13

Lexical Semantics

13.1 Beyond Formal Semantics

13.1.1 *La langue et la parole*

Formal semantics provides clean grounds and well-mastered devices for bridging language and logic. Although debated, the assumption of such a link is common sense. There is obviously a connection – at least partial – between sentences and logical representations. However, there are more controversial issues. For instance, can the whole language be handled in terms of logical forms? Language practice, psychology, or pragmatics are not taken into account. These areas pertain to cognition: processes of symbolization, conceptualization, or understanding.

Bibliography on nonformal semantics is uncountable. Let us have a glimpse at it with Ferdinand de Saussure (1916), the founder of modern linguistics. Much of Saussure's work, but not exclusively, was devoted to the area of what we would call now real-world semantics. He first made the distinction between the cultural background of a community of people of a same language embodied in words and grammatical structures and physical messages of individuals expressed by the means of a tongue. He called these two layers language and speech – *la langue et la parole*, in his words.

13.1.2 Language and the Structure of the World

Starting from the crucial distinction between *langue* and *parole*, Saussure went on to consider linguistic values of words, taking examples from various languages (1916, Chap. 4). Comparing words to economic units, Saussure described them as structural units tied together into a net of relationships. These units would have no sense isolated, but taken together are the mediators between thought and the way individuals express themselves. Accepting Saussure's theory, languages are not only devices to communicate with others but also to seize and understand reality. This entails that the structure of knowledge and thought is deeply intermingled within a linguistic

structure. And of course, to fit communication, this device has to be shared by a community of people.

In this chapter, we will limit ourselves to some aspects on how a language and, more specifically, words relate to the structure of the world. Words of a specific tongue also embed a specific view of the universe. We believe that most concepts are common to all languages and can be structured in the same way. However, certain words cover concepts somewhat differently according to languages. In addition, the ambiguity they introduce is puzzling since it rarely corresponds from one language to another. We will present techniques to structure a lexicon and to resolve ambiguity. Within this framework, we will examine verb structures and case grammars that will provide us with a way to loop back to sentence representation and to formal semantics.

13.2 Lexical Structures

13.2.1 Some Basic Terms and Concepts

To organize words, we must first have a clear idea of what they express. In dictionaries, this is given by definitions. **Definitions** are statements that explain the meaning of words or phrases. Some words have nearly the same definition and hence nearly the same meaning. They are said to be **synonyms**. In fact, perfect synonyms are rare if they ever exist. We can relax the synonymy definition and restate it as: synonyms are words that have the same meaning in a specific context. Synonymy is then rather considered as a graded similarity of meaning. **Antonyms** are words with opposite meanings.

Contrary to synonymy, a same word – or string of characters – may have several meanings. It is then said to be ambiguous. Word ambiguity is commonly divided between **homonymy** (or **homography**) and **polysemy**:

- When words of a same spelling have completely unrelated meanings, such as for the strings *lot* in *a lot of* and *a parking lot*, they are said to be **homonyms** or **homographs**.
- When a word extends its meaning from concrete to abstract and to concepts tied by analogy, it is said to be **polysemous**. Consider the example of *tools* used in *computer tools* and in *carpenter tools*, where the latter is a concrete object and the former a computer program.

13.2.2 Ontological Organization

There are several ways to organize words within a lexicon. Most dictionaries for European languages sort words alphabetically. An obvious advantage of this method is to provide an easy access to words. However, alphabetical organization is of little help when we want to process semantic properties. A more intuitive way is to organize words according to their meaning. The lexicon structure then corresponds to

broad categories where we arrange and group the words. Such a classification certainly better reflects the structure of our knowledge of the world and is more adequate for semantic processing.

A first classification dates back to ancient Greek philosophy when Aristotle established his famous division of words into ten main categories (Fig. 13.1). Such a lexicon structure, and beyond it, the representation of the world it entails, is often called an **ontology** in computational linguistics.

> Expressions, which are in no way composite, signify substance, quantity, quality, relation, place, time, position, state, action, or affection. To sketch my meaning roughly, examples of substance are "man" or "the horse", of quantity, such terms as "two cubits long" or "three cubits long", of quality, such attributes as "white", "grammatical". "Double", "half", "greater", fall under the category of relation; "in the market place", "in the Lyceum", under that of place; "yesterday", "last year", under that of time. "Lying", "sitting", are terms indicating position, "shod", "armed", state; "to lance", "to cauterize", action; "to be lanced", "to be cauterized", affection.
>
> Aristotle, *Categories*, IV. (trans. E.M. Edghill)

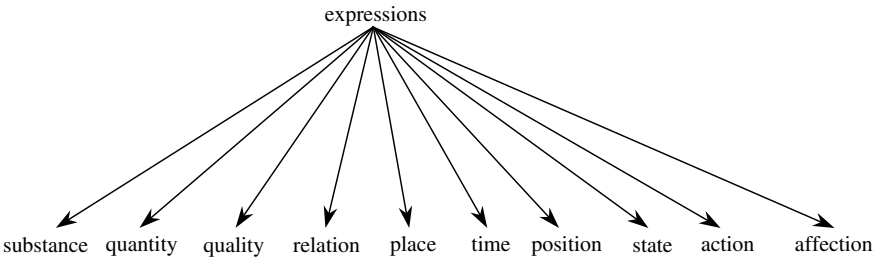

Fig. 13.1. Aristotle's ontology.

We can deepen the classification hierarchy. Aristotle's substance is what we could call now an entity. It includes *man* and *horse* as well as *meal* and *table*. It is easy to introduce further divisions between these words. To refine them, we insert new nodes under the *substance* class. Figure 13.2 shows a symbolic tree distinguishing between *animates*, *human beings*, *animals*, *food*, and *furniture*. This tree representation – now ubiquitous – is traditionally attributed to Porphyry.

13.2.3 Lexical Classes and Relations

An ontological structure defines classes and relationships relative to each word of the lexicon. The most obvious way to group words within an ontological tree is to cut a branch under a word. The branch contains then the **hyponyms** of that word: more specific and specialized terms. For instance, hyponyms of *animals* are *mammals*,

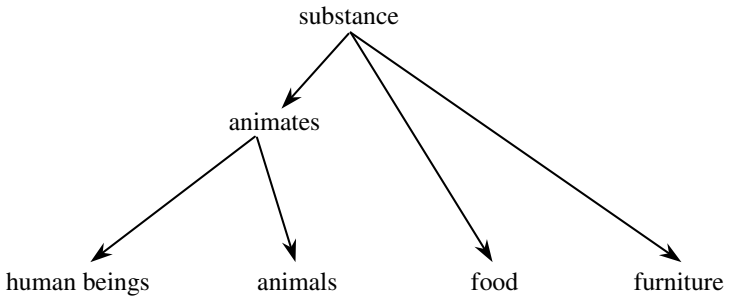

Fig. 13.2. Extending Aristotle's ontology.

carnivores, *felines*, or *cats*. We can go the reverse direction, from specific to more general, and abstract heading to the root of the tree. Thus we get the **hypernyms** of a word. Hypernyms of *hedgehogs* are *insectivores*, *mammals*, *animals*, and *substance*.

It is easy to express hypernymy and hyponymy using Prolog facts. Let us define the `is_a/2` predicate to connect two concepts. We can represent the hierarchy of the previous paragraph as:

```
%% is_a(?Word, ?Hypernym)

is_a(hedgehog, insectivore).
is_a(cat, feline).
is_a(feline, carnivore).
is_a(insectivore, mammal).
is_a(carnivore, mammal).
is_a(mammal, animal).
is_a(animal, animate_being).
```

Hypernymy and hyponymy are reversed relationships and are both transitive. This can trivially be expressed in Prolog:

```
hypernym(X, Y) :- is_a(X, Y).
hypernym(X, Y) :- is_a(X, Z), hypernym(Z, Y).

hyponym(X, Y) :- hypernym(X, Y).
```

Beyond the tree structure, we can enrich relationships and link parts to the whole. *Feet*, *legs*, *hands*, *arms*, *chest*, and *head* are parts of *human beings*. This relation is called **meronymy**. Meronymy is also transitive. That is, if *nose*, *mouth*, *brain* are meronyms of *head*, they are also meronyms of *human beings*. Again it is easy to encode this relation using Prolog facts. Let us use the `has_a/2` predicate:

```
%% has_a(?Word, ?Meronym).

has_a(human_being, foot).
```

```
has_a(human_being, leg).
has_a(human_being, hand).
has_a(human_being, arm).
has_a(human_being, chest).
has_a(human_being, head).
has_a(head, nose).
has_a(head, mouth).
has_a(head, brain).
```

The opposite of meronymy is called **holonymy**.

13.2.4 Semantic Networks

We can generalize the organization of words and knowledge and extend it to any kind of relationships that may link two concepts. Words are figured as a set of nodes, and relationships are labeled arcs that connect them. This representation is called a semantic network (Quillian 1967).

Figure 13.3 shows an extension of Fig. 13.2 where we have added the relations *eat* and *possess*. As we see, the graph contains two *eat* links: the first one between *carnivores* and *meat*, and the second one between *animates* and *food*. Once a semantic net has been designed, we search relations between two concepts, climbing up from specific to general. Inheritance enables then to assign relations `eat(X, meat)` to nodes `X` under *carnivores*, and `eat(Y, food)` to other nodes `Y` under *animates*.

Inheritance makes the design of a semantic network easier, therefore the core structure of the graph remains centered on hypernymy, that is "is a" links. Other properties come as a supplement to it. There are then many ways to augment a net. Design decisions depend on the application. The verbs linking words representing the agent of an action and its object are among common and useful arcs.

13.3 Building a Lexicon

Dictionaries – or lexicons – are repositories of a language's words. They are organized as a set of entries – the words – containing one or more senses. Current dictionaries attempt to itemize all the senses of words and typically contain more than 50,000 entries. Others are focused on specific domains. Dictionaries associate words or senses with grammatical models and definitions. Grammatical models such as the part of speech or a verb's conjugation class indicate morphological and syntactic properties of words; this enables their lemmatization and parsing. Models can also extend to semantic and pragmatic classifications. Many dictionaries cross-reference words using synonyms and give usage examples to show how a word is used in context.

As we could have guessed, wide-coverage lexical databases are central to most natural language processing applications. Instead of creating a new base from scratch,

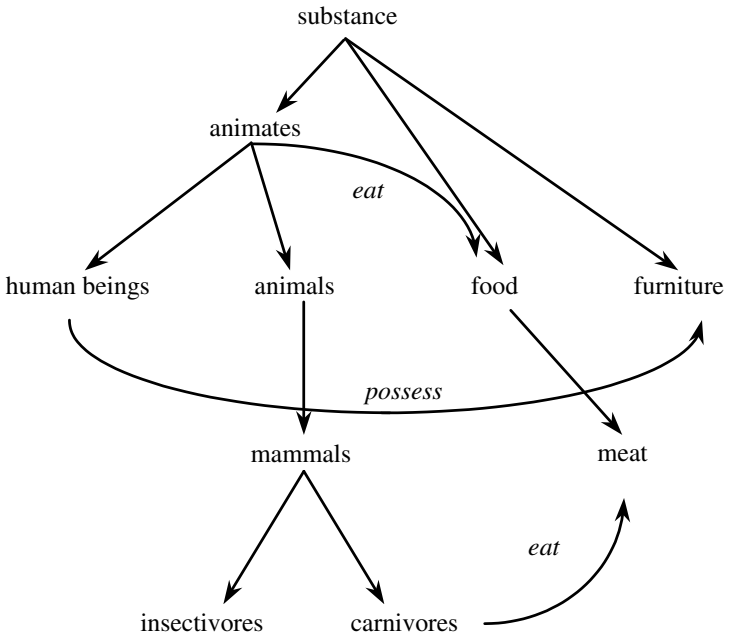

Fig. 13.3. A semantic network.

many computerized dictionaries have been derived from existing paper lexicons and transcribed in a computer readable format, which are called machine-readable dictionaries (MRDs). Computerized dictionaries often take the structure of their paper counterparts and are organized as a set of entries corresponding to word senses with their syntactical model, semantic annotations, and definition.

Learner or nonnative speaker dictionaries are often preferred as primary resources to derive lexical databases. They describe precisely pronunciation and syntactical features, such as a verb's subcategory or an inflection paradigm, while other dictionaries sometimes take it for granted by native speakers. Some dictionaries also tie words to specialized domains with labels such as: anatomy, computer science, linguistics, etc. or to general semantic codes: life, body, people, food, etc. Finally, most learner's dictionaries define each entry with a controlled vocabulary limited to two to three thousand words. This ensures a consistency in definitions, ease of understanding, and avoids circular – looping – definitions.

General lexicographic sources for English include the *Longman Dictionary of Contemporary English* (LDOCE, Procter 1978), the *Oxford Advanced Learner's Dictionary* (OALD, Hornby 1995), the *Collins Cobuild English Language Dictionary* (COBUILD, Sinclair 1987) or the *Cambridge International Dictionary of English* (CIDE, Procter 1995). Among them, the computerized version of the LDOCE gained the largest popularity within the academic computational linguistics community.

13.3.1 The Lexicon and Word Senses

As we saw in Chap. 1, many words are ambiguous, that is, a same string of letters has more than one meaning. Most dictionaries arrange homonyms that have clearly different meanings under different entries. The OALD, (1995 edition) lists three entries for *bank*: two nouns, *organization* and *raised ground*, and a verb *turn*. Polysemy, which refers to meaning variations within a same entry, is subtler. Dictionaries divide entries into submeanings with more or less precision according to the dictionary. These are the senses of a word. Let us take the example of the sentence

The patron ordered a meal

to realize concretely what word senses are. We will annotate each word of the sentence with its correct sense, and we will use definitions of the OALD to carry out this operation.

In the sentence, there are three content words: *patron*, *order*, and *meal*. For each of these words, the OALD lists more than one sense. *Patron* has one main entry for which the dictionary makes out two meanings:

1. a person who gives money or support to a person, an organization, a cause or an activity
2. a customer of a shop, restaurant, theater

Order has two entries. The first one is a noun and the other is a verb for which the OLAD details four sub-meanings:

1. to give an order to somebody
2. to request somebody to supply or make goods, etc.
3. to request somebody to bring food, drink, etc. in a hotel, restaurant, etc.
4. to put something in order

And finally, *meal* has two entries – two homographs – one as in *oatmeal*, and the other being divided into two submeanings:

1. an occasion where food is eaten
2. the food eaten on such occasion

That is, with such a simple sentence, we already have 16 choices ($2 \times 4 \times 2$; Table 13.1).

Classically, senses of a word are numbered relatively to a specific dictionary using the entry number and then the sense number within the entry. So *requesting somebody to bring food, drink, etc. in a hotel, restaurant, etc.*, which is the 3rd sense of the 2nd entry of *order* in the OALD is denoted **order (2.3)**. The proper sense sequence of *The patron ordered a meal* is then *patron* (1.2) *order* (2.3) *meal* (1.2).

Table 13.1. Sense ambiguity in the sentence *The patron ordered a meal*.

Words	Definitions	OALD sense numbers
The patron	**Correct sense:**	
	A customer of a shop, restaurant, theater	**1.2**
	Alternate sense:	
	A person who gives money or support to a person, an organization, a cause or an activity	**1.1**
ordered	**Correct sense:**	
	To request somebody to bring food, drink, etc in a hotel, restaurant etc.	**2.3**
	Alternate senses:	
	To give an order to somebody	**2.1**
	To request somebody to supply or make goods, etc.	**2.2**
	To put something in order	**2.4**
a meal	**Correct sense:**	
	The food eaten on such occasion	**1.2**
	Alternate sense:	
	An occasion where food is eaten	**1.1**

13.3.2 Verb Models

Dictionaries contain information on words' pronunciations, parts of speech, declension, and conjugation models. Some enrich their annotations with more precise syntactic structures such as the verb construction. In effect, most verbs constrain their subject, object, or adjuncts into a relatively rigid construction (Table 13.2).

Table 13.2. Some verb constructions.

English	*depend* + *on* + object noun group
	I like + verb-*ing* (gerund)
	require + verb-*ing* (gerund)
French	*dépendre* + *de* + object noun group
	Ça me plaît de + infinitive
	demander + *de* + infinitive
German	*hängen* + *von* + dative noun group + *ab*
	es gefällt mir + *zu* + infinitive
	verlangen + accusative noun group

Some dictionaries such as the OALD or the LDOCE provide the reader with this argument structure information. They include the traditional transitive and intransitive verb distinction, but descriptions go further. The OALD itemized 28 different

types of verb patterns. Intransitive verbs, for example, are subdivided into four categories:

- **V** are verbs used alone.
- **Vpr** are verbs followed by a prepositional phrase.
- **Vadv** are verbs followed by an adverb.
- **Vp** are verbs followed by a particle.

A verb entry contains one or more of these models to indicate possible constructions.

Some dictionaries refine verb patterns with semantic classes. They indicate precisely the ontological type of the subject, direct object, indirect object, and sometimes adjuncts. Verbs with different argument types will be mapped onto as many lexical senses. For instance, Rich and Knight (1991) quote three kinds of *wanting*:

1. wanting something to happen
2. wanting an object
3. wanting a person

We can map the 2nd construction onto a DCG rule specifying it in its arguments:

```
%% word(+POS, +Construction, +Subject, +Object)

word(verb, transitive, persons, phys_objects) -->
  [want].
```

Argument types enforce constraints, making sure that the subject is a person and that the object is a physical object. These are called **selectional restrictions**. They may help parsing by reducing syntactic ambiguity.

The LDOCE lists selectional restrictions of frequent verbs that give the expected semantic type of their subject and objects. It uses semantic classes such as inanimate, human, plant, vehicle, etc. The Collins Robert French–English dictionary (Atkins 1996) is another example of a dictionary that includes such ontological information with a large coverage.

13.3.3 Definitions

The main function of dictionaries is to provide the user with definitions, that is, short texts describing a word. The typical definition of a noun first classifies it in a *genus* or superclass using a hypernym. Then, it describes in which way the noun is specific using attributes to differentiate it from other members of the superclass. This part of the definition is called the *differentia specifica*. Examples from the OALD include (general in bold and specific underlined):

bank (1.1): ***a land*** *sloping up along each side of a canal or a river.*
hedgehog: ***a small animal*** *with stiff spines covering its back.*
waiter: ***a person*** *employed to serve customers at their table in a restaurant, etc.*

from *Le Robert Micro* (Rey 1988)

bord (1.1): ***contour, limite, extrémité*** *d'une surface*.
hérisson (1.1): ***petit mammifère*** *au corps recouvert de piquants, qui se nourrit essentiellement d'insectes*.
serveur (1.1): ***personne*** *qui sert les clients dans un café, un restaurant*.

and from *Der kleine Wahrig* (Wahrig 1978)

Ufer (1.1): ***Rand*** *eines Gewässers, Gestade*.
Igel (1.1): ***ein kleines insektfressendes Säugetier*** *mit kurzgedrungenem Körper und auf dem Rücken aufrichtbaren Stacheln.*
Ober (1.2) –> Kellner: ***Angestellter*** *in einer Gaststätte zum Bedienen der Gäste*.

13.4 An Example of Exhaustive Lexical Organization: WordNet

WordNet (Miller 1995, Fellbaum 1998) is a lexical database of English. It is probably the most successful attempt to organize word information with a computer. It has served as a research model for other languages such as Dutch, German, Italian, or Spanish. A key to this success is WordNet's coverage – it contains more than 120,000 words – and its liberal availability online: users can download it under the form of Prolog facts from its home at Princeton University.

WordNet arranges words or word forms along with word meanings into a lexical matrix (Fig. 13.4). The lexical matrix addresses both synonymy and polysemy. A horizontal line defines a set of synonymous words – a *synset* in WordNet's parlance. A column shows the different meanings of a word form. In Fig. 13.4, F_1 and F_2 are synonyms (both have meaning M_1) and F_2 is polysemous (it has meanings M_1 and M_2). Synsets are the core of WordNet. They represent concepts and knowledge that they map onto words.

Word	**Word forms**				
meanings	F_1	F_2	...	...	F_n
M_1	$E_{1,1}$	$E_{1,2}$			
M_2		$E_{2,2}$			
M_m					$E_{m,n}$

Fig. 13.4. The lexical matrix (Miller et al. 1993).

From synonymy and synsets, WordNet sets other semantic relations between words, taking their part of speech into account. WordNet creators found this property relevant, citing cognitive investigations: when people have to associate words spontaneously, they prefer consistently to group words with the same part of speech rather than words that have a different one.

WordNet considers open class words: nouns, verbs, adjectives, and adverbs. It has set aside function words. According to classes, the organization and relationships

between words are somewhat different. However, semantic relations remain based on synsets and thus are valid for any word of a synset.

13.4.1 Nouns

WordNet singles out twenty-five primitive concepts or **semantic primes** (Fig. 13.5), and it partitions the noun set accordingly. Within each of the corresponding topics, WordNet uses a hypernymic organization and arranges nouns under the form of a hierarchical lexical tree. WordNet contains 95,000 nouns.

{*act, action, activity*}	{*food*}	{*possession*}
{*animal, fauna*}	{*group, collection*}	{*process*}
{*artifact*}	{*location, place*}	{*quantity, amount*}
{*attribute*}	{*motive*}	{*relation*}
{*body, corpus*}	{*natural object*}	{*shape*}
{*cognition, knowledge*}	{*natural phenomenon*}	{*state, condition*}
{*communication*}	{*person, human being*}	{*substance*}
{*event, happening*}	{*plant, flora*}	{*time*}
{*feeling, emotion*}		

Fig. 13.5. WordNet's 25 semantic primes.

In addition to the 25 base domains, WordNet adds top divisions (Fig. 13.6). This enables it to gather some classes and to link them to a single node. Figure 13.7 shows the hierarchy leading to {*thing, entity*}.

{*entity, something*}	{*state*}	{*group, grouping*}
{*psychological feature*}	{*event*}	{*possession*}
{*abstraction*}	{*act, human action, human activity*}	{*phenomenon*}

Fig. 13.6. Nouns' top nodes.

To picture the word hierarchy and synsets with an example, let us take *meal*. It has two senses in WordNet:

1. *meal, repast* – (the food served and eaten at one time)
2. *meal* – (coarsely ground foodstuff; especially seeds of various cereal grasses or pulse)

For sense 1, synonyms are *nutriment*, *nourishment*, *sustenance*, *aliment*, *alimentation*, and *victuals*; and hypernyms are (from the word up to the root):

- *nutriment, nourishment, sustenance, aliment, alimentation, victuals* – (a source of nourishment)

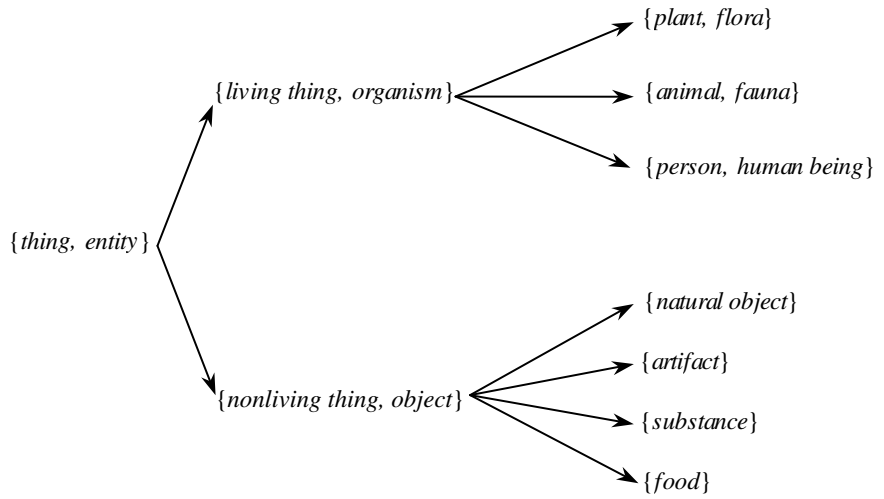

Fig. 13.7. {*thing, entity*} node of WordNet's hierarchy.

- *food, nutrient* – (any substance that can be metabolized by an organism to give energy and build tissue)
- *substance, matter* – (that which has mass and occupies space; "an atom is the smallest indivisible unit of matter")
- *object, physical object* – (a physical (tangible and visible) entity; "it was full of rackets, balls, and other objects")
- *entity, something* – (anything having existence (living or nonliving))

13.4.2 Adjectives

WordNet divides adjectives into two general classes: descriptive and relational, and into a more specific one: color adjectives. WordNet contains 20,000 adjectives.

Descriptive adjectives modify a noun and qualify one of its attributes. Examples include *hot* and *cold*, as in *hot meal* and *cold meal*, where *hot* and *cold* both describe the temperature attribute of *meal*. Another example is *heavy* and *light*, which give a value to the weight attribute of a noun (more precisely to the object it represents). As for other words, adjectives are grouped into synsets, and for each adjective synset, there is a link to the attribute it describes.

In addition to synonymy, WordNet uses antonymy as a core concept to organize descriptive adjectives. It clusters all of them around bipolar couples: word–antonym together with their respective synsets. *Hot* and *cold* or *wet* and *dry* are typical couples of antonyms, and WordNet enumerates 2,500 of them.

Antonymy relation, however, is not valid for all the members of a synset. *Torrid* is a synonym of *hot* but it cannot be considered as an antonym of *cold*. To cope with this, WordNet makes a distinction between bipolar antonymy and opposite concepts

– or indirect antonyms. There is no direct antonym for *torrid*, but using its synset, WordNet can link it indirectly to *cold* via *hot*.

Relational adjectives (**pertainyms**) such as *fraternal*, *contextual*, or *dental* are modified nouns and behave much like them on the semantic side, although they have the syntactic properties of adjectives. WordNet encodes them with a reference to their related noun: *fraternal* with *fraternity* or *brother*, *contextual* with *context*, and *dental* with *teeth* or *dentistry*. As opposed to descriptive adjectives, WordNet does not associate them to an attribute.

13.4.3 Verbs

WordNet partitions verbs into 15 categories. Fourteen of these categories are semantic domains: bodily functions and care, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social interaction, and weather. A last part contains verbs referring to states: verbs of being, having, and spatial relations (Table 13.3). WordNet has a total of 10,300 verbs.

Table 13.3. Name and description of verb files provided with the WordNet 1.6 distribution.

File	Description
Body	Verbs of grooming, dressing, and bodily care
Change	Verbs of size, temperature change, intensifying, etc.
Cognition	Verbs of thinking, judging, analyzing, doubting
Communication	Verbs of telling, asking, ordering, singing
Competition	Verbs of fighting and athletic activities
Consumption	Verbs of eating and drinking
Contact	Verbs of touching, hitting, tying, digging
Creation	Verbs of sewing, baking, painting, performing
Emotion	Verbs of feeling
Motion	Verbs of walking, flying, swimming
Perception	Verbs of seeing, hearing, feeling
Possession	Verbs of buying, selling, owning
Social	Verbs of political and social activities and events
Stative	Verbs of being, having, spatial relations
Weather	Verbs of raining, snowing, thawing, thundering

The first relation WordNet sets between verbs is synonymy, as for other words. However, synonymy is more delicate to delimit because verb meanings are quite sensitive to the context. That is, two verbs with apparently the same meaning, such as *rise* and *ascend*, do not occur with the same type of subject. This is a general case, and most verbs are selective with the type of their nominal arguments: subject, object, or adjunct. Moreover, as a verb often has no lexical synonym, WordNet encodes synsets with a small explanation – a gloss. For example, the verb *order* has nine senses whose sense 2 is represented by the synset {*order, make a request for*

something}. *Bring* has 11 senses, and sense 1 is the synset {*bring, convey, take, take something or somebody with oneself somewhere*}.

Then, WordNet organizes verbs according principles similar to hyponymy and meronymy for nouns. However, it cannot apply these principles directly because they do not match exactly that of nominals. WordNet replaces them respectively with troponymy and entailment.

WordNet designers found the `is_a` relationship not relevant or clumsy for verbs (Fellbaum 1998, p. 79):

> *to amble* is a kind of *to walk* is not a felicitous sentence.

To name specializations of more generic verbs, they coined the word **troponyms**. *Amble* is then a troponym of *walk*. This is roughly a kind of verbal hyponymy related to a manner, a cause, or an intensity that makes the description of an action more precise. Since *tropos* is a Greek word for manner or fashion, it enables us to rephrase the hierarchical relation between *amble* and *walk* as *to amble is to walk in a particular manner*.

The second principle of verb organization is **entailment** – or implication – as *he is snoring* implies *he is sleeping*, or *she is limping* implies *she is walking*. Relations between verbs in these examples are not of the same kind. The latter is related to troponymy: *limping* is a specialization or an extension of *walking*. The former is an inclusion: the action of *snoring* is always included in an action of *sleeping*. In total, WordNet makes out four kinds of entailments. In addition to extension and inclusion, the two other entailments are backward presupposition – an action must have been preceded by another one, as with the pair *succeed/try* – and cause – an action leads to another one, as with *give/have*.

13.5 Automatic Word Sense Disambiguation

Although ambiguity is ubiquitous in texts, native speakers recognize the proper sense of a word intuitively. In the beginning of computational linguistics, some people declared it a human faculty impossible to reproduce automatically. This is no longer the case. There have been considerable improvements recently, and researchers have good reason to believe that a computer will be able to discriminate among word senses. Here we will present an overview of techniques to carry out word sense disambiguation, that, alone or combined, show promising results.

13.5.1 Senses as Tags

Let us again consider the sentence *The patron ordered a meal*. Solving ambiguity has an obvious definition. It consists in linking a word with its correct sense entry in a dictionary. We can recast this as a tagging problem. We regard a dictionary as a sense inventory and senses as a finite set of labels that we call semantic tags. The number of tags per word ranges from one to more than a dozen. Ambiguous words

can receive several tags, and disambiguation consists in retaining one single tag – a unique sense – per word. Semantic tagging applies most frequently to open class words: nouns, verbs, adjectives, and adverbs.

Compared to part-of-speech tagging, a major contrast comes from the tagset. Dictionaries have somewhat different classifications of word senses and there is no complete consensus on them. According to authors, sense division is also finer or coarser. Applications can even use sets of word senses specifically designed for them. If broad categories of senses are common across most dictionaries and can be transcoded, there always will be some cases with a lack of agreement. The dissimilarity in classification and how we perceive senses can be measured by asking two or more different people to annotated a same text. The agreement between manual annotators – or **interannotator agreement** – is usually significantly lower for semantic tagging than, say, for part-of-speech annotation. There is no definitive solution to it, however. We must be aware of it and live with it.

We can carry out semantic tagging using techniques similar to those that we have used with parts of speech. Namely, we can resort to numerical or symbolic techniques. Numerical techniques attempt to optimize a sequence of semantic tags using statistics from a hand-annotated corpus. Symbolic techniques apply constraints to discard wrong semantic readings and retain the good ones.

SemCor (Landes et al. 1998) is a frequently used resource to train systems for English. It comes as a freely available corpus in which all the words are annotated with the WordNet nomenclature.

13.5.2 Associating a Word with a Context

The basic idea of most disambiguation techniques is to use the context of a word (Wilks et al. 1996, Chap. 11). The noun *bank*, for example, has two major senses[1] that will probably appear in clear-cut contexts. Sense one (**bank1**) resorts to finance and money; sense 2 (**bank2**) pertains to riversides and sloping ground. Context may be given by the words of the sentence or of the paragraph where the word occurs. This means that, depending on the words surrounding *bank* or what the text is about, a reader can select one of its two senses.

Some finer and more local relations such as the order of two words or the grammatical relations may also give the context. Disambiguating *meal* in *The patron ordered a meal* requires such considerations, because the two senses of this word belong to the same topic:

1. an occasion where food is eaten
2. the food eaten on such occasion

13.5.3 Guessing the Topic

The idea of this technique is first to define a limited number of topics, that is, a list of general areas, to attach a topic to each sense of words, and then to guess the

[1] OALD lists a third sense of *bank* as being a row of similar objects.

topic (or topics) of a paragraph or of a sentence. This technique implies that correct word senses will make the paragraph topic converge and enable us to discard senses attached to other topics. To make disambiguation possible, topics must, of course, be different along with each sense of a word.

According to applications, topics may come from dictionaries that flag some words with broad classifications – subject tags. For instance, the LDCOE categorizes words with 300 subject codes or domains that we can use as topics: agriculture, business, economics, engineering, etc. These tags usually annotate more specialized words.[2] Topics could also be a small set of hypernyms drawn from a comprehensive lexical database. For instance, using WordNet, **bank1** could be attached to financial institution (finance or institution) and **bank2** to slope:

```
%% topic(?Word, ?OALD_Sense, ?Topic).

topic(bank, bank1, [finance, institution]).
topic(bank, bank2, [slope]).
```

The disambiguation algorithm operates on a context that corresponds to a sequence of words such as a paragraph, a fixed number of sentences, or a fixed number of words, from 10 to 100, where the topic is supposed to be stable. A procedure annotates the words in the window with the possible subject tags when they are available. It yields possible sense sequences. The algorithm then retains the sense sequence that has the maximum of subject tags in common. A variation of this algorithm annotates nouns only. This method is referred to as a **bag-of-words** approach because it does not take the word order into account.

13.5.4 Naïve Bayes

The naïve Bayes classifier is an alternate statistical strategy that uses the bag-of-word approach. It also computes the sense of a word given its context. For a polysemous word w with n senses s_1, s_2, ..., s_n, the context C is defined as the sequence of words surrounding it: $w_{-m}, w_{-m+1}, ..., w_{-1}, w, w_1, ..., w_{m-1}, w_m$. The optimal sense $\hat{s}$ corresponds to $\underset{s_i, 1 \leq i \leq n}{\arg\max} P(s_i|C)$.

Using the Bayes rules, we have:

$$\begin{aligned}\hat{s} &= \underset{s_i, 1 \leq i \leq n}{\arg\max} P(s_i)P(C|s_i), \\ &= \underset{s_i, 1 \leq i \leq n}{\arg\max} P(s_i)P(w_{-m}, w_{-m+1}, ..., w_{-1}, w_1, ..., w_{m-1}, w_m|s_j).\end{aligned}$$

And using the bag-of-word assumption, we replace

$$P(w_{-m}, w_{-m+1}, ..., w_{-1}, w_1, ..., w_{m-1}, w_m|s_j)$$

with the product of probabilities:

[2] LDOCE annotates the rest of nonspecialized words with another set of semantic codes: the key concepts.

$$\hat{s} = \underset{s_i, 1 \leq i \leq n}{\arg\max} P(s_j) \prod_{i=-m, i \neq 0}^{m} P(w_i | s_j).$$

$P(s_j)$ and $P(w_i|s_j)$ are both estimated from hand-annotated corpora.

13.5.5 Using Constraints on Verbs

As we saw, most verb arguments have a structure and a semantic type that is relatively rigid. Another set of disambiguation techniques exploits these properties and takes verb constructions and local relations into account. We start here from clauses, and for each one we detect the verb group and noun groups. The idea is to apply the selectional restrictions imposed by the verb group to its depending noun groups and thus reject wrong senses.

This technique needs a group detector and a shallow parser to identify the verbs' subject and object. The sense tagger operates on headwords, that is, here on the main noun and the main verb of each group. The tagger goes through the verbs that it annotates with their possible semantic constructions. It also annotates nouns with their possible senses. Finally, for each verb sense, the tagger retains subject and object senses that agree with the selectional restrictions.

Although this technique sets aside some parts of sentences, such as adjuncts, it reduces ambiguity and can be used with a combination of other techniques. In contrast to the previous technique, it has a more local viewpoint.

In addition, we can operate a disambiguation within groups using other selectional restrictions on adjectives and adverbs. We need to extend the description of adjectives with features giving the semantic type of the noun they expect to modify. Adverbs also have to include their modifier type. As an example, the word *mean* can have the properties of being an adjective and of qualifying only persons:

```
%% word(+Category, +Qualify)

word(adjective, persons) --> [mean].
```

13.5.6 Using Dictionary Definitions

We saw that using the naïve Bayes approach to tag senses in unrestricted texts requires an immense hand-annotation effort. It is possible to avoid it using unsupervised methods. Unsupervised methods have no training step or are trained on raw texts. These techniques are very appealing, especially in word sense disambiguation, because they avoid the need of human labor to annotate the words.

Wilks and Stevenson (1997) described an algorithm that only uses word definitions from general dictionaries as semantic resource. Their method was inspired by a paper by Lesk (1986).

The algorithm tags each word with all its possible senses listed in the dictionary and links each sense with its definition in a dictionary. It first applies constraints on parts of speech and then identifies the context using the definitions: it selects senses

whose definitions overlap best within the range of a window of N words, a sentence, or a paragraph. This is made easier with dictionaries, such as the LDOCE, whose definitions are written using a controlled defining vocabulary. Simplified main steps of the program are:

1. A name recognition module identifies the proper nouns of the text.
2. A lemmatization module transforms each word into its canonical form. It associates each content word with its set of possible senses listed in the dictionary and with the corresponding textual definitions. Words occurring in definitions are also lemmatized.
3. A part-of-speech tagger annotates each word with its part of speech. At this step, the program can discard some senses because they have grammatical categories different from that of the words in the sentence.
4. The algorithm then computes the definition overlap for each sequence of possible senses. The overlap function considers a sequence of senses and their textual definition – one definition per word. The algorithm concatenates definitions of this sequence and counts the occurrences of each definition word: n. Each definition word brings a score of $n - 1$. So, if a definition word appears once, it will contribute nothing to the function; if it appears twice, it will contribute 1, and so on. Then, the algorithm adds up the counts and associates this score to the sense sequence.
5. The algorithm retains the sequence that has the maximum overlap, which is the largest number of definition words in common.

Wilks and Stevenson (1997) improved this algorithm using topics as defined in Sect. 13.5.3. Basically, they compute an overlap function for topics within the range of a paragraph:

6. The algorithm annotates nouns of a paragraph with possible subject tags when available. It retains the sequence that has the maximum of subject tags in common. This computation is similar to that of step 4.
7. The results of steps 4 and 6 are combined in a simplistic way. When both tags do not correspond, the first one in the dictionary entry list is retained. This is based on the assumption that entries are ordered by frequency of occurrence.

Step 4 of this algorithm can lead to very intensive computations. If a sentence has 15 words with 6 senses each, it leads to $6^{15} \sim 4.7\ 10^{11}$ intersections. Wilks and Stevenson used simulated annealing to approximate the function. See also Wilks et al. (1996, Chap. 11).

13.5.7 An Unsupervised Algorithm to Tag Senses

Yarowsky (1995) proposed a slightly supervised and effective algorithm based on two assumptions on sense distribution:

- Nearby words provide strong clues to the sense of a word. This means that a word has **one sense per collocation**.

- The sense of a word is consistent within any given document. This means that a word has **one sense per discourse**.

The algorithm is basically a classifier. Given a polysemous word w with n senses s_1, s_2, ..., s_n and a set of examples of the word surrounded by the neighboring words, the algorithm assigns each example a class corresponding to one of the senses. Each word in the examples is defined by a set of features, which are, as for naïve Bayes, the surrounding words. The algorithm starts from a few manually annotated examples that serve as a seed set to derive incrementally a sequence of classifiers for the remaining unlabeled examples. It uses an objective function that measures the performance of the classification. The algorithm is repeated until it has classified all the examples.

The algorithm has an initialization step and two loops. It extracts the set of all the examples of word w with the surrounding words from the training corpus. It results in N contexts c_1, ..., c_N of, say, ten words, centered around w. These examples will be the input. In its original article, Yarowsky used the word *plant* and its two main senses $s_1 = living$ and $s_2 = factory$. The algorithm gradually annotates all the examples of the corpus with one of the two senses. It produces a sequence of annotated corpora $Corpus(0)$, $Corpus(1)$, ..., $Corpus(n)$ and builds classifiers that correspond to the sets of collocations of the first sense, $Coll_1^k$, and of the second one, $Coll_2^k$. $Corpus(0)$ is the original, unannotated set of examples.

1. **Initialization**. This step manually identifies initial collocations, and the first sense classifier tags the examples whose context contains one of the collocations with the corresponding sense label. Yarowsky used the words *life* for the first sense, $Coll_1^1 = \{life\}$, and *manufacturing* for the second one, $Coll_2^1 = \{manufacturing\}$. Both words enabled the disambiguation 2% of the examples in $Corpus(1)$.
2. **Outer Loop**. This loop uses the "one sense per collocation" principle. It identifies the examples where the intersection of the context and one of the collocation sets is nonempty: $c_k \cap Coll_i^j \neq \emptyset$ with $1 \leq k \leq N$, $i = 1, 2$, and j is the iteration index of the loop. It annotates the corresponding examples with the sense s_i. It results in $Corpus(j)$. In Yarowksy's paper, contexts of *plant* that contained one word of the first set were tagged with the first sense, and others that contained one word of the second set were tagged with the second sense. The algorithm applies optionally the "one sense per discourse" constraint.
   - **Inner Loop**. The objective function determines for each sense other collocations that partition the training data $Corpus(j)$ and ranks them by the purity of the distribution. It builds new sets of classifiers $Coll_i^{j+1}$ with collocations where the objective function is above a certain threshold. This step identifies *cell*, *microscopic*, *animal*, and *species* as collocates of the first sense $Coll_1^{j+1} = \{life, cell, microscopic, animal, species\}$ and *equipment*, *employee*, and *automate* as collocates of the second sense: $Coll_2^{j+1} = \{manufacturing, equipment, employee, automate\}$.
3. Repeat the outer loop until it converges (the partition is stable).

The algorithm identifies collocations with an objective function that determines the "strongest feature." It uses the log-likelihood ratio that is defined for a word w with two senses as $\log \frac{P(Sense_1|w_k)}{P(Sense_2|w_k)}$. It ranks the resulting values depending on w_k for all w_k members of the contexts $w_{-m}, w_{-m+1}, ..., w_{-1}, w, w_1, ..., w_{m-1}, w_m$, where the collocations the most strongly tied to a specific sense will show the largest values, either positive or negative.

The "one sense per collocation" principle implies that counts of 0 are frequent. In another paper, Yarowsky (1996) describes techniques to smooth data. Once the collocation sets have been built, the resulting classifiers can be applied to other corpora.

13.5.8 Senses and Languages

Word senses do not correspond in a straightforward way across languages. In a famous comparison, Hjelmslev (1943) exemplified it with the values of French words *arbre* 'tree', *bois* 'wood', and *forêt* 'forest' and their mapping onto German and Danish scales (Table 13.4). He went on and remarked that the word covering the material sense in French (*bois*) and in Danish (*træ*) could also have the plant sense but in different ways: a group of trees in French, a single tree in Danish. In a more striking example, Hjelmslev cited color naming that is roughly common to European languages with the exception of Celtic languages such as Welsh or Breton, which does not makes the same distinction between blue and green (Table 13.5).

Table 13.4. Values of *arbre*, *bois*, and *forêt* in German and Danish.

French	German	Danish
arbre	*Baum*	
	Holz	*Træ*
bois		
forêt	*Wald*	*Skov*

Table 13.5. Color values in French and Welsh.

French	Welsh
	gwyrdd
vert	
bleu	*glas*
gris	
	llwyd
brun	

There are many other examples where one word in English can be rendered by more words in French or German, or the reverse. Finding the equivalent word from one language to another often requires identifying its correct sense in both languages. It is no great surprise that word sense disambiguation was attempted first within the context of automatic machine translation projects.

This raises some questions about the proper granularity of sense division for a translation application. In some cases, sense division that is available in monolingual dictionaries is not sufficient and must be split within as many senses as there are in both languages combined. In other cases, all senses of one word correspond from one language to another. Therefore their distinction is not necessary and the senses can be merged. This problem is still wide open and is beyond the scope of this book.

13.6 Case Grammars

13.6.1 Cases in Latin

Some languages, like Latin, Russian, and to a lesser extent German, indicate grammatical functions in a sentence by a set of inflections: the cases. Basically, Latin cases are relative to the verb, and a case is assigned to each noun group: the noun and its depending adjectives. Latin has six cases that we can roughly associate to a semantic property:

- **Nominative** marks the subject of the sentence.
- **Accusative** indicates the object of the verb.
- **Dative** describes the beneficiary of a gift or of an action. It corresponds to the indirect object of the verb.
- **Genitive** describes the possession. As opposed to other cases, it is relative to a noun that the word in the genitive modifies or qualifies.
- **Ablative** describes the manner, the instrument, or the cause of an action. It corresponds to the adjunct function.
- **Vocative** is used to name and to address a god or a person.
- **Locative** is a seventh and an archaic case. It indicates the location of the speaker in some particular expressions.

Latin, like Russian, has quite a flexible word order. That is, we can arrange words of a sentence in different manners without modifying its meaning. The subject can appear at the beginning as well as at the end of a sentence. It has no specific location as in English or in French.

A flexible word order makes cases necessary for a sentence to be understandable. They indicate functions of groups: *who did what to whom, when, and where* and hence the arguments of a verb. Searching the subject, for example, corresponds to searching the noun phrase at the nominative case. Let us apply these principles to parse the following example:

Servus	*senatoris*	*domino*	*januam*	*clave*	*aperit*
Slave	senator	master	door	key	opens

Aperit is the verb in the third-person singular of present and means open (*aperire*). It is the predicate relative to which nouns will be the arguments. Each Latin noun has a model of inflection, also called a declension, five in total. *Servus* follows the second declension and means the slave. It is in the nominative case and hence the subject of the sentence. *Senatoris*, third declension, is the genitive case of *senator* and is the noun complement of *servus*. *Domino*, second declension, means master and is the dative of *dominus*. It corresponds to the indirect object of the verb. *Januam*, first declension, is the accusative of *janua* – door – and is the object. Finally, *clave*, third declension, is the ablative of *clavis* – the key – and the instrument of the action. Once we have identified cases, we can safely translate the sentence as:

The slave of the senator opens the door to the master with a key.

Cases are also useful to discover what goes with what such as an adjective and its head noun. Both will have the same case even if the noun group is fragmented within the sentence.

13.6.2 Cases and Thematic Roles

Case grammars stem from the idea that each verb – or each verb sense – has a finite number of possible cases. Case grammars rest on syntactic and semantic observations of languages like Latin and offer a framework to represent sentences. Hjelmslev (1935), and more recently, Fillmore (1968) are known to have posited that cases were universal and limited to a handful. Because of declensions, cases are obvious to those who learned Latin. However, it is somewhat hidden to speakers of English or French only. That is probably why, compared to compositionality, the acceptance of the case theory and its transposition to English or French has been slower.

Surveying a set of languages ranging from Estonian to Walapai, Fillmore percolated a dozen core cases, or **thematic roles**. A first classification led him to define (Fillmore 1968, p. 24):

- **Agentive** (A) – the case of the instigator of the action, which is typically animate
- **Instrumental** (I) – the case of the force or object typically inanimate causing the event
- **Dative** (D) – the case of the entity typically animate affected by the action
- **Factitive** (F) – the case of the object or being resulting from the event
- **Locative** (L) – the case of the identifying the place of the event or the orientation of the action
- **Objective** (O) – the most general case indicating the entity that is acted upon or that changes

As an example, Fillmore (1968, p. 27) attached to the verb *open* a frame containing an objective case that always occurs in the sentence, and optional instrumental and agentive cases denoted in parentheses: [O, (I), (A)]. This frame enables us to represent sentences in Table 13.6. One must note that the objective case, here filled with *the door*, sometimes corresponds to the grammatical subject and sometimes to the grammatical object.

Table 13.6. Examples of case frames (Fillmore 1968, p. 27).

Sentences	Case frames
The door opened	[O = door, (I), (A)]
John opened the door	[O = door, (I), (A) = John]
The wind opened the door	[O = door, (I), (A) = wind]
John opened the door with a chisel	[O = door, (I) = chisel, (A) = John]

To be complete and represent our Latin sentence, we add a dative case:

The slave of the senator opens the door to the master with a key.
[O = the door, (I) = a key, (A) = the slave of the senator, (D) = the master]

Later a multitude of authors proposed extensions to these cases. Most general and useful are:

- **Source** – the place from which something moves
- **Goal** – the place to which something moves
- **Beneficiary** – the being, typically animate, on whose behalf the event occurred
- **Time** – the time at which the event occurred

Over the time, Fillmore himself slightly changed the structure and name of his cases. Here is a more abstract classification of cases together with their description.

- **Agent** – primary animate energy source
- **Experiencer** – psychological locus of an experience
- **Theme** – primary moving object
- **Patient** – object which undergoes a change
- **Source** – starting point of a motion or change
- **Goal** – destination, target of a motion
- **Location** – location of an object or event
- **Path** – trajectory of a motion, between source and goal
- **Content** – content of an event of feeling, thinking, speaking, etc.

Some verbs do not fit into this case scheme, in spite of its generality. Fillmore again cited some of them such as the verb set *buy*, *sell*, *pay*, *spend*, *charge*, etc., whose cases are the quadruplet **buyer**, **seller**, **goods**, **money**, and the set *replace*, *substitute*, *swap*, etc., whose cases are **old**, **new**, **position**, **causer**. In addition, some applications may require other more specific cases.

13.6.3 Parsing with Cases

Parsing with the case grammar formalism transforms a sentence – or a part of it – into a kind of logical form: the frame. The predicate is the main verb, and its arguments represent the cases (or the roles). The parsing process merely maps noun groups or other features such as the tense or adverbs onto the cases. According to

the verbs, some cases will be obligatory, such as the agent for most verbs. They will be assigned with exactly one argument. Others cases will be optional. They will be assigned with at most one value. In addition, cases are constrained by an ontological type. Table 13.7. shows a representation of the sentence

The waiter brought the meal to the patron

which links noun groups and the verb tense to cases.

Table 13.7. *Bring* cases with constraints.

Case	Type		Value
Agentive	Animate	(Obligatory)	*The waiter*
Objective (or theme)		(Obligatory)	*the meal*
Dative	Animate	(Optional)	*the patron*
Time		(Obligatory)	past

We can relate verbs cases to Tesnière's *actants* and *circonstants* (1966), which are idiosyncratic patterns of verbs encapsulated into a predicate argument structure. Tesnière first made a distinction between the typical cases of a verb – *actants* – and its optional modifiers – *circonstants*. A verb attracts a definite number of actants corresponding to its **valence**. Drawing on the semantic side, cases fit well an ontology of nouns and lead to subcategories of verb patterns. The agent, or the subject, of verb *eat* is generally animate. The instrument of *open* should comply with the instrument ontological subtree. These semantic properties related to verb cases are another viewpoint on sectional restrictions.

13.6.4 Semantic Grammars

Originally, parsing with a case grammar was carried out using a combination of techniques: shallow parsing and "expectations" on verb arguments. First, the parser detects the verb group and its depending noun groups or noun phrases. Then, the parser fills the cases according to "markers": topological relations, ontological compatibility (selectional restrictions), prepositions, tense, and for German, syntactic cases.

In many circumstances, we can assimilate the Agent to the subject of a sentence and the Theme to the object. Languages like English and French have a rather rigid word order in a sentence, and functions correspond to a specific location relative to a the verb. The subject is generally the first noun phrase; the object comes after the verb. In German, they are inflected respectively with the nominative and accusative cases.

Adjuncts are more mobile, and a combination of constraints on prepositions and selectional restrictions can be productive to fill modifier cases such as Source, Goal, and Instrument. Prepositions such as *from*, *to*, or *into* often indicate a Source and a Goal. We can add a double-check and match them to location classes such as places,

cities, countries, etc. Other prepositions are more ambiguous, such as *by* in English, *pour* in French, and *auf* in German. Ontological categories come first as conditions to carry out the parse. They enable us to attach noun groups to classes and to choose a case complying with the selectional restrictions of the verb.

Phrase-structure rules can help us implement a limited system to process cases. It suffices to replace parts of speech and phrase categories with ontological classes in rules. This leads to **semantic grammars** dedicated to specific applications, such as this one describing a piece of the real and gory life of animals:

```
sentence --> npInsectivores, ingest,
npCrawlingInsects.
npInsectivores --> det, insectivores.
npCrawlingInsects --> det, crawlingInsects.

insectivores --> [mole].
insectivores --> [hedgehog].
ingest --> [devoured].
ingest --> [ate].
crawlingInsects --> [worms].
crawlingInsects --> [caterpillars].
det --> [the].
```

Rules describe prototypic situations, and parsing checks the compatibility of the types in the sentence. They produce a semantic parse tree.

Semantic grammars were once popular because they were easy to implement. However, they are limited to one application. Changing context or simply modifying it often requires a complete redesign of the rules.

13.7 Extending Case Grammars

13.7.1 FrameNet

The FrameNet research project started from Fillmore's theory on case grammars (1968). Reflecting on it, Fillmore noticed how difficult (impossible?) it was to work out a small set of generic cases applicable to all the verbs. He then altered his original ideas to form a new theory on **frame semantics** (Fillmore 1976). With frame semantics, Fillmore no longer considers universal cases but a set of frames resembling predicate-argument structures, where each frame is specific to a class of verbs. Frames are supposed to represent prototypical conceptual structures shared by a language community, i.e., here English.

FrameNet is a concrete outcome of the frame semantics theory. It aims at describing the frame properties of all the English verbs as well as some nouns and adjectives, and at annotating them in a large corpus. Like WordNet, FrameNet takes the shape of an extensive lexical database, which associates a word sense to a frame

with a set frame elements (FEs). FrameNet also links the frames to annotations in the 100-million word British National Corpus.

Ruppenhofer et al. (2005) list Revenge as an example of frame, which features five frame elements: Avenger, Punishment, Offender, Injury, and Injured_party. The Revenge frame serves as a semantic model to 15 lexical units, i.e., verb, noun, or adjective senses:

> *avenge.v, avenger.n, get back (at).v, get_even.v, retaliate.v, retaliation.n, retribution.n, retributive.a, retributory.a, revenge.n, revenge.v, revengeful.a, revenger.n, vengeance.n, vengeful.a*, and *vindictive.a*

where the *.v* suffix denotes a verb, *.n* a noun, and *.a* an adjective.

Once the frame was defined, the FrameNet team annotated the corresponding lexical units in sentences extracted from its corpus. The annotation identifies one lexical unit per sentence, which is the **target**, and brackets its frame elements as in these examples from Ruppenhofer et al. (2005):

1. [$_{<Avenger>}$ His brothers] **avenged** [$_{<Injured\_party>}$ him].
2. With this, [$_{<Agent>}$ El Cid] