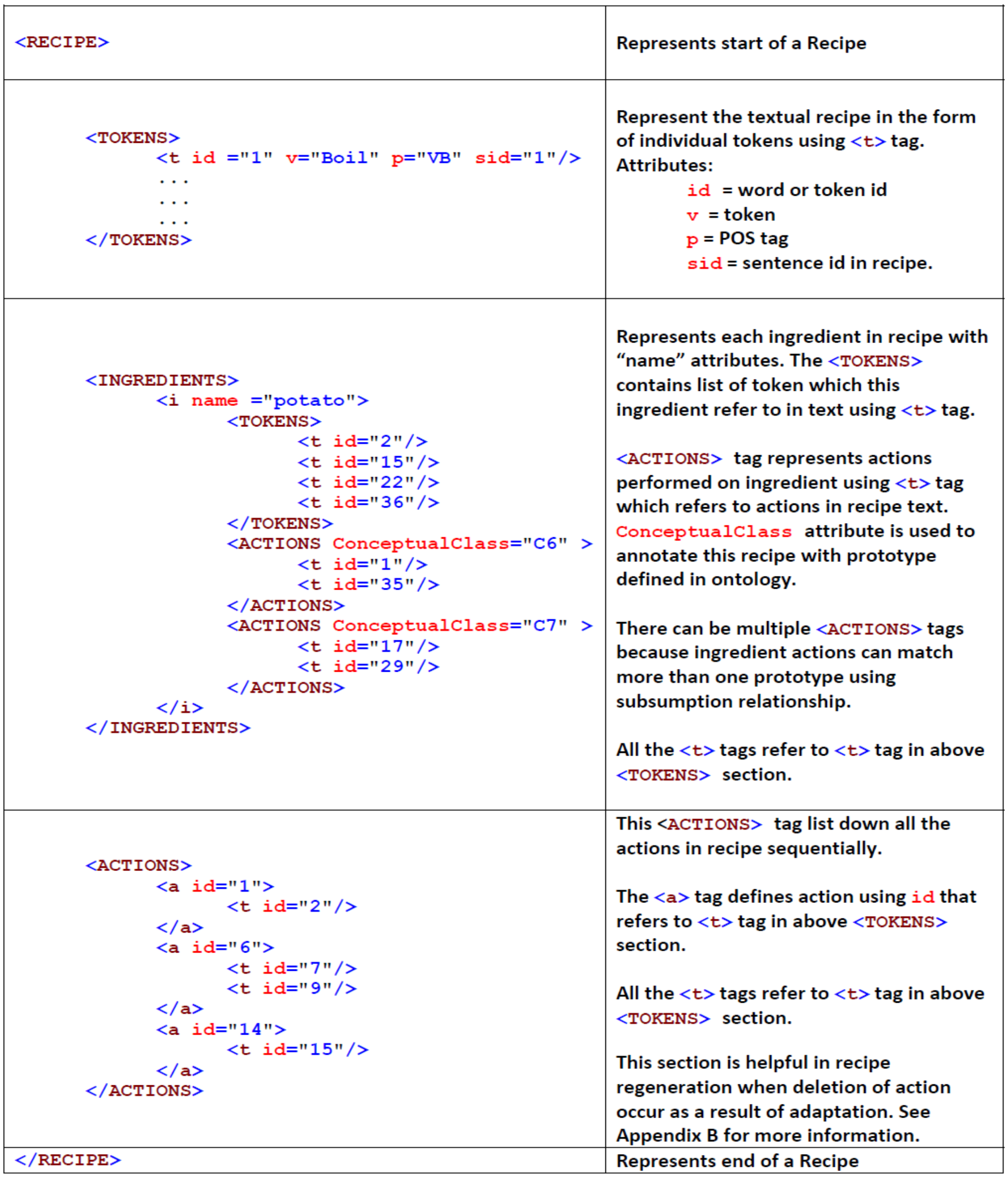
This prototype is then searched for subsumption relationship and it is worked out that

$$Potato \sqsubset C1 \text{ and } Potato \sqsubset C3$$

Since, our new ingredient prototype could not be completely matched i.e. is not equal to any prototype in ontology that's why we will keep both annotation C1 and C3 in our annotation representation for this recipe.

5.3 Representation of Annotation

The Fig. 5.1 presents the representation format used for representing our annotation.



```
<RECIPE>
```

Represents start of a Recipe

```
<TOKENS>
      <t id ="1" v="Boil" p="VB" sid="1"/>
      ...
      ...
      ...
</TOKENS>
```

Represent the textual recipe in the form of individual tokens using <t> tag.
Attributes:

- id = word or token id
- v = token
- p = POS tag
- sid = sentence id in recipe.

```
<INGREDIENTS>
      <i name ="potato">
            <TOKENS>
                  <t id="2"/>
                  <t id="15"/>
                  <t id="22"/>
                  <t id="36"/>
            </TOKENS>
            <ACTIONS ConceptualClass="C6" >
                  <t id="1"/>
                  <t id="35"/>
            </ACTIONS>
            <ACTIONS ConceptualClass="C7" >
                  <t id="17"/>
                  <t id="29"/>
            </ACTIONS>
      </i>
</INGREDIENTS>
```

Represents each ingredient in recipe with “name” attributes. The <TOKENS> contains list of token which this ingredient refer to in text using <t> tag.

<ACTIONS> tag represents actions performed on ingredient using <t> tag which refers to actions in recipe text. ConceptualClass attribute is used to annotate this recipe with prototype defined in ontology.

There can be multiple <ACTIONS> tags because ingredient actions can match more than one prototype using subsumption relationship.

All the <t> tags refer to <t> tag in above <TOKENS> section.

```
<ACTIONS>
      <a id="1">
            <t id="2"/>
      </a>
      <a id="6">
            <t id="7"/>
            <t id="9"/>
      </a>
      <a id="14">
            <t id="15"/>
      </a>
</ACTIONS>
```

This <ACTIONS> tag list down all the actions in recipe sequentially.

