CHAPTER 5

SEMANTIC ANALYSIS

CHAPTER OVERVIEW

This chapter deals with the meaning of written text. The general idea of semantic interpretation is to take natural language sentences or utterances and map them onto some representation of meaning. The chapter begins with an introduction to semantic analysis. Next, general characteristics of meaning representations and meaning structures are described. A general approach to semantic analysis based on semantic compositionality is then discussed. Lexical semantics is the next topic of discussion in this chapter. Finally, word sense disambiguation is discussed. This includes a discussion of selectional restriction-based and context-based disambiguation approaches, and word sense disambiguation application and its evaluation.

5.1 INTRODUCTION

The goal of computational linguistics has long been to produce a semantic analysis. Semantics is associated with the meaning of language. The general idea of semantic interpretation is to take natural language sentences or utterances and map them onto some representation of meaning. Semantic analysis is concerned with creating representations for the meaning of linguistic inputs. This chapter deals with the meaning of written text. We can divide semantics into two parts as follows:

- The study of the meaning of individual words (*lexical semantics*)
- The study of how individual words combine to give meaning to a sentence (or larger units)

Once we have the meaning of the words, we need to combine them into the meaning of the whole sentence. The principle of semantic compositionality (sometimes called Freg's principle) states that the meaning of the whole is comprised of the meanings of its parts (Keenan and Faltz 1985), i.e., the meaning of a sentence can be composed from the meaning of its constituent words. Natural languages do not always obey this principle of compositionality. Often, the meaning of the whole is only partially

dependent on the meaning of its constituents (as in collocates) and sometimes entirely different from the meanings of individuals (e.g., in idioms).

Hence, this decomposition of semantics may not be realistic. Word meaning is just one component of semantics. The relationship that exists between words, the domain, the word order, the syntactic structure, the underlying context, and the real world knowledge, all contribute to the meaning of a sentence. Still, compositional and lexical (word) semantics remains the dominant approach to semantics in computational linguistics. There are theories supporting the idea that linguistic knowledge is knowledge about words (Joshi 1985). This means that lexicon contains all the knowledge about language. These theories completely dispense with grammar as an independent entity, which has been accepted for so many years. The idea that syntax and lexicon can not be described adequately without reference to each other is gaining increasing support from the results of corpus analysis, which attempts to investigate role of context in syntax and semantics.

Lexical semantics has been the starting point for all the early theories of semantics. The most common paradigm involves decomposing lexical meanings in terms of semantic primitives or atomic units of meaning. However, this theory turns out to be inadequate in handling compositional semantics. This has led to the development of model theoretic semantics, which, along with the structural semantics, is the dominant approach to semantics within linguistics. Model theoretic semantics is inspired by the semantics that logicians use for formal logical languages. Logicians focus on 'logical words' (and, or, not, if, all, some, only, etc.) and do not pay attention to non-logical words (most nouns, verbs and adjectives). The primary advantage of this theory is its ability to explain compositional semantics—how the meaning of a sentence is determined from the meanings of its parts. One important aspect of meaning is that it relates sentences to the outside world. However, we do not know what the world is. A model theoretic approach to semantics attempts to create a model of the world and determines the truth of a sentence using this model. In this theory, the truth of a sentence does not mean that the sentence is actually true; it simply means the sentence is true in the world being modelled. Model theoretic semantics is effective in studying pragmatics as well as semantics.

We organize the discussion in this chapter around two main topics: the meaning representation of the whole sentence and the meaning of words (i.e., lexical semantics). Section 5.2 deals with sentence level meaning representation. In particular, we discuss the general characteristics of

meaning representation languages (Section 5.2.1), the meaning structure of the language (Section 5.2.2), and computational approaches to semantic analysis (syntax-driven semantic analysis and semantic grammars in Section 5.2.3). Next, we discuss the internal structure of words, their relationships, and their meanings in Section 5.3. The meaning of the words themselves cannot be derived by separating them from their context; especially the correct sense of a word cannot be identified unless we examine the words in context. This task of word sense disambiguation is of considerable theoretical and practical interest. We give a special mention to the various types of ambiguities and approaches to word sense disambiguation in this chapter (Sections 5.4 and 5.5).

5.2 MEANING REPRESENTATION

Many language tasks require non-linguistic knowledge of the world besides phonological, morphological, and syntactic knowledge. The semantic representation bridges the gap from linguistic inputs to the non-linguistic knowledge involving the meaning of linguistic inputs. The language used to describe the syntax and semantics of these representations are called meaning representation languages. The process of creating such representation and assigning it to linguistic inputs is called semantic analysis. Commonly used meaning representation languages are first order predicate calculus (FOPC) and semantic networks and conceptual dependency. We hope you are already familiar with these knowledge representation formalisms; a review of them is provided in the Appendix. Conceptual graphs were introduced to capture semantics of natural language, and have also been used for knowledge representation formalism. We discuss this formalism and its use in information retrieval in Chapter 10. We now focus on the general characteristics of meaning representation languages, the meaning structure of languages, and how meaning representations are created and assigned to sentences.

5.2.1 Characteristics of Meaning Representation Languages

There are certain characteristics that a meaning representation language must possess.

Verifiability

Meaning representations must be verifiable, i.e., we must be able to determine the truth of our representation. We determine the truth by comparing the meaning representation of an input with the repository of facts existing in the domain (i.e., representations in knowledge base). It is

important, therefore, that our meaning representation language should facilitate such a comparison. Consider the following question:

Does Kingfisher serve Hyderabad? (5.1)

The underlying proposition of this question is that Kingfisher serves Hyderabad. We can represent this as

Serves (Kingfisher, Hyderabad)

This representation will be matched against a knowledge base of facts. If the system finds a matching proposition, it returns an affirmative reply. It answers no if its knowledge of domain is complete and do not know if it has reason to believe that its knowledge base lacks information. This example illustrates the meaning of verifiability. To summarize, verifiability is a system's ability to compare the input representation to the situation that exists in the world as modelled in the knowledge base.

Unambiguous

The second requirement of a meaning representation language is that it should be unambiguous, i.e., it should support representations that have only one possible interpretation. Like other domains, the domain of semantics is also full of ambiguities–these are discussed in Sections 5.4 and 5.5. As our reasoning and subsequent action is based upon the semantic content of inputs, it is important that the final meaning representation of an input be unambiguous, regardless of ambiguity in the raw input.

A concept related to ambiguity is vagueness. Vagueness refers to lack of precision. Unlike ambiguity, it does not lead to multiple representations but makes it difficult to determine what to do with meaning representations. Consider the following sentence:

I want to go to a hill station. (5.2)

This input provides enough information for a tour organizer to provide reasonable response, but it is quite vague as to where the user really wants to go. For some purposes, this vague representation of inputs may be sufficient, while a more specific representation may be needed for other purposes. A meaning representation language should support a certain level of vagueness.

Canonical Form

Consider the following alternative ways of expressing the content of sentence (5.1):

Does Kingfisher offer a flight to Hyderabad? (5.3a)

Does Kingfisher have a flight to Hyderabad? (5.3b)

These representations use different words and have different syntactic structures from sentence (5.1). This gives rise to different meaning representations. But we know that all three mean the same. Such a situation creates a problem during matching. If a system's knowledge base contains only a single representation of the fact, then all but one sentence will fail to match. One solution is to store all possible alternatives in the knowledge base, but this may lead to inconsistency in the knowledge base. What we want is to have the same meaning representations of inputs that mean the same. This necessitates the use of canonical forms. Canonical forms complicate the task of semantic analysis. To assign the same representation to sentences (5.1), (5.3a), and (5.3b), the system must know that 'having a flight to', 'offering a flight to', and 'serving to' Hyderabad all mean the same thing in this context. Hence, the different syntactic parses underlying these requests must lead to the same meaning representation. There exist systematic meaning relationships among word senses and among grammatical constituents. We use them to assign the same representation to various inputs. Words have different senses. The process of choosing the right sense in context is called word sense disambiguation and is discussed in Section 5.5.

Inference and Variables

Given a set of facts about the world in the knowledge base, and the input representation, we can derive new facts that logically follow the known facts. Inference refers to system's ability to draw valid conclusions based on the meaning representation of inputs and representation of facts in its knowledge base. A meaning representation language should support this type of derivation (inferencing). Another requirement for the meaning representation language is that it should allow the use of variables. Consider the following, more complex, scenario:

I would like to catch a flight to Hyderabad. (5.4)

This sentence does not mention any particular flight; instead it refers to an unknown and unnamed flight that goes to Hyderabad. The simple matching we discussed so far does not work in this case. Answering this request requires a more complex matching that involves the use of variables. We represent it as

Goes (x, Hyderabad)

To satisfy this request, we have to search for a known object in the knowledge base such that, substituting x by it matches the whole proposition.

Expressiveness

Natural languages cover a wide variety of content (or subject matter). A meaning representation language must be able to represent the meaning of this wide range of content. Therefore, to be of any use a meaning representation language must be equipped with expressive power.

5.2.2 Meaning Structure of the Language

The previous section focused on the role of meaning representation languages without discussing the meaning structure of language. This section focuses on fundamental predicate argument structure.

Predicate Argument Structure

It seems that the semantic structure of all human languages have an underlying predicate argument structure which contributes significantly to its semantic structure. This structure states that specific relationships exist among the concepts expressed through the words and phrases of a sentence. It is mainly this structure that makes it possible to create a single composite meaning representation from the meanings of the various parts of an input. One of the important tasks of a grammar is to help organize this predicate argument structure. Therefore, a meaning representation language should support the representation of the predicate argument structures.

The predicate argument structure is mainly manifested by the constraints that various grammatical categories put on other constituents appearing with them in some linguistic unit. For example, a verb prescribes specific constraints on the number, syntactic category, and location of the phrases that are expected to occur with them in syntactic structures.

Example 5.1 Consider the following sentences:

Murthy likes music.
Murthy likes watching movies.
Murthy likes to perform in the theatre.

The syntactic argument frames for these examples are

NP likes NP
NP likes VP
NP likes inf-VP

These syntactic frames tell the number, position, and syntactic category of the arguments that are expected to occur with a verb. For example, the frame for the first sentence in Example 5.1 specifies the following facts:

1. There are two arguments to this predicate.
2. Both arguments are NPs.

The first argument occurs before the verb and plays the role of the subject.

The second argument occurs after verb and plays the role of the direct object.

These syntactic frames provide useful information about syntax and also carry useful semantic information that can be utilized in creating meaning representation. For example, in each of the three examples given earlier, the pre-verbal argument plays the role of the entity doing the act of liking, and the post-verbal argument plays the role of the thing being liked. These generalizations help in associating the surface arguments of a verb with the semantic roles the arguments play in its underlying meaning representation. The semantic role that a noun phrase plays in a sentence is termed *thematic-role* or *theta-role*. The term case-role denotes the same concept. Often, the verbs place some semantic restriction, called selectional restriction or selectional preference, on the type of arguments that can play these semantic roles. These restrictions help in arriving at a correct meaning representation. For instance, the verb eat requires that its pre-verbal argument, i.e., the eater, must be some animate entity, and the post-verbal argument, the thing being eaten, must be some edible item. Verbs are not the only objects that can have a predicate-argument structure. Prepositions and nouns may also be specified using a predicate argument structure. For example, the phrase *a cup on the table* represents a relationship between a *cup* and a *table* expressed through the preposition *on*. The meaning representation of this phrase is given as

On (cup, table)

Any useful meaning representation language must support the representation of predicate argument structure. More specifically, it must support the representation of predicate argument structure with a variable number of predicates, the semantic labelling of arguments to predicates, and the semantic restrictions placed on those arguments. Knowledge representation formalisms, such as first-order predicate calculus, semantic networks, conceptual graphs, etc., all support representation of predicate argument structure.

5.2.3 Syntax-driven Semantic Analysis

We now turn our attention to how meaning representations are created and assigned to linguistic units. The first approach we discuss is syntax-driven semantic analysis.

Syntax-driven semantic analysis is a computational approach to semantic analysis that utilizes static knowledge from the lexicon and the grammar.

This type of meaning representation corresponds to literal meaning, which is context independent and inference free. The basic notion that drives this approach is the principle of compositionality, which states that the meaning of the whole can be composed from the meaning of its parts. What this principle tells us is that we can create meaning representations for sentences from the meanings of their consistent words. Syntax-driven semantic analysis uses syntactic analysis to guide the process of arranging semantic representations.

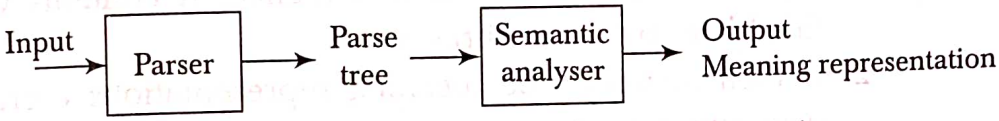

Figure 5.1 A simple approach to syntax-driven semantic analysis

Figure 5.1 shows the schematic diagram of this approach. The output produced by syntactic analysis is passed as input to a semantic analyser to produce a meaning representation. Usually, a semantic analyser produces multiple ambiguous meaning representations as output. This is due to the ambiguities arising from the syntax (as discussed in Chapter 4) and the lexicon. A semantic analyser creates a representation for all possible interpretations, leaving the task of resolving this ambiguity, which requires access to domain-specific and contextual knowledge, to a subsequent stage of processing. We can use part-of-speech tagger, PP-attachment mechanism, and word sense disambiguation mechanism to reduce the number of possible meaning representations produced by the semantic analyser.

We now illustrate how a semantic analyser actually builds the meaning representation of a sentence, with the help of an example. Consider the parse tree for the sentence (5.5) shown in Figure 5.2.

President nominates speaker. (5.5)

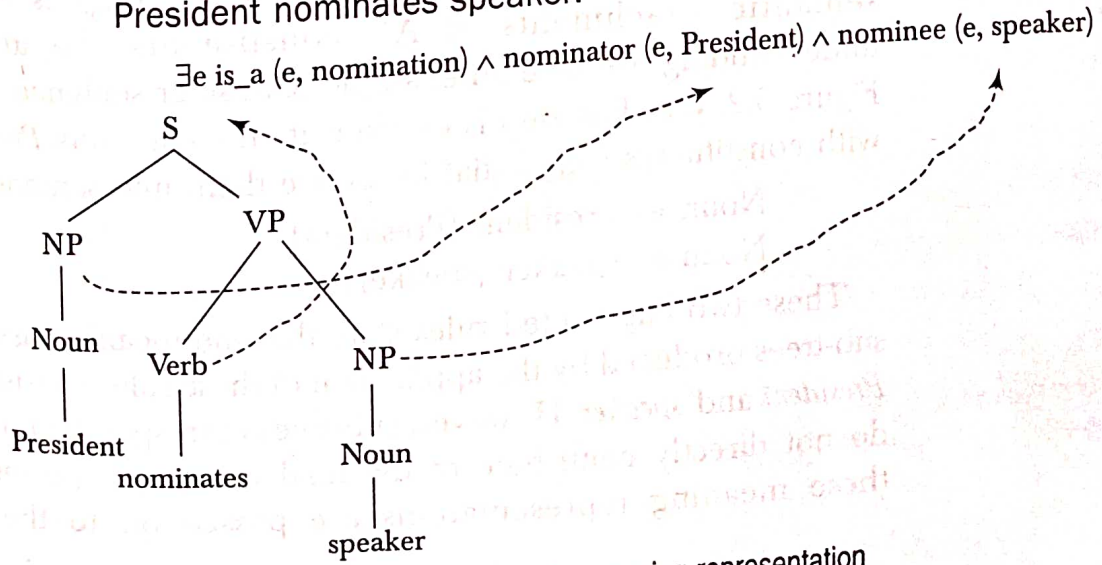

Figure 5.2 Mapping syntactic constituents to meaning representation

Using the parse tree, a semantic analyser produces a semantic representation in following steps.

1. First, it retrieves a meaning representation from the sub-tree corresponding to the verb *nominates*. The program that interprets the parse tree must have the knowledge that it is the verb whose template will define the meaning representation of the whole sentence, and about position of its arguments and their roles. The meaning representation of the verb acts as a template for the meaning representation of the whole sentence. It contains variables that are filled later by noun phrases.
2. It then identifies the meaning representations corresponding to the two noun phrases.
3. And finally, it associates or binds the meaning representations of noun phrases to the variables appearing in the meaning representation of the verb, to give the meaning representation for the sentence as a whole (as shown in Figure 5.2).

As any reasonable grammar will have infinite number of such trees, mapping every possible tree to its semantic representation is not a viable approach. We need to identify general mappings, which is achieved by augmenting the lexicon and the grammar rule with semantic attachment, and devising a mapping between rules of the grammar and the rules of semantic representation. This is known as rule-to-rule hypothesis. The semantic attachments are instructions on mapping components of a rule to a semantic representation. An augmented rule takes the following form:

$$A \rightarrow \alpha_1\alpha_2\alpha_3 \ldots \alpha_n \{f(\alpha_{i \cdot \text{sem}}, \ldots, \alpha_{k \cdot \text{sem}})\}$$

The text appearing in {...} specifies that the meaning representation assigned to construction A (represented as $A_{\cdot sem}$) is a function of the semantic attachments of A's constituents. To achieve a better understanding, we take an example. Consider sentence (5.5) as shown in Figure 5.2. Our first step is to associate the constants *President* and *speaker* with constituents (rules) that introduce them into sentence:

Noun ← President {President}
Noun ← speaker {speaker}

These two augmented rules state that the meaning associated with the sub-trees produced by the application of these rules consist of the constants *President* and *speaker*. However, sub-trees corresponding to these constants do not directly contribute to the final meaning representation. Instead, these meaning representations are passed on to their parents who

contribute to the final meaning representation as indicated by the dotted arrows in the Figure 5.2. The augmented rule for noun phrase is

$$NP \rightarrow Noun \{Noun_{.sem}\}$$

This rule states that the meaning representation of the noun phrase is the same as the meaning representation of its constituents. Now, we need to specify the semantics of the event underlying the sentence, i.e., the verb *nominates*. The *nomination* event involves a *nominator* and a *nominee*. The semantics for *nominate* can be represented by the logical formula:

$$\exists e, x, y \text{ is\_a } (e, \text{nomination}) \wedge \text{nominator } (e, x) \wedge \text{nominee } (e, y) \quad (5.6)$$

This logical formula is used as the semantic attachment of *nominate*, resulting in the following augmented rule.

$$\text{verb} \rightarrow \text{nominates } \{\exists e, x, y \text{ is\_a (nomination)} \wedge \text{nominator } (e, x) \wedge \text{nominee } (e, y)\}$$

The next constituent to be considered is VP, which dominates both *nominates* and the *speaker*. The grammar rule used is

$$VP \rightarrow \text{verb NP } \{\text{verb}_{.sem} (NP_{.sem})\}$$

However, to create the semantic representation of VP, we cannot simply copy the meaning representations of its children. The goal for VP semantics of '*nominate*' is represented by the logical formula 5.6. To achieve this goal, we need to incorporate the meaning of NP into the meaning of the verb and assign it to the VP. This requires that the variable y be replaced with the logical term *speaker*, as the second argument of the nominee role of *nomination* event. Hence, $VP_{.sem}$ must tell us two things:

1. Which variable in the verb's semantic attachment is to be replaced by which argument.
2. How the replacement is to be performed.

But the semantic attachment of verb does not tell us how to replace each of its quantified variables. So, we need to revise the verb's semantic attachment.

We can use lambda calculus as the 'glue-language' to combine semantic representations systematically. Lambda calculus is an extension of FOPC. A lambda expression consists of a λ-operator, a list of variables (parameters), and an FOPC expression in those variables.

$$\lambda x P(x)$$

The parameter list in the lambda expression makes available the variable within the body of logical expressions, for binding to external arguments provided by the semantics of other constituents. This process of binding

is known as *lambda reduction*. We illustrate this with the help of an example.

Example 5.2 Consider the application of $\lambda xP(x)$ to the constant *Taj*.

$$\lambda xP(x)\ (Taj)$$

The result of performing a λ-reduction on this expression is

$$P(Taj)$$

The reduction process replaces the variable in the body of the expression with *Taj* and removes λ. Thus, lambda calculus is able to satisfy the two requirements of VP semantics as identified earlier. The formal parameter list tells us variables within the body that are available for replacement and λ-reduction tells how to perform replacement. Changing the semantic attachment of the verb to a λ-expression, we get

$$\text{Verb} \rightarrow \text{nominates } \{\lambda y \exists\ e, x \text{ is\_a } (e, \text{nomination}) \wedge \text{nominator } (e, x) \wedge \text{nominee } (e, y)\}$$

Now y is externally available, and can be bound by the application of this expression to a logical term.

Let us go back to the VP semantics in our example sentence (5.5).

$$\text{VP} \rightarrow \text{verb NP } \{\text{verb}_{\cdot\text{sem}}\ (\text{NP}_{\cdot\text{sem}})\}$$

The value of $\text{NP}_{\cdot\text{sem}}$ is *speaker*. Application of the $\text{verb}_{\cdot\text{sem}}$ to the $\text{NP}_{\cdot\text{sem}}$ results in the substitution of *speaker* for each occurrence of variable y in the expression. The meaning of a verb phrase *nominates speaker* is thus:

$$\exists\ e, x \text{ is a } (e, \text{nomination}) \wedge \text{nominator } (e, x) \wedge \text{nomine } (e, \text{speaker})\}$$

We now create the semantic attachment for the S rule.

$$\text{S} \rightarrow \text{NP VP} \qquad \{\text{VP}_{\cdot\text{sem}}\ (\text{NP}_{\cdot\text{sem}})\}$$

This rule says that the semantic representation of S ($\text{S}_{\cdot\text{sem}}$) can be constructed by applying $\text{VP}_{\cdot\text{sem}}$ to $\text{NP}_{\cdot\text{sem}}$. As in the VP rule, a simple copying does not work. This rule requires the nominator role in the event representation to be bound to $\text{NP}_{\cdot\text{sem}}$. However, the value of $\text{VP}_{\cdot\text{sem}}$ does not contain any λ-expression. So, let us revise the semantic attachment of verb as follows:

$$\text{Verb} \rightarrow \text{nominates } \{\lambda y\ \lambda x \exists\ e \text{ is\_a } (e, \text{nomination}) \wedge \text{nominator } (e, x) \wedge \text{nominee } (e, y)\}$$

The attachment is now a nested λ-expression. In a nested λ-expression, $\lambda x\ \lambda y\ P(x,y)$ the first reduction binds the variable x and removes the outer λ. The resulting λ-expression can be applied to another term to arrive at a fully specified formula.

$\lambda x\ \lambda y\ P(x, y)(A)$ results in $\lambda y\ P(A, y)$, which in turn can be applied to another term, say B, as $\lambda y\ P(A, y)\ (B)$, resulting in $P(A, B)$.

With the revised attachment of the verb, the value of the $VP_{.sem}$ is $\lambda x \exists e$ is_a $(e,$ nomination$) \wedge$ nominator $(e, x) \wedge$ nominee $(e,$ speaker$)\}$, which is applied to the $NP_{.sem}$. The value of the $NP_{.sem}$ is President. After λ-reduction we get,

$\exists e$ is_a $(e,$ nomination$) \wedge$ nominator $(e,$ President$) \wedge$
nominee $(e,$ speaker$)\}$

The parse tree for this example, with each node annotated with semantic value, is shown in Figure 5.3.

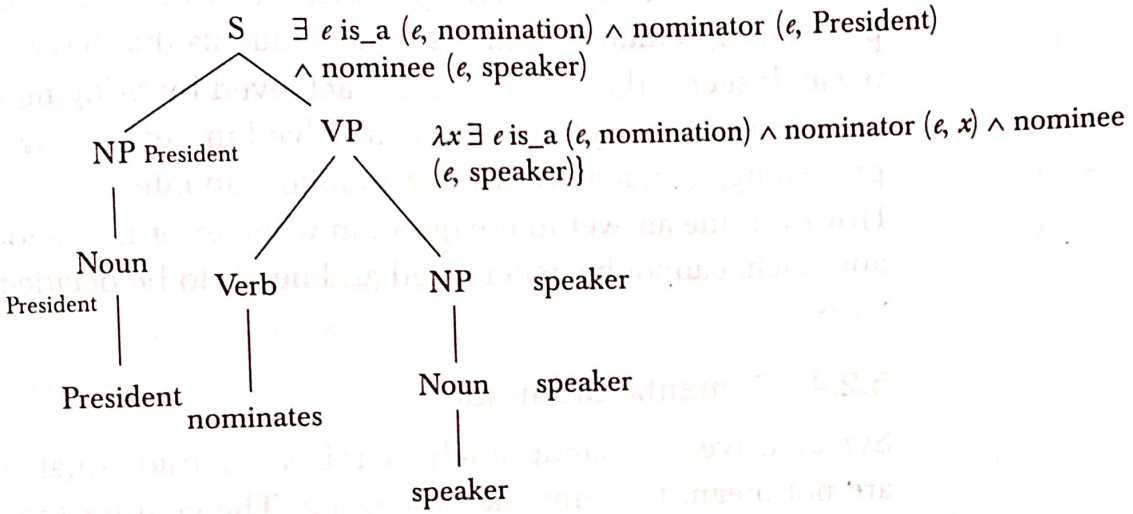

Figure 5.3 Parse tree with semantic values of each node for *sentence (5.5)*

As pointed pout earlier, idioms and collocates represent sentences in which the meaning does not depend on the meaning of the constituents, or depends on it only partially. This poses a challenge to the principle of compositionality. For example, consider the following sentence:

The old man finally kicked the bucket. (5.7)

The phrase *kick the bucket* has nothing to do with a *kick* or a *bucket*. To handle these situations, we need to introduce new grammar rules with semantic attachments that introduce logical terms and predicates, which are not related to any of the constituents of the rule. For example, for the idiom *kick the bucket*, we can introduce following grammar rule.

VP $\rightarrow$ kicked the bucket {died}

The lower case in the right hand side, represents words in the input and the semantic attachment is the meaning of the phrase. This illustrates that the meaning of the idiom is not based on the meaning of any of its parts. A similar problem is presented by collocates which represent sentences in which the meaning depends only partially on the constituents.

There are two approaches to semantic analysis: the pipeline approach and the integrated semantic approach. The schematic diagram we

presented at the beginning of the chapter depicts a pipeline approach. In it, syntactic analysis is performed to give a parse tree. Then we walk through the parse tree applying semantic attachments in a bottom-up fashion.

In integrated semantic analysis, we modify the parse to include operations on semantic attachments. A semantic representation of the input fragments is created as they are being parsed. The advantage of this approach is that it can block a state as soon as an ill-formed semantic fragment is detected. This means, however, that effort is wasted in performing semantic analysis of constituents that have no role in the final parse. It seems that if the benefits achieved by bringing semantics early in the process outweigh the efforts involved in performing spurious semantic processing, then it is worthwhile to follow an integrated semantic approach. However, the answer to the question whether or not to follow an integrated approach, cannot be generalized and needs to be decided on an individual basis.

5.2.4 Semantic Grammars

Syntax-driven semantic analysis relies on traditional grammars, which are not meant for semantic processing. The goal for these grammars is to capture syntactic generalization and avoid over-generation; it is not to facilitate meaning representations. Often, the syntactic structures produced by these grammars do not fit semantic structures very well. Some of the obvious problems are as follows (adapted from Jurafsky and Martin 2000):

- Important semantic elements are often widely distributed across parse trees.
- Parse trees often contain constituents not important to making semantic distinctions.
- The general nature of many syntactic constituents results in semantic attachments that create meaningless representations.

An alternative approach is to bring semantic capabilities into the grammar itself. A grammar developed with the intention of handling semantics is called *semantic grammar*. Semantic grammars were first introduced in the domain of question-answering and intelligent tutoring. They are constructed specifically in terms of semantics information. Rules and constituents correspond directly to entities and activities in the domain. As an example, consider the following request:

I want to go from Delhi to Chennai on 24th December. (5.8)

We can create a rule of the form:

InfoRequest: User wants to go to City from City TimeExpr.

Like the rules for idioms, these rules also mix terminals and non-terminals on their right hand side. In this rule, User, City, and TimeExpr are non-terminals from the domain. The presence of these non-terminals eliminates the need for lambda expression. The immediate constituents of the rule provide the necessary information to build semantic representation.

One of the main motivations behind the use of semantic grammar is dealing with anaphors and ellipsis, which is discussed in the following chapter. The advantage of semantic grammar is that we get exactly the semantic rules we need. However, it also has its drawbacks. The first is lack of generality. As the rules are domain specific, we need to develop a new grammar for each new domain. Second, as the semantic grammar lacks syntactic generalizations, the number of rules needed becomes quite large. For example, a traditional grammar rule for a noun phrase can cover both *Indian hotel* and *Chinese food*, but a semantic grammar must introduce two semantic grammar rules to handle these phrases. We also need to introduce separate rules to handle phrases like *expensive hotel*, *vegetarian hotel*, etc.

5.3 LEXICAL SEMANTICS

So far, we have focused on the creation of meaning representation of whole sentence. The approach we took supports the view that words contribute to the meaning of a sentence but do not themselves have meaning. It is perhaps this view that has led some to consider the lexicon as a simple list of words. However, this is narrow concept, too far from reality. Words have meaning; they have internal structure, and are involved in different relationships with other words. All this information can be captured in a systematic structure by a lexicon. In this section, we focus on this and other issues related to words. More precisely, we focus on lexical semantics, which is concerned with the linguistic study of systematic, meaning-related structure of words or lexemes (the minimal unit in a lexicon).

5.3.1 Relationships

One way to approach lexical semantics is to study the relationship among lexemes (an abstract representation of a 'word', the lexical entry in a dictionary). Semantics of a lexeme can be understood by analysing the

relationships of lexemes with other lexemes. Lexical semantic information is useful for a wide variety of NLP applications. This section discusses a variety of relationship that holds among lexemes and their senses.

Homonymy

The first relationship that we discuss is homonymy which is perhaps the simplest relationship that exists among lexemes. Homonyms are words *that have the same form but have different, unrelated meanings.* A classic example of homonym is bank (river bank or financial institution). A related idea is that of *homophones* which refers to words that are pronounced in the same way but differ in meaning or spelling or both (e.g., be and bee, bear and bare).

Polysemy

Many words have more than one meaning or *sense.* Unlike homonyms, polysemes are words with related meanings. This linguistic phenomenon is called polysemy or lexical ambiguity. Words that have several senses are ambiguous and called polysemous. For example, the word 'chair' can refer to a piece of furniture, a person, the act of presiding over a discussion, etc. The word 'employ' is a polyseme as its two meanings—to hire (employ a person) and to accept (employ an idea)—are related.

In a particular use, only one of these meaning is correct. How we distinguish between these multiple senses is discussed in Section 5.5.

Hypernymy, Hyponymy, and Antonymy

A *hypernym* is a word with a more general sense. The word automobile is a hypernym for a car and a truck. A *hyponym* is a word with a more specific meaning. In the relationship between car and automobile, car is hyponym of automobile. *Antonymy* is a semantic relationship that holds between words that express opposite meanings. The word Dark is an antonym of light, and white is an antonym of black. Words may also be involved in a part-whole relationship called *meronymy*, e.g., 'wall', 'ceiling', 'floor' all are meronyms of 'room'.

Synonymy defines the relationship between different words that have a similar meaning. A simple way to decide whether two words are synonymous is to check for substitutability. Two words are synonyms in a context if they can be substituted for each other without changing the meaning of the sentence.

These relationships are useful in organizing words in lexical databases. One widely known lexical database is WordNet (see Chapter 12).

5.3.2 Internal Structure of Words

Having discussed the relationship between words, we now turn our attention to the structure of words. In our earlier discussions, we pointed out that there is a fundamental predicate argument structure that governs the meaning representation of a sentence. We have also seen that there is certain class of words that defines predicate and predicate structure, and another class of words that provides arguments to predicates. In this section, we discuss how the meanings of words are structured to support this idea.

Thematic Role

Thematic role is the semantic relationship between a predicate (e.g., a verb) and an argument (e.g., the noun phrases) of a sentence. For example, in sentence 5.6, the role of subject of the verb *nominate* is *nominator.* Similarly, an *eating* event will have an *eater*, a *reading* event will have a *reader*, a *singing* event will have a *singer*, and so on. *Eater, singer*, and *reader* have something in common. They are all performers of some event, are animate, and have a direct causal responsibly for their events. The thematic role provides a means to capture the semantic commonality between the actors of the events. In terms of thematic role, the subjects of the verb *eat*, *sing*, and *read* are agents. Agent is a thematic role which refers to the deliberate performers of an event or action. Similarly, the arguments that are affected by the action are assigned the thematic role Theme.

Thematic roles were first introduced by Gruber (1965) and Fillmore (1968), though the theory of semantic roles dates back to the work of Panini in Sanskrit grammar in 500–400 BC (Kearns 2000). The notion of thematic role was called Karaka in Paninian grammar. Panini established classes of NPs according to the broad interpretation of their grammatical form. Panini proposed six Karakas: Karma (action, object that is desired), Karana (means, instrument), Karta (agent, that which acts), Sampradana (transmission, what we aim at with the object), Apadana (removal, what is left when we move away), and Adhikarana (locative). Although thematic roles have widely appealed to linguists, they remain the subjects of disagreements, such as what roles to include and exactly how to define them. Some of the commonly used thematic roles along with their rough definitions are listed in Table 5.1. The thematic role has been used as shallow semantic language and as interlingua in machine translation.

Table 5.1 Commonly used thematic roles

Thematic role	Definition
Agent	Deliberate performer of the event or action, e.g., *I* opened the lock with a key.
Theme	The arguments that are most directly affected by an event, e.g., I opened the *lock* with a key.
Instrument	The instrument used to carry out the action, e.g., I opened the lock with a *key*.
Experiencer	Arguments that undergo a sensory, cognitive, or emotional experience, e.g., *Shweta* hates cricket.
Force	Unconscious performer of the event, e.g., The *storm* blown over many trees.
Location	Places where the event occurs, e.g., Children are playing in the *garden*.
Source	Origin of the object of a transfer event, e.g., Flight will take off in a short while from *Chennai*.
Goal	The destination of a transfer event.

Selectional Restrictions

Selectional restrictions are semantic constraints that are placed on arguments for a given word (sense). As an example, consider the verb *eat*, which requires food items as an object (theme) and an animate as a subject (agent). Selectional restrictions can be observed easily in cases where they are violated as in the following sentence:

A table eats grass. (5.9)

The semantics of selection restrictions can be captured in a meaning representation using thematic roles:

$$\exists e, x, y \text{ eating}(e) \wedge \text{agent}(e, x) \wedge \text{theme}(e, y)$$

This representation means that there is an eating event, and x and y is associated with this via agent and theme relation respectively. To express that y must be edible we add new terms as follows:

$$\exists e, x, y \text{ eating}(e) \wedge \text{agent}(e, x) \wedge \text{theme}(e, y) \wedge \text{is a } (y, \text{Edible Thing})$$

When a phrase like *ate an apple* is encountered, a semantic analyser can use the following representation:

$$\exists e, x, y \text{ eating}(e) \wedge \text{agent}(e, x) \wedge \text{theme}(e, y) \wedge \text{is a } (y, \text{apple})$$

This is a reasonable representation, as category *apple* is consistent with its membership in the category *Edible Thing*. A phrase such as *ate a book* will be semantically ill-formed, as *book* is inconsistent with the membership in *Edible Thing*. However, this approach requires that the knowledge base contains representations of facts about the concepts that make up selectional restrictions.

Yet another approach to specify selectional restrictions is through WordNet hyponym relation. In this approach, WordNet synsets (synonym sets) are used to specify selectional restrictions on the arguments. A

meaning representation is considered well-formed if one of the hyponyms of the word that fills a thematic role, matches the synset used to specify selectional restriction. Let us elaborate on this approach.

Example 5.3 Suppose the synset used to specify selectional restriction on the Theme role of the verb *eat* is {food}. This means that fillers of the theme role can only be words that belong to this synset and its hyponyms.

Now consider the phrase *ate an apple.*

In order to decide whether *apple* is valid filler for theme role, we need to examine the hypernym chain of *apple*, as shown in Figure 5.4. As *food* is in the chain of hypernyms, it is considered an appropriate filler.

Sense 1
apple—(fruit with red, yellow, or green skin, sweet to tart, and crisp whitish flesh)
⇒ edible fruit—(edible reproductive body of a seed plant especially one having sweet flesh)
⇒ produce, green goods, green groceries, garden truck—(fresh fruits and vegetables grown for the market)
⇒ food—(any solid substance that is used as a source of nourishment; "food and drink")
⇒ solid—(a substance that is solid at room temperature and pressure)
⇒ substance, matter—(that which has mass and occupies space; "an atom is the smallest indivisible unit of matter")
⇒ entity—(that which is perceived, known, or inferred to have its own distinct existence (living or nonliving))
⇒ fruit—(the ripened reproductive body of a seed plant)

Figure 5.4 Part of WordNet hypernym hierarchy of word *'apple'*

There can be valid sentences violating selectional restrictions. Perhaps you recall news stories about people demonstrating their skill in eating glass or bulbs, or pulling cars or planes with their teeth instead of cranes. Though, these are atypical examples involving violation of selectional restrictions, they are perfectly well-formed and interpretable. Adhering to selection restriction in these cases will eliminate all possible literal senses, leading the semantic processing to a dead end. One way to avoid this is to consider semantic constraints as preferences instead of restrictions.

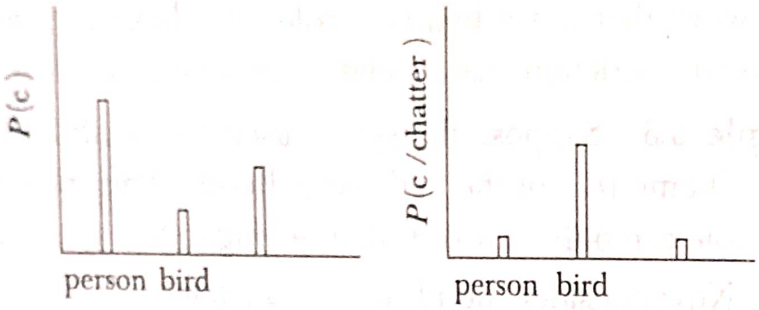

Figure 5.5 Prior and posterior distribution over argument classes

Selectional restrictions have been used in parsing to eliminate a parse that violates these restrictions. It can also be used to infer the meaning of words missing in a dictionary. Traditionally, selectional restriction has been associated with word sense disambiguation. We now discuss a model of selectional preferences proposed by Resnik (1996). The model is general enough to be applicable to any class of words that impose semantic constraints on its arguments. A few applicable instances are: verb-object, verb-subject, noun-noun, adjective-noun, and verb-prepositional phrase. The model is probabilistic in nature and attempts to capture co-occurrence behaviour of predicates and conceptual classes filling the roles of arguments. Figure 5.5 depicts the basic idea. The prior distribution $P(c)$ captures the probability of a class 'c' occurring as the argument in predicate argument relation R, regardless of the identity of the predicate. For example, given the verb-subject relationship, the prior probability for *person* tends to be significantly higher than the prior probability for *bird*. However, once the predicate is known, the probability can change. Take the case of the verb *chatter*. Given this verb, the probability of *bird* as a subject is expected to be significantly higher and the probability of *person* to be lower. In probabilistic terms, it is the difference between conditional (or posterior) and the prior distribution that determines selectional preferences. One way to quantify this difference is to use relative entropy. The model defines a selectional preference strength (SPS) of a predicate p as

$$S_R(\mathrm{p}) = D(P(\mathrm{c/p}) \parallel P(\mathrm{c})) = \sum_{\mathrm{c}} P(\mathrm{c/p}) \log \frac{P(\mathrm{c/p})}{P(\mathrm{c})}$$

where $P(c)$ is probability distribution of class c.

$P(c/p)$ is the probability distribution of class c in the direct object or subject position of p. $S_R(p)$ is a measure of the amount of information that predicate p provides about the conceptual class of its argument. The better $P(c)$ approximates $P(c/p)$, the less influence p has on its argument, and therefore the weaker is its selectional preference.

With this discussion, it follows naturally that the semantic well-formedness of an argument to a predicate can be determined using its relative contribution to SPS. Arguments that fit well are expected to show strong association, i.e., they tend to have higher posterior probability compared to prior probability, as is the case of *bird* in Figure 5.5. Formally, selectional association between a predicate p and a class c is defined as

$$A(\mathrm{p}, \mathrm{c}) = \frac{P(\mathrm{c/p}) \log \dfrac{P(\mathrm{c/p})}{P(\mathrm{c})}}{S_R(\mathrm{p})}$$

While assigning association strength to a word instead of word class, we can simply define it as $A(\mathrm{p}, \mathrm{c})$, or if it is a member of several classes, we can define its association strength as the highest association strength of any of its classes.

The probability that a direct object in class c occurs given the predicate p can be calculated as

$$P(\mathrm{c/p}) = \frac{P(\mathrm{p}, \mathrm{c})}{P(\mathrm{c})}$$

$P(\mathrm{p})$ can be estimated by $C(\mathrm{p})/\sum_{\mathrm{p}'} C(\mathrm{p}')$, its relative frequency with respect to other predicates. Resnik (1996) proposed the following estimate for $P(\mathrm{p}, \mathrm{c})$

$$P(\mathrm{p}, \mathrm{c}) \approx \frac{1}{N} \sum_{\mathrm{w} \in \mathrm{c}} \frac{\mathrm{count}_R(\mathrm{p}, \mathrm{w})}{|\mathrm{classes}(\mathrm{w})|}$$

where $\mathrm{count}_R(\mathrm{p}, \mathrm{w})$ is the number of times the word w occurs as argument of p with respect to relation R; class(w) is the number of classes to which the word w belongs; and N is the total number of predicate-object pairs in the corpus.

This model has been used to predict valid arguments as well as to assign the correct sense to a word in its context.

5.4 AMBIGUITY

Most of us find it difficult to understand why computers are not able to understand language the way people are. This is because we do not realize how vague and ambiguous natural languages are. Ambiguity, i.e., having more than one meaning, can be the result of syntax or semantics. Vagueness is not the same as ambiguity. In vagueness, words or phrases

5.4.3 Pragmatic Ambiguity

Pragmatic ambiguity refers to a situation where the context of a phrase gives it multiple interpretation (Kamsties 2001) as in, *Give it to the kids.* Here 'it' may refer to many things depending on the context. Consider a larger context.

Cake is on the table. I have prepared some snacks. Give it to kids.

It is not clear whether 'it' refers to cake or snacks or both. Perhaps a larger context may help us.

Cake is on the table. I have prepared some snacks. Give it to kids. Kids enjoyed cake and snacks.

Now, it is clear that 'it' refers to both the snacks and the cake. Resolving these types of ambiguities require discourse processing, which is discussed in the next chapter. In this chapter, we focus mainly on lexical semantic ambiguity. We also discuss how sectional restriction can be used to handle structural ambiguities.

5.5 WORD SENSE DISAMBIGUATION

Having discussed various types of ambiguities we now focus on identifying the correct sense of words in a particular use. The first attempt at automated sense disambiguation was made in the context of machine translation. In his famous *Memorandum,* Weaver (1949) discusses the need for word sense disambiguation (WSD) in machine translation, and outlines an approach to WSD, which underlies all subsequent work on the topic.

> *If one examines the words in a book, one at a time as through an opaque mask with a hole in it one word wide, then it is obviously impossible to determine, one at a time, the meaning of the words. [...] But if one lengthens the slit in the opaque mask, until one can see not only the central word in question but also say N words on either side, then if N is large enough one can unambiguously decide the meaning of the central word. [...]The practical question is: "What minimum value of N will, at least in a tolerable fraction of cases, lead to the correct choice of meaning for the central word ?"*

The gist of this excerpt is that words can be disambiguated (Weaver 1949) only in context, or, equivalently, context contains useful information to disambiguate a word. A number of WSD approaches are found in literature that use the context of an ambiguous word to disambiguate its meaning. Others use selection restriction information to resolve ambiguity, particularly to handle PP attachment ambiguity. In the following lines,

we first elaborate use of selectional restriction (or preferences) in WSD and then discuss stand alone WSD approaches.

5.5.1 Selectional Restriction-based Word Sense Disambiguation

We pointed out in Section 5.3 that selectional restrictions or preferences can be used in parsing to eliminate flawed meaning representations. This can be viewed as a form of indirect word sense disambiguation. We now explore this idea. Consider the following sentences:

The institute will *employ* new employees. ('to hire') (5.15a)

The committee *employed* her proposal. ('to accept') (5.15b)

One can intuitively differentiate the senses of *employ* in sentences (5.15a) and (5.15b) with the complements of each *employ*. To be more precise, *employ* in (5.15a) restricts its subject and object nouns to those associated with the semantic features human/organization and human, respectively. On the other hand, *employ* in (5.15b) restricts its subject and object nouns to those associated with the semantic features human/organization and idea, respectively. Consequently, given employees as the object, the sense to hire is selected as the interpretation of employ in (5.15a), and the sense *to accept* is ruled out. The same reasoning can be used to select the sense *to accept* as the interpretation of *employ* in (5.15b).

In fact, the disambiguation process can be seen as mutually propagating semantic constraints to each polysemous word through selectional restriction.

PP attachment ambiguity can also be resolved using selectional restriction. For example, in sentence (5.14), *finding* is an activity that some animate being, not a tree, can perform. This selectional restriction information will help eliminate the parse that attempts to assign *tree* as subject of *found*.

Similarly, the PP attachment ambiguity in the sentence *The man saw the bird with the telescope* will be handled correctly using selectional restrictions. However, this approach does not always work as in sentence, *The man saw the girl with the telescope*. Though, selectional restriction helps us identify that seeing is an activity that can be performed using a telescope, it is not unusual to see with naked eye a girl having a telescope.

The selectional restriction-based disambiguation can be integrated into a semantic analyser. We can follow a *rule-to-rule strategy* as discussed in Section 5.2. With this approach, fragments of meaning representations are created and checked for selectional restriction violation as soon as their syntactic constituents are generated. A parse that violates selectional restrictions are blocked from further processing. The selectional restriction

about arguments of predicates can be encoded in terms of WordNet synsets.

The selectional preference model proposed by Resnik (discussed in Section 5.3) has also been used for WSD. Recall that we have already discussed how to compute selectional association between a predicate and a class. To apply these notions in disambiguation, we simply select the sense that has highest selectional association between a hypernym and the predicate. As Resnik's algorithm makes use of WordNet in disambiguation, we discuss it along with other knowledge-based algorithms. The approach, however, is applicable only in cases where the predicate is unambiguous and selects the correct sense of the argument. We need some other method to handle cases when both the predicate and argument are ambiguous.

5.5.2 Context-based Word Sense Disambiguation Approaches

Approaches to stand-alone WSD that make use of context of ambiguous word basically fall into one of the following two general categories:

- Knowledge-based
- Corpus-based

However, this strict classification does not hold for a number of algorithms, which attempt to combine the features of both. The algorithm proposed by Yarowsky (1995) and Luk (1995) are two that take a hybrid approach.

Knowledge-based (dictionary-based) approaches utilize information from an explicit lexicon or knowledge base to disambiguate a word. The lexicon may be a machine-readable dictionary, a thesaurus, or ontology. Hand coded knowledge may also be used. The work has been carried out using existing lexical knowledge sources such as WordNet (Aggire and Rigau 1996; Resnik 1995; Voorhees 1995), *LDOCE* (Guthrei et al. 1991) and *Roget's International Thesaurus* (Yarowsky 1992).

A *Corpus-based approach* extracts information on word senses from a large sense-tagged corpus. The information used to annotate an ambiguous word is distributional information, context, and further knowledge that has been annotated in the corpus or added during pre-processing. Distributional information about an ambiguous word refers to the frequency distribution of its senses. Context refers to words found to the right and/or left of a word. Additional knowledge sources that can be used include lemmas, part-of-speech and syntactic annotations, etc.

The data acquisition bottleneck (Gale et al. 1992a) is the major difficulty of a corpus-based approach. In order to be useful for WSD, a corpus has to be annotated with word senses. Creating sense tagged corpus is a

labour intensive task. Obtaining sense tagged data in languages other than English is very difficult. Manually sense tagging corpora with the help of a dictionary sense listing or WordNet hierarchy has been utilized. However, it is time consuming. An alternative is to use bootstrapping or unsupervised approaches which are less demanding in terms of tagged data, although tagged data is still needed for evaluation.

There are two possible approaches to *corpus-based* WSD systems:

1. Supervised WSD
2. Unsupervised WSD

Supervised approaches rely on a sense-tagged training corpus for disambiguation. It is an application of the machine learning approach for creating a classifier. It corresponds to the classification task. During training, information about context words and distributional information about the different senses of ambiguous words are collected from the corpus. During testing, the most likely or most similar sense, as computed on the basis of training data, is chosen. The existence of a sense-tagged corpus is a prerequisite for these algorithms.

Unsupervised approaches make use of raw or unlabelled text corpora for training and annotated data for evaluation. A completely unsupervised algorithm cannot do sense tagging. It only does sense discrimination, i.e., it just discriminates among various senses instead of tagging them. Unsupervised WSD algorithms can be considered as a clustering task. We cluster the context of an ambiguous word into a number of groups and discriminate between them without labelling them (Schutz 1998). As in part-of-speech tagging, the performance of unsupervised algorithms is usually on the lower side as compared to supervised algorithms.

Knowledge-based Approaches

A knowledge-based approach utilizes a machine readable dictionary, thesaurus, or ontology to assign the correct sense to an ambiguous word. We begin the discussion with Lesk's work which pioneered the use of dictionary definitions in word sense disambiguation.

Lesk's algorithm: Lesk's method consists in determining the overlap between words in the sense definitions of ambiguous words and the definitions of context words surrounding these ambiguous words in a given text. The most likely sense is the one having relatively large overlap between its definition and the context. Yarowsky (1992) took this idea further by including additional evidences from corpora and training machine learning algorithms.

Suppose that an ambiguous word w has k senses s_1, s_2, s_3,…,s_k, and ds_1, ds_2, ds_3,…,ds_k are the definitions of these senses represented as a bag of words occurring in the definition. The context to be disambiguated is c. dw_j is the dictionary definition of a word w_j occurring in context c, where sense distinctions are ignored. if w_j has multiple senses s_{j1}, s_{j2},…,s_{jm}, then $dw_j = \cup_{ji} ds_{ji}$. Lesk's algorithm can be described as shown in Figure 5.8.

for $i = 1$ to k do
    score(s_i) = overlap (d_{sk}, $\cup_{w_j \in c}$ dw_j)
end
choose s' s.t. $s' = \text{argmax}_{sk}$ score(s_k)

Figure 5.8 Lesk's algorithm

The overlap can be simply the number of words in common between the sense definition and the definitions of the words w_j in the context. Some of the alternative measures that can be used are shown in Figure 5.9.

Dice Coefficient	$\frac{2\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$
Jaccard Coefficient	$\frac{2\lvert X \cap Y \rvert}{X \cup Y}$
Cosine	$\frac{2\lvert X \cap Y \rvert}{\sqrt{\lvert X \rvert \times \lvert Y \rvert}}$

Figure 5.9 Similarity measures

To give a more concrete notion of the algorithm, we consider an example.

Example 5.4 Consider the three senses of noun *ash* in WordNet 2.0 along with their definition.

Sense	Definition
s_1 ash	the residue that remains when something is burned
s_2 ash, ash tree	any of various deciduous pinnate-leaved ornamental or timber trees of the genus Fraxinus
s_3 ash	strong elastic wood of any of various ash trees; used for furniture and tool handles and sporting goods such as baseball bats)

We have to disambiguate the following two contexts:

1. The house was burnt to ashes while the owner returned.
2. This table is made of ash wood.

Using the number of words that the contexts have in common with the sense definition, the scores assigned to these context is

Context	Scores		
	s_1	s_2	s_3
1.	1	0	0
2.	0	0	1

The two contexts are disambiguated correctly using these scores.

Lesk reported disambiguation accuracies of 50–70% when the method was applied on a sample word. Lesk's algorithm is simple to implement and requires no training data. Though the disambiguation accuracy is not impressive, his method served as the basis for most dictionary-based disambiguation work that follows. Lesk's method is highly sensitive to words found in the dictionary. As you can see in Example 5.4, disambiguation was based only on one word that was found to co-occur in the definition of the context word and dictionary sense definition of 'ash'. The presence or absence of a single word may change the result drastically as a direct overlap between contexts and individual sense definition, is needed to disambiguate a word. Lesk acknowledged this problem and mentioned that his algorithm may fail to correctly disambiguate a number of ambiguous words if no words co-occurred between the context and the sense definitions of the word. He suggested the use of dictionaries with larger definitions such as the *Oxford English Dictionary*. The suggestion, however, remains untested.

One way to handle this lack of evidence problem observed by Lesk, is to expand the definition by including words related to, but not appearing in, individual sense definitions. For example, in the sentence, *I went to the bank to deposit cash*, the word *deposit* appears in the context of the ambiguous word *bank*, but the definition of *bank* in the *American Heritage Dictionary* does not contain *deposit*. We can expand the sense definition of *bank* by including deposit. But, knowing that deposit can be used for expansion does not solve our problem. We must know in which sense of *bank* this expansion is to be carried out. One can easily guess that this expansion is to be carried out in the definition corresponding to the financial sense. But how can this be done automatically? Thesauruses and dictionaries that include subject codes or semantic categories in their entries can help us. These subject codes roughly correspond to the conceptual categories of words. The underlying assumption in using semantic categorization for disambiguation is that semantic categories of words in a context determine the semantic category of the context as a whole, and thus identify the word sense being used. For example, *Longman's Dictionary of Contemporary*

English (LDOCE) (Procter 1978), includes the subject code EC (economic) for the financial sense of bank. This subject code helps us in knowing that *deposit* is related to the *financial* sense of bank. This expansion improves the chance of overlap with the correct sense. Roget's thesaurus also provides semantic categories (Roget 1946). Walker (1987) used this idea and proposed a simple algorithm by incorporating subject codes. His algorithm is based on the assumption that the subject codes assigned to a word reflects the sense of the word. If a word has more than one subject code then it will have more than one sense.

Let $t(s_k)$ be the subject code of sense s_k of an ambiguous word w occurring in context c. We disambiguate w by counting the number of words in the context having subj(s_k) as one of their subject code. The sense with the highest count is selected. We can see the algorithm in Figure 5.10.

for $i = 1$ to k do

score$(s_k) = \sum_{w_j \text{ in } c} d(\text{subj}(s_k), w_j)$

end

choose s' s.t. $s' = \arg\max_{sk}$ score (s_k)

d(subj(s_k), w_j) = 1 iff subj(s_k) is one of the subject code of w_j and 0 otherwise. The score is number of words in context whose subject code matches with the subject code of sense s_k.

Figure 5.10 Walker's algorithm for sense disambiguation

A general categorization of words into topics is often inappropriate for a particular domain. For example, *mouse* may be listed both as an animal and an electronic device in a thesaurus; but in a computer magazine it will rarely be in use for the thesaurus category 'animal'. A general topic categorization may also have a problem of coverage.

Wilks et al. (1990) attempted to expand dictionary definition with words that commonly co-occur with that definition; the idea being that commonly co-occurring words are semantically related to those in the definition. He derived this information from definitions in the dictionary. Wilks's work was based on the LDOCE. This vocabulary produces a large number of word co-occurrences. To perform word sense disambiguation, the context in which a target word occurs is also expanded to include words that co-occur with those already in the context. This expansion adds semantically related word to definitions, which increases the chances of correct disambiguation.

Yarowsky (1992) proposed an algorithm to adapt topic classification. He derived classes of words by starting with words in common categories in Roget's thesaurus. Other than word relationships Roget's thesaurus also supplies an explicit concept hierarchy consisting of up to eight increasingly refined levels. Each occurrence of a word under different categories of the thesaurus represents a different sense of word, i.e., the categories correspond roughly to word senses (Yarowsky 1992). A set of words in the same category are semantically related.

We now discuss a WSD algorithm proposed by Resnik (1997) which uses the WordNet class hierarchy to disambiguate noun senses. The sentences in the training corpus need to be parsed in order to extract syntactic relations such as subject-verb, verb-object, adjective-noun, head-modifier, and modifier-head. A syntactic relation involves two words: a noun n to be disambiguated and a verb, adjective or another noun which is involved in some relation with n. Let n has k senses $s_1, \ldots, s_k$. Suppose the syntactic relation R holds for n and the verb v. For each of these k senses, Resnik's method computes a class C_i as:

$$C_i = \{c | c \text{ is an ancestor of } s_i\}$$

$$a_i = \max_{c \in C_i} (A_R(\text{v}, c))$$

We approximate this value using frequency counts:

$$a_i \approx \max_{c \in C_i} \left(\sum_{w \in c} \frac{\text{count}_R(\text{v}, \text{w})}{|\text{classes}(\text{w})|} \right)$$

where count_R (v, w) is the number of times the word w occurs in syntactic relation R with v, and class(w) is the number of classes to which the word w belongs. The algorithm selects the sense s_i for which a_i is greatest.

Let us illustrate the algorithm with the help of an example (adapted from Ng and Zelle (1997). Consider the following sentence:

I would like to drink coffee.

Suppose we want to disambiguate the noun *coffee*. The syntactic relation between *drink* and *coffee* is that of verb-object. The noun *coffee* has four senses in WordNet: coffee as a kind of (1) beverage, (2) tree, (3) seed, and (4) colour. The algorithm searches the parsed training corpus to get all occurrences of *drink* involved in a verb-object relation. Let us assume that the corpus contains only three such occurrences—*drink tea, drink milk,* and *drink cocoa*. The goal is to identify that the nouns *tea, milk,* and *cocoa* are most similar to the beverage sense of *coffee* without requiring that they be manually tagged with the correct sense in the training corpus. As

shown in Figure 5.11, each of the four nouns *coffee, tea, milk,* and *cocoa* have multiple senses. For example, the noun *tea* has five senses: as a kind of beverage, meal, leaf, party, or bush. The sense that is most commonly shared by these four nouns is the *beverage* sense, whereas the remaining senses are only pointed to by one individual noun.

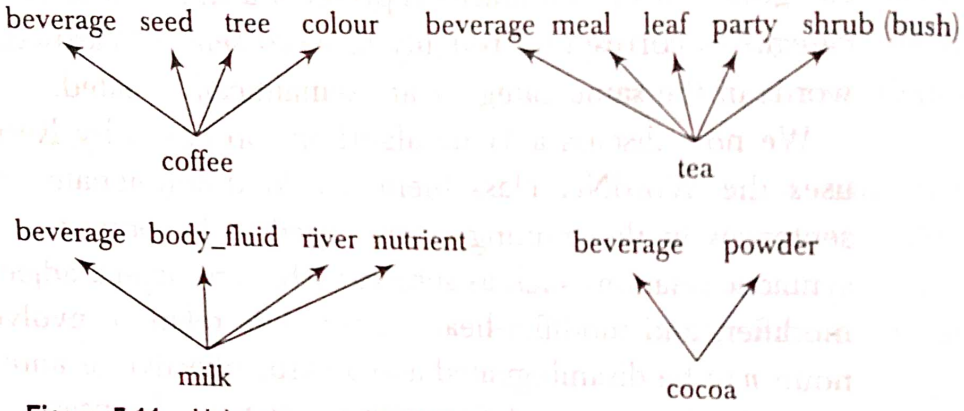

Figure 5.11 Vaious senses of *coffee, tea, milk,* and *cocoa*

In this example, n is the noun *coffee*, with four senses s_1 = beverage, s_2 = tree, s_3 = seed, and s_4 = *colour*. Assume that each of the syntactic relations *drink coffee, drink tea, drink milk,* and *drink cocoa* occur exactly once in our raw corpus. Given the concepts shown in the figure, we calculate the associations $a_1 = 1/4 + 1/5 + 1/4 + 1/2 = 1.2$, $a_2 = a_3 = a_3 = 1/4 = 0.25$. The highest value is a_1, hence, the algorithm selects sense s_1, i.e., the *beverage* sense. See Resnik (1997) for further details.

Supervised Learning of WSD

Most popular approaches to WSD use supervised machine learning methods to train a classifier using a set of labelled instances of the ambiguous word. The labels in the training set are contextually appropriate sense of the ambiguous word w. This classifier is then used to decide the correct sense of unlabelled instances of the ambiguous word. Typically, the learning algorithm requires its training examples, as well as test examples, to be encoded in the form of feature vectors. As discussed earlier, the context of the word provides useful information to disambiguate a word. We map the context into the feature vector. The linguistic features commonly used are collocational and co-occurrence features. Collocational features refer to position specific features, located to the left or right of the target word. Typical features include the word and its part-of-speech. For example, the feature vector consisting of two words to the left and one word to the right of the ambiguous word *silver* in sentence 5.9b is [made VBD a DT silver JJ speech NN]. The co-occurrence features ignore

the position information. In this, words themselves are used as features. Usually, a small number of frequently used words are selected as features and the frequency of these words (features) within a fixed size window with the target word at the centre, are used as feature values. The context is represented as vector of these features.

The most widely known supervised algorithms are *Bayesian classification*, *k-nearest neighbour (k-NN) classification*, and *decision lists*. We now discuss the use of *Naive Bayes classification* approach of word sense disambiguation.

Bayesian Classification The specific algorithm we describe here was introduced by Gale (1992). The classifier assumes that we have a corpus in which each occurrence of an ambiguous word is labelled with its correct sense. The words around the ambiguous word are used to define a context window. The classifier treats the context of word w as a bag of words without structure. No feature selection is done. All the words occurring in the context window contribute in deciding which sense of the ambiguous word is likely to be used with it. What we want to find is the most likely sense s' for an input context c of an ambiguous word w. This is obtained as

$$s' = \arg\max_{s_k} P(s_k/c)$$

As it is difficult to collect statistics for this equation, we apply the Bayesian formula to compute it.

$$s' = \arg\max_{s_k} \frac{P(\mathrm{c}/s_k)}{P(\mathrm{c})} \times P(s_k)$$

Gale's classifier is an instance of Naive Bayes classifier. Naive Bayes classifier is widely used in machine learning due to its efficiency and its ability to combine evidence from a large number of features. In our case, the features are words w_j that occur in the window defining the context of ambiguous word w. The Naive Bayes assumption is that the features are independent of one another. With this assumption, we can estimate the conditional probability of the context as the product of the probabilities of its individual features given that sense. Thus, we get the following approximation:

$$P(\mathrm{c}/s_k) = \prod_{\mathrm{w}_j \,\mathrm{inc}} P(\mathrm{w}_j/s_\mathrm{k})$$

The Naive Bayes assumption has two consequences.

First, the structure and order of the word within the context is ignored. That is, we treat context as a bag of words.