The <a> tag defines action using id that refers to <t> tag in above <TOKENS> section.

All the <t> tags refer to <t> tag in above <TOKENS> section.

This section is helpful in recipe regeneration when deletion of action occur as a result of adaptation. See Appendix B for more information.

```
</RECIPE>
```

Represents end of a Recipe

Figure 5.1: Representation format used for annotation

5.4 Use of Annotation

This kind of annotation can be used for many purposes such as efficient answering of search queries, clustering, text summarization etc. Since our domain of work is recipe document, we use this sort of annotation for adaption purpose.

5.4.1 Adaptation

In recipe, adaption is the process of intelligently replacing ingredient(s) from recipe with other ingredient(s) to make recipe adapted for replacing ingredient(s). In following examples, we show how recipe can be adapted using our annotation. Followig are the rules for recipe adaptation.

1 - If both ingredients have same prototype, then simply replace instances of first ingredient in text with second one. The information about traces of first ingredient in text is obtained from $< INGREDIENTS >$ section in our annotation format.

2 - If prototypes are not same then we have following two cases.

  2.1 - Add Action - if the replacing prototype contains action that are not in the current prototype then these action have to be accommodated. The sentence is created in the format "action followed by ingredient" e.g. "Boil carrot" and put in the appropriate place in text.

  2.2 - Delete Action - actions are deleted which are not possible for replacing ingredient. Sentence containing action is deleted if there is no other action in sentence or no other ingredient in sentence on which this action is being performed. Otherwise if there is another action then sentence is reformulated using the $< ACTIONS >$ section in our annotation representation without action which has to be deleted. If there are other ingredients in the sentence on which this action is being performed then delete the ingredient from this sentence and reformulate the sentence using $< ACTIONS >$ section.

5.4.2 A trivial Potato to Carrot Adaption

This example follows adaptation rule 1. For this example, the two ontology for potato and carrot shown in Fig. 5.2 are considered.

Let us consider the following recipe. In this recipe, the red color text is culinary actions and blue color text is ingredient(This scheme will be followed for all the examples). Here we follow step by step approach to show the annotation and adaption process

> *"Boil potatoes until done. Saute butter and garlic in large skillet. Add potatoes and toss to coat. Put potatoes in baking dish; layer all ingredients except for the sour cream. Bake potatoes until cheese is melted."*

Step 1: Information Extraction

The above recipe is analyzed for extracting potato and culinary action performed on potato. When our information extraction technique defined in section 5.1 is applied

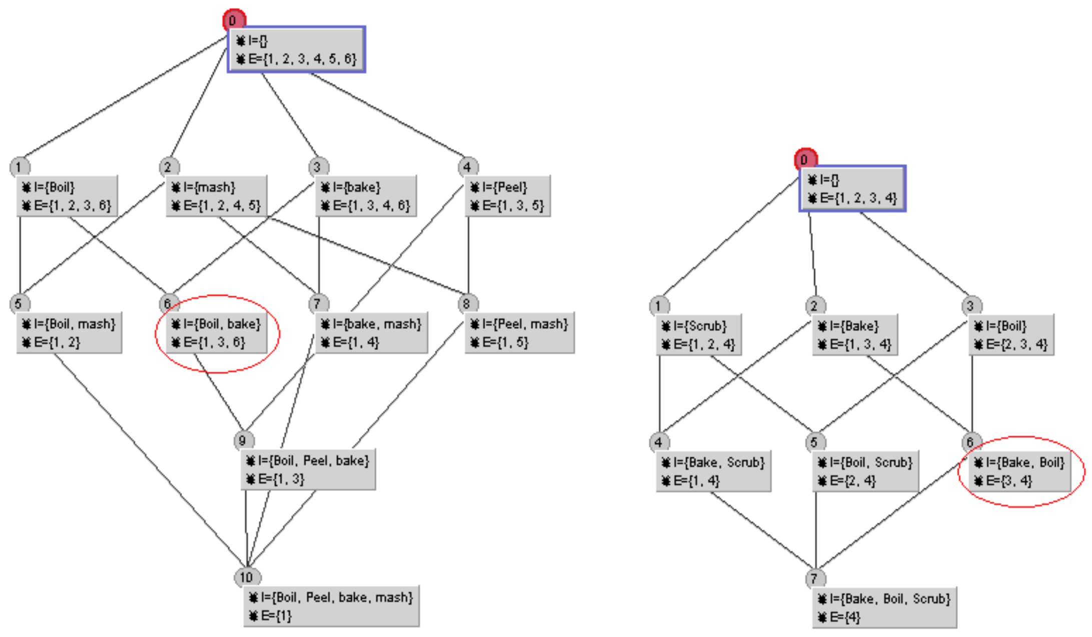


Figure 5.2: Potato and Carrot Ontology

on above recipe the following DL representation is obtained.

$$Potato = Boil \sqcap Bake$$

Step 2: Ontology Lookup

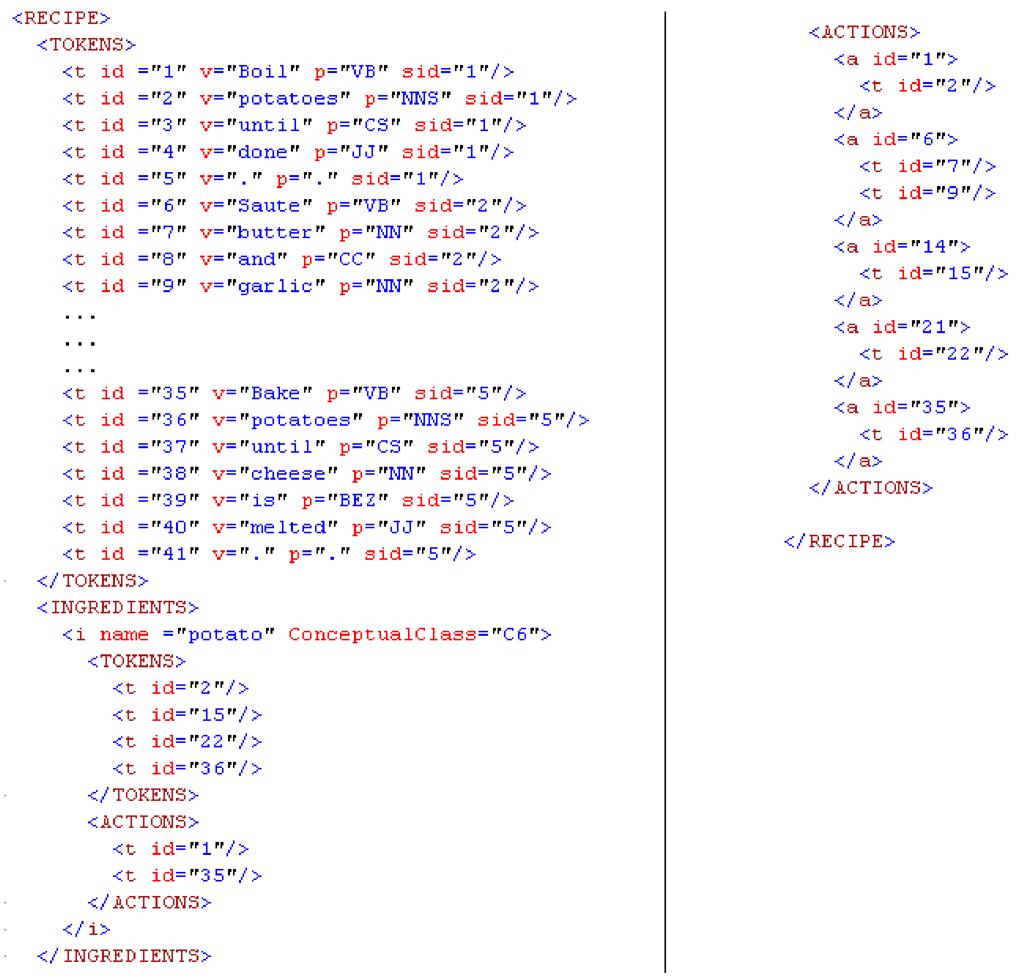
The potato ontology in Fig. 5.2 is looked up using structural subsumption algorithm to see any potato prototype matches with one defined by given recipe. It is found out that there exists such prototype defined by prototype 6 as shown by red circle in Fig. 5.2. Therefore, the recipe is annotated with prototype 6 in potato ontology.

Step 3: Representation

The Fig. 5.3 shows the final representation of our annotation. In ingredient sections, we have only shown potato for simplicity. Its annotation in ontology is prototype 6 i.e. "C6".

Step 4: Adaptation

Is it possible to adapt given potato recipe for carrot?. Let see if it is possible. Potato here is described by "Potato (Boil, Bake)" prototype. As we have already developed ingredient ontology for each ingredient, we search for our carrot ontology for best prototype that can replace potato here. It is found that the prototype "Carrot (Boil, Bake)" exists in carrot ontology as shown by red circle on right side of Fig. 5.2.



```
<RECIPE>
  <TOKENS>
    <t id ="1" v="Boil" p="VB" sid="1"/>
    <t id ="2" v="potatoes" p="NNS" sid="1"/>
    <t id ="3" v="until" p="CS" sid="1"/>
    <t id ="4" v="done" p="JJ" sid="1"/>
    <t id ="5" v="." p="." sid="1"/>
    <t id ="6" v="Saute" p="VB" sid="2"/>
    <t id ="7" v="butter" p="NN" sid="2"/>
    <t id ="8" v="and" p="CC" sid="2"/>
    <t id ="9" v="garlic" p="NN" sid="2"/>
    ...
    ...
    ...
    <t id ="35" v="Bake" p="VB" sid="5"/>
    <t id ="36" v="potatoes" p="NNS" sid="5"/>
    <t id ="37" v="until" p="CS" sid="5"/>
    <t id ="38" v="cheese" p="NN" sid="5"/>
    <t id ="39" v="is" p="BEZ" sid="5"/>
    <t id ="40" v="melted" p="JJ" sid="5"/>
    <t id ="41" v="." p="." sid="5"/>
  </TOKENS>
  <INGREDIENTS>
    <i name ="potato" ConceptualClass="C6">
      <TOKENS>
        <t id="2"/>
        <t id="15"/>
        <t id="22"/>
        <t id="36"/>
      </TOKENS>
      <ACTIONS>
        <t id="1"/>
        <t id="35"/>
      </ACTIONS>
    </i>
  </INGREDIENTS>

  <ACTIONS>
    <a id="1">
      <t id="2"/>
    </a>
    <a id="6">
      <t id="7"/>
      <t id="9"/>
    </a>
    <a id="14">
      <t id="15"/>
    </a>
    <a id="21">
      <t id="22"/>
    </a>
    <a id="35">
      <t id="36"/>
    </a>
  </ACTIONS>

</RECIPE>
```

Figure 5.3: Representation of annotation for example recipe

Therefore, adaptation is trivial string replacement between potato instances found in the text with carrot. The <INGREDIENTS> section in our annotation format tells us that Potato should be replaced at $2^{nd}, 15^{th}, 22^{th}$ and 36^{th} position in text. Since no action is being changed so sentence reformulation is not required. The adapted recipe is

> *"Boil carrots until done. Saute butter and garlic in large skillet. Add carrots and toss to coat. Put carrots in baking dish; layer all ingredients except for the sour cream. Bake carrots until cheese is melted."*

5.4.3 Adaption using Deletion of Actions

Now consider following example. For this example, the ontology shown in Fig. 5.4 is used. The ontology on left side is for potato and on right side is for cabbage.

"Peel potatoes and cut into large pieces. Boil in salted water for 15 to 20 minutes, or until tender. Drain potatoes. Mash potatoes in large bowl. Add milk in small amounts, beating after each addition, until desired consistency is reached. Add butter, 1/4 teaspoon salt, and pepper. Beating until mashed potatoes are light and fluffy."

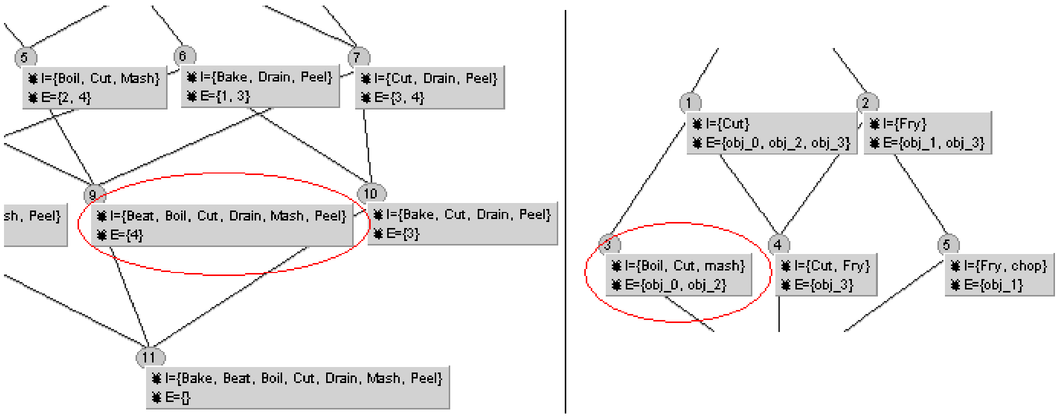


Figure 5.4: Patato and Cabbage Ontology

Step 1: Information Extraction

Our information extraction technique produces following DL represention for the potato in this recipe.

$$Potato = Peel\ \sqcap Cut\ \sqcap Boil\ \sqcap Drain\ \sqcap Mash\ \sqcap Beat$$

Step 2: Ontology Lookup

The ontology look up process annotates the recipe with prototype 9 in the potato ontology as shown in Fig. 5.4 using red circle.

Step 3: Representation

The Fig. 5.5 shows the final representation of our annotation. In ingredient sections, we have only shown potato for simplicity. Its annotation in ontology is prototype 9 i.e. "C9".

Step 4: Adaptation

Now, we want to cook mashed potato but unfortunately we don't have potato available. Instead, we have cabbages. Can we turn this recipe into mashed cabbages recipe? After searching our ontology of cabbages, we found that there is no prototype that matches with mash potato recipe. Then, we search for any subsumption

```
<RECIPE>
  <TOKENS>
    <t id ="1" v="Peel" p="VB" sid="1"/>
    <t id ="2" v="potatoes" p="NNS" sid="1"/>
    <t id ="3" v="and" p="CC" sid="1"/>
    <t id ="4" v="cut" p="VB" sid="1"/>
    <t id ="5" v="into" p="IN" sid="1"/>
    ...
    ...
    ...
    <t id ="59" v="Beating" p="VB" sid="6"/>
    <t id ="60" v="until" p="CS" sid="6"/>
    <t id ="61" v="mashed" p="JJ" sid="6"/>
    <t id ="62" v="potatoes" p="NNS" sid="6"/>
    ...
    ...
  </TOKENS>
  <INGREDIENTS>
    <i name ="potato">
      <TOKENS>
        <t id="2"/>
        <t id="24"/>
        <t id="27"/>
        <t id="62"/>
      </TOKENS>
      <ACTIONS ConceptualClass="C9" >
        <t id="1"/>
        <t id="4"/>
        <t id="9"/>
        <t id="23"/>
        <t id="26"/>
        <t id="59"/>
      </ACTIONS>
    </i>
  </INGREDIENTS>
```

```
    <ACTIONS>
      <a id="1">
        <t id="2"/>
      </a>
      <a id="4">
      </a>
      <a id="9">
      </a>
      <a id="24">
        <t id="27"/>
      </a>
      <a id="26">
        <t id="27"/>
      </a>
      <a id="32">
        <t id="33"/>
      </a>
      <a id="38">
      </a>
      <a id="49">
        <t id="50"/>
        <t id="54"/>
        <t id="57"/>
      </a>
      <a id="59">
        <t id="62"/>
      </a>
    </ACTIONS>

  </RECIPE>
```

Figure 5.5: Representation of annotation for example recipe

relation and it is found that we have cabbages recipe that has prototype "cabbage (Cut, Boil, mash)" as shown in cabbage ontology in Fig. 5.4 using red circle. There are three culinary actions (Peel, Drain and Beat) that are not possible for the case of cabbages in our ontology. First of all the instances of potatoes are replaced with cabbages. Then actions are deleted which are not necessary for cabbages. Here, adaptation rule 2.2 is applied. "Peel" action cannot be directly deleted because our $< ACTIONS >$ section says that there is another action in sentence one. Therefore sentence one is regenerated from $< ACTIONS >$ sections. The sentences containing "Drain" and "Beat" can be directly deleted as there are no other actions or ingredient in these sentences. At the end, adapted recipe looks like

> *" Cut cabbages. Boil in salted water for 15 to 20 minutes, or until tender. Mash cabbages in large bowl. Add milk in small amounts, beating after each addition, until desired consistency is reached. Add butter, 1/4 teaspoon salt, and pepper."*

5.4.4 Adaption using Addition and Deletion of Actions

Let us consider following fried chicken example now. This time we only show here adaptation process. Complete representation of annotation of this recipe is presented in appendix A.

> *"Clean the chicken and dry it with kitchen towel. Marinate chicken with salt, pepper, lemon juice & corn flour (preferably over night).Heat oil in a big pan and fry chicken on high flame keep on changing direction of chicken so it should cook equally from all sides."*

Can we replace chicken with potato? Here chicken is defined by "Chicken (Clean, Dry, Marinate, Fry)". After searching for appropriate potato prototype we get "Potato (Peel, Slice, Fry)". This time we not only delete the actions but also have to add new actions to make recipe adapted for potato. The sentence one is analyzed first. It contains actions that need to be deleted. According to adaptation rule 2.2 we can delete this sentence completely as it contains only actions that need to be deleted. Furthermore, it does not contain any other ingredients. "Marinate" also need to be deleted but there are other ingredients which are associated with "Marinate" here. We cannot delete this sentence or "Marinate" action completely instead we delete "chicken" from here. This makes sentence unnatural. For addition of "Peel" and "Slice" actions, the adaptation rule 2.1 is followed i.e. we simply add sentence at the start as "Peel and slice potatoes". The adapted recipe is as follow.

> *"Peel and slice potatoes. Marinate with salt, pepper, lemon juice & corn flour (preferably over night). Heat oil in a big pan and fry potatoes on high flame keep on changing direction of potatoes so it should cook equally from all sides."*

5.5 Conclusion

This chapter has presented the brief description of our annotation process. Three crucial task i.e. Information Extraction (IE) and representation of information from recipe text, Ontology Lookup, and representation of metadata or annotation that were created as result of our annotation process. IE process is the same as described in chapter 3. But after extraction the information is presented in Description Logic(DL) format. This information is then looked up in ontology for conceptual class it belongs to. Since both extracted information and ontology are in DL format, therefore, standard DL reasoning and inference technique can be applied for ontology lookup. The chapter detailed structural subsumption algorithm for this purpose. Once metadata is available, it has to be represented in a format suitable for further processing. The chapter has presented an XML based representation for annotation. At the end, few examples were explained to show that our approach for SA is really useful for recipe adaptation purpose.

Chapter 6

Conclusions

In this thesis, we have shown a new approach for semantic annotation. In this kind of annotation, the document is annotated according to the conceptual information it contains. This conceptual information is described by ontologies. In ontology, the concepts that formally describe information are equipped with set of properties. Any document that contains these properties is annotated with corresponding concept in ontology.

To mine these properties, we have developed two robust text analysis techniques. One is based on shallow parsing approach and other is based on dependency parsing approach. In shallow parsing approach which we call Rule based information extraction, grammatical rules or patterns have been designed to extract the information from the text. Rule based information extraction requires the POS tagger, morphological analyzer and set of dictionaries to work. The other approach to information extraction is dependency based information extraction. In this approach, the dependency analysis of the text is performed to extract out information. This approach is quite handy as it does not require the use of POS tagger. But morphological analyzer and dictionaries are still necessary to extract information. We have also performed the evaluation of two approaches on the corpus of recipe documents. The results of the evaluation have been presented in chapter 3. According to results obtained from the experiments, it was concluded that "Dependency Based" Information Extraction can be helpful for the scenarios discuss in this thesis.

The information extraction techniques are used for both construction of ontology and semantic annotation. We have also described a method of ontology construction using Formal Concept Analysis (FCA) approach. Once ontology is constructed, it is represented in the system using Description Logic formalism. Then annotation has been defined as mining appropriate properties in the document, converting these properties to DL based representation and looking up ontology for corresponding concept for annotation. These annotations are formally represented using xml based format defined in chapter 5. We apply our approach on cooking recipe corpus for recipe adaption purpose. Examples are given at the end of chapter 5 to show adaptation process using our annotation approach.

There are still some open issues that need to be addressed for efficient adaption. Information extraction techniques devised for this approach require further improvement as lots of information is not extracted by our current process. Especially, discourse analysis process should be improved. Furthermore, for recipe adaptation, interaction of ingredient should also be taken into consideration as in our point of view it could be helpful. The other important issue is the consideration of adverbial clauses like how much to cook, time, pot etc. Currently, adverbs are not annotated that makes regeneration of sentences little bit unnatural. We will try to solve these issues in our future work.

Bibliography

[1] Galicia: http://www.iro.umontreal.ca/galicia/. 2005. [cited at p. 43]

[2] Harith Alani, Sanghee Kim, David E. Millard, Mark J. Weal, Wendy Hall, Paul H. Lewis, and Nigel Shadbolt. Web based knowledge extraction and consolidation for automatic ontology instantiation. In *2 nd Int. Conf. Knowledge Capture (KCap'03), Workshop on Knowledge Markup and Semantic Annotation*, 2003. [cited at p. 10]

[3] Popov B., Kiryakov A., Kirilov A., Manov D., Ognyanoff D., and Goranov M. Kim - semantic annotation platform. *2nd International Semantic Web Conference-ISWC2003*, pages 834–849, 2003. Florida (USA). [cited at p. 2, 8, 9, 13]

[4] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook.* Cambridge University Press, 2003. [cited at p. 12]

[5] Michael Bain. *Advances in Artificial Intelligence*, volume 2903, chapter Inductive Construction of Ontologies from Formal Concept Analysis, pages 88–99. Springer Berlin / Heidelberg, February 19 2004. [cited at p. 39]

[6] Erol Bozsak, Marc Ehrig, Siegfried Handschuh, Andreas Hotho, Alexander Maedche, Boris Motik, Daniel Oberle, Christoph Schmitz, Steffen Staab, Ljiljana Stojanovic, Nenad Stojanovic, Rudi Studer, Gerd Stumme, York Sure, Julien Tane, Raphael Volz, and Valentin Zacharias. Kaon - towards a large scale semantic web. In *In Proceedings of the Third International Conference on E-Commerce and Web Technologies.* Springer Lecture Notes in Computer Science., 2002. [cited at p. 11]

[7] Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2005. [cited at p. 39]

[8] Paul Buitelaar, Daniel Olejnik, and Michael Sintek. A protg plug-in for ontology extraction from text based on linguistic analysis. In *1st European Semantic Web Symposium*, pages 31–44, Heraklion, 2004. [cited at p. 39, 45]

[9] Fabio Ciravegna. Designing adaptive information extraction for the semantic web in amilcare. *Annotation for the Semantic Web, Frontiers in Artificial Intelligence and Applications*, 2003. [cited at p. 9]

[10] Fabio Ciravegna, Alexiei Dingli, Daniela Petrelli, and Yorick Wilks. User-system cooperation in document annotation based on information extraction. *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 02)*, October 2002. Sigenza (Spain). [cited at p. 2, 8, 9, 13]

[11] Domingue, Dr John, Dzbor, Dr Martin, Motta, and Prof Enrico. Magpie: Browsing and navigating on the semantic web. *Proceedings ACM Conference on Intelligent User Interfaces (IUI)*, pages 191–197, January 2004. Portugal. [cited at p. 2, 8, 9, 13]

[12] Jrme Euzenat. Eight questions about semantic web annotations. *IEEE INTELLIGENT SYSTEMS*, 17(2):55–62, 2002. [cited at p. 7, 8]

[13] D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2002. [cited at p. 11]

[14] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin, 1999. [cited at p. 39, 40]

[15] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In *In Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer*, 1993. [cited at p. 1, 7, 11]

[16] Cunningham H., Maynard D., Bontcheva K., and Tablan V. Gate: A framework and graphical development environment for robust nlp tools and applications. *40th Anniversary Meeting of the Association for omputational Linguistics-ACL'02*, 2002. [cited at p. 2, 8, 13]

[17] Cunningham H., Maynard D., and Tablan. Jape: A java annotation patterns engine. 2000. [cited at p. 9]

[18] Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-cream : Semi-automatic creation of metadata. *Semantic Authoring, Annotation & Knowledge Markup - Preliminary Workshop Programme*, 2002. [cited at p. 9]

[19] Marianne Huchard, Amedeo Napoli, Mohamed Rouane Hacene, and Petko Valtchev. *Selected Contributions in Data Analysis and Classification*, chapter Mining Description Logics Concepts with Relational Concept Analysis, pages 259–270. Springer Berlin Heidelberg, 2007. [cited at p. 42]

[20] Chung Hee Hwang. Incompletely and imprecisely speaking: Using dynamic ontologies for representing and retrieving information. *The 6th International Workshop on Knowledge Representation*, pages 14–20, 1999. [cited at p. 39, 45]

[21] Hans Kamp. A theory of truth and semantic representation. In Stokhof (eds.) Groenendijk, Janssen, editor, *Formal Methods in the Study of Language, Mathematisch Centrum, Amsterdam*, 1981. [cited at p. 29]

[22] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. Semantic annotation, indexing, and retrieval. *Elsevier's Journal of Web Semantics*, 2, 2005. [cited at p. 7]

[23] Paul Kogut and William Holmes. Aerodaml: Applying information extraction to generate daml annotations from web pages. In *First International Conference on Knowledge Capture (K-CAP 2001). Workshop on Knowledge Markup and Semantic Annotation*, 2001. [cited at p. 9]

[24] Maurizio Lenzerini, Diego Milano, and Antonella Poggi. Ontology representation and reasoning. [cited at p. 12]

[25] Vargas-Vera M., Motta E., Domingue J., Lanzoni M., Stutt A., and Ciravegna F. Mnm: Ontology driven semi-automatic and automatic support for semantic markup. *The 13th International Conference on Knowledge Engineering and Management (EKAW2002)*, pages 379–391, 2002. Spain. [cited at p. 2, 8, 9, 13]

[26] A. Maedche and S. Staab. Semi-automatic engineering of ontologies from text. In *In Proceedings of the 12th Internal Conference on Software and Knowledge Engineering*, pages 231–239, 2000. [cited at p. 39, 45]

[27] Roberto Navigli, Paola Velardi, and Aldo Gangemi. Ontology learning and its application to automated terminology translation. *IEEE Intelligent Systems*, 18(1):22–31, 2003. [cited at p. 39, 45]

[28] Youngja Park. Glossont: A concept-focused ontology building tool. In *KR*, pages 498–506. Whistler, 2004. [cited at p. 39, 45]

[29] The Standford Parser. http://nlp.stanford.edu/software/lex-parser.shtml. [cited at p. 22, 30]

[30] Ulrike Sattler. http://www.cs.man.ac.uk/ sattler/reasoners.html. [cited at p. 13]

[31] Berners-Lee T., Hendler J., and Lassila. *The Semantic Web*. Scientific American, 2001. [cited at p. 1]

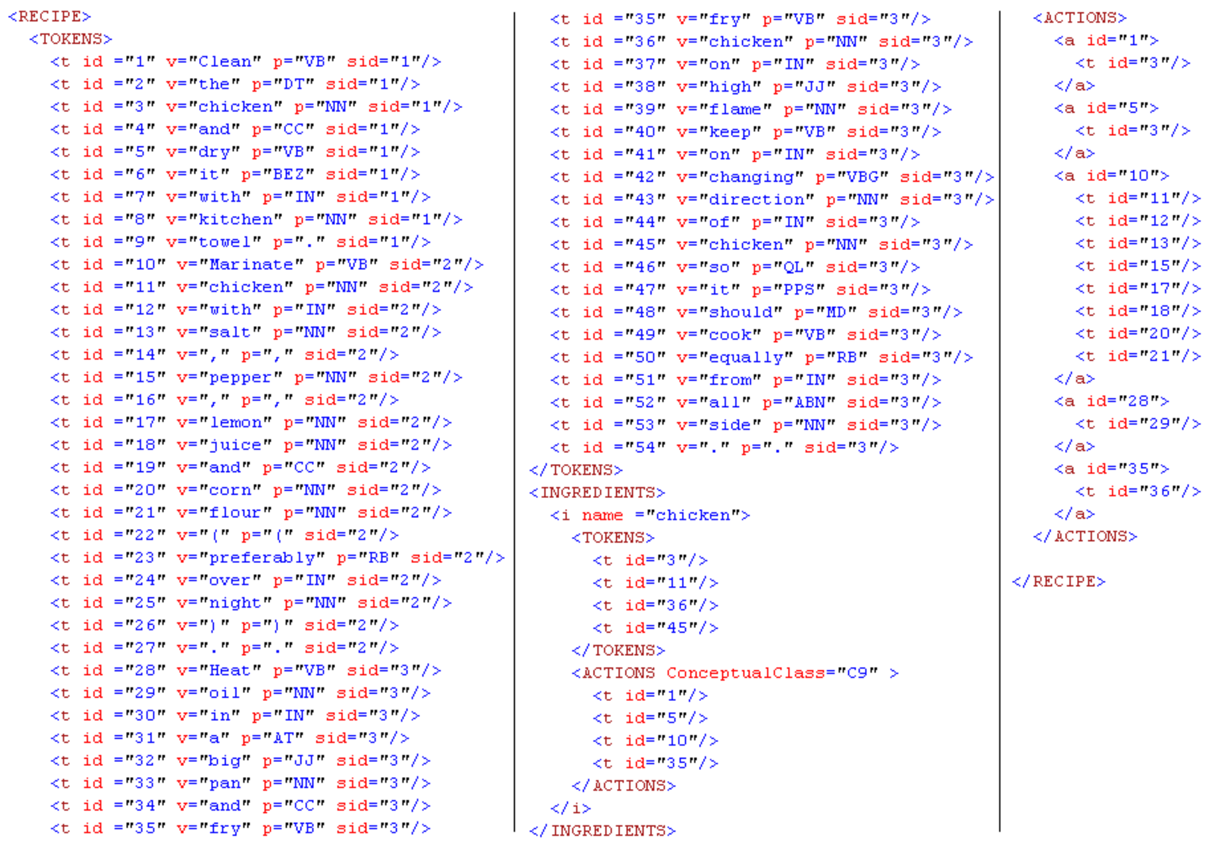
[32] The Standford POS Tagger. http://nlp.stanford.edu/software/tagger.shtml. [cited at p. 30]

Appendices

Appendix A

Annotation Representation

A.1 Complete Representation of Recipe



```
<RECIPE>
  <TOKENS>
    <t id ="1" v="Clean" p="VB" sid="1"/>
    <t id ="2" v="the" p="DT" sid="1"/>
    <t id ="3" v="chicken" p="NN" sid="1"/>
    <t id ="4" v="and" p="CC" sid="1"/>
    <t id ="5" v="dry" p="VB" sid="1"/>
    <t id ="6" v="it" p="BEZ" sid="1"/>
    <t id ="7" v="with" p="IN" sid="1"/>
    <t id ="8" v="kitchen" p="NN" sid="1"/>
    <t id ="9" v="towel" p="." sid="1"/>
    <t id ="10" v="Marinate" p="VB" sid="2"/>
    <t id ="11" v="chicken" p="NN" sid="2"/>
    <t id ="12" v="with" p="IN" sid="2"/>
    <t id ="13" v="salt" p="NN" sid="2"/>
    <t id ="14" v="," p="," sid="2"/>
    <t id ="15" v="pepper" p="NN" sid="2"/>
    <t id ="16" v="," p="," sid="2"/>
    <t id ="17" v="lemon" p="NN" sid="2"/>
    <t id ="18" v="juice" p="NN" sid="2"/>
    <t id ="19" v="and" p="CC" sid="2"/>
    <t id ="20" v="corn" p="NN" sid="2"/>
    <t id ="21" v="flour" p="NN" sid="2"/>
    <t id ="22" v="(" p="(" sid="2"/>
    <t id ="23" v="preferably" p="RB" sid="2"/>
    <t id ="24" v="over" p="IN" sid="2"/>
    <t id ="25" v="night" p="NN" sid="2"/>
    <t id ="26" v=")" p=")" sid="2"/>
    <t id ="27" v="." p="." sid="2"/>
    <t id ="28" v="Heat" p="VB" sid="3"/>
    <t id ="29" v="oil" p="NN" sid="3"/>
    <t id ="30" v="in" p="IN" sid="3"/>
    <t id ="31" v="a" p="AT" sid="3"/>
    <t id ="32" v="big" p="JJ" sid="3"/>
    <t id ="33" v="pan" p="NN" sid="3"/>
    <t id ="34" v="and" p="CC" sid="3"/>
    <t id ="35" v="fry" p="VB" sid="3"/>
```

```
  <t id ="35" v="fry" p="VB" sid="3"/>
  <t id ="36" v="chicken" p="NN" sid="3"/>
  <t id ="37" v="on" p="IN" sid="3"/>
  <t id ="38" v="high" p="JJ" sid="3"/>
  <t id ="39" v="flame" p="NN" sid="3"/>
  <t id ="40" v="keep" p="VB" sid="3"/>
  <t id ="41" v="on" p="IN" sid="3"/>
  <t id ="42" v="changing" p="VBG" sid="3"/>
  <t id ="43" v="direction" p="NN" sid="3"/>
  <t id ="44" v="of" p="IN" sid="3"/>
  <t id ="45" v="chicken" p="NN" sid="3"/>
  <t id ="46" v="so" p="QL" sid="3"/>
  <t id ="47" v="it" p="PPS" sid="3"/>
  <t id ="48" v="should" p="MD" sid="3"/>
  <t id ="49" v="cook" p="VB" sid="3"/>
  <t id ="50" v="equally" p="RB" sid="3"/>
  <t id ="51" v="from" p="IN" sid="3"/>
  <t id ="52" v="all" p="ABN" sid="3"/>
  <t id ="53" v="side" p="NN" sid="3"/>
  <t id ="54" v="." p="." sid="3"/>
</TOKENS>
<INGREDIENTS>
  <i name ="chicken">
    <TOKENS>
      <t id="3"/>
      <t id="11"/>
      <t id="36"/>
      <t id="45"/>
    </TOKENS>
    <ACTIONS ConceptualClass="C9" >
      <t id="1"/>
      <t id="5"/>
      <t id="10"/>
      <t id="35"/>
    </ACTIONS>
  </i>
</INGREDIENTS>
```

```
  <ACTIONS>
    <a id="1">
      <t id="3"/>
    </a>
    <a id="5">
      <t id="3"/>
    </a>
    <a id="10">
      <t id="11"/>
      <t id="12"/>
      <t id="13"/>
      <t id="15"/>
      <t id="17"/>
      <t id="18"/>
      <t id="20"/>
      <t id="21"/>
    </a>
    <a id="28">
      <t id="29"/>
    </a>
    <a id="35">
      <t id="36"/>
    </a>
  </ACTIONS>

</RECIPE>
```

Figure A.1: Representation of Recipe

A.2 Receipe Regeneration from Annotations

From above representation, the recipe can be generated from $< ACTIONS >$ section. This is useful for recipe regeneration when actions are deleted from the recipe. From this representation, we find out using in $< ACTIONS >$ section that first action is "Clean" that has id=1 which is being applied to "chicken" that has id=3. From this, we can generate the recipe statement as

"Clean chicken."

As soon as, we apply the same approach to second action, it comes out that the action is "dry" and its being applied to the same token as action with id=3. Therefore we can combine the two statements to make it look more natural i.e.

"Clean and dry chicken."

The third action is "Marinate" having id=10. This is the divalent verb. Therefore, you will also see in its token list, the index for the auxiliary verb that is used to separate two arguments. Here in this case it is "with" having id=12. Since there are multiple ingredient in the statement, a list will have to be formed. The resultant statement will be as follow.

"Marinate chicken with salt, paper, lemon, juice, corn and flour."

The problem here is that we have not handled compound nouns here. Therefore lemon juice is generated as lemon and juice separately. Similar is the case with corn flour. Similarly, the last two statement produce following statements.

"Heat oil" and "fry chicken"

The overall generated recipe is

> "Clean and dry chicken. Marinate chicken with salt, paper, lemon, juice, corn and flour. Heat oil. Fry chicken."

This is very summarized version of the original recipe which is shown below

> "Clean the chicken and dry it with kitchen towel. Marinate chicken with salt, pepper, lemon juice and corn flour (preferably over night).Heat oil in a big pan and fry chicken on high flame keep on changing direction of chicken so it should cook equally from all sides."

Appendix B

Description of Program & Data

All the programs and data that have developed as part of master thesis on "Domain Specific information extraction for semantic annotation" have been provided in accompanied CD with this thesis.

There are two folder in the CD. "InformationExtraction" folder contains the source code, data and all the relevant files for the information extraction techniques. The "LatticeGeneration" folder contains the source code, data and all the relevant files for ontology generation from extracted information.

There are two well known text analysis techniques have been implemented in this work. One is "Rule Based approach" and other is "Dependency Based Syntax Analysis". The "InformationExtraction" folder contains the source code and data for both approaches. There are following files in this "InformationExtraction".

MainAnnotator.java	Contains Main Method to execute this java program.
Eval.java	Implements code to measure Precision and Recall.
Annotator/Parser/StanfordParser.java	Implements Dependency based information extraction code.
Annotator/Parser/RuleBaseExtracter.java	Implements Rule based information extraction code.
Annotator/Parser/DataReaderWriter.java	Implements methods for input/output from files.
Annotator/Parser/Util.java	Utility methods.

To run the program, you need to take care of following things.

1. StandfordParser.Java requires stanford parser. You can download parser from http://nlp.stanford.edu/software/lex-parser.shtml. Stanford-Parser.jar should be in the java ClassPath to run the application.

2. This program also utalizes the morphological engine provided as a part of stanford POS tagger. It can be downloaded from http://nlp.stanford.edu/software/tagger.shtml. The stanford-postagger.jar should be in class path to run the application.

3. RuleBaseExtracter.java uses the customed trained pos tagger. We have used brill's tagger provided as part of python Natural Language Tool Kit(NLTK) to train and tag the recipe text. The customed trained Brill's tagger has been provided as part of this program. It is in the "tagger" directory. To run the program,in the following piece of code, change the path of the python application.

   $Processproc = Runtime.getRuntime().exec("C:/Python25/pythontagger/tagger.py");$

Apart from the files mentioned above, the following Directory structure and files should be preserved apart from the files mentioned above to run the application.

edu	folder that contains unjarred version of stanford applications.
Data/englishPCFG.ser.gz	Stanford Parser model.
Data/IngDic.txt	Ingredients Dictionary.
Data/ActDic.txt	Actions Dictionary.
Data/PrepDic.txt	Preposition Dictionary to fix pos tagging error on preposition.
Data/DetDic.txt	Determiner Dictionary to fix pos tagging error on determiner.
Data/ConjDic.txt	Conjunctions Dictionary to fix pos tagging error on conjunction.
Data/Recipies/Original	Contains original recipe text.
Data/Recipies/DepExtractedMapping	Contains information extracted using dependency apporach on original recipes.
Data/Recipies/RuleExtractedMapping	Contains information extracted using rule base approach on original recipes.
Data/Recipies/ManuallyExtractedMapping	Contains manually extracted information for measuring performance.
tagger/RecipesBrillTagger.pkl	Customed Train Brill's Tagger Python Object saved in file.
tagger/tagger.py	Tagger file used for tagging recipe text. This program is called from java program.
tagger/token.txt	tokens to tag.
tagger/tag.txt	tag output program.

To run the program on any platform LINUX or WINDOWS.

1) Compile : Javac MainAnnotator.java

2) Run : Java MainAnnotator

The "LatticeGeneration" folder contains following files and folder

GenerateLattice.py	Paython script ot generate Galicia Lattice file.
Data	This folder contains extracted data.

Run the GenerateLattice.py file as normal python program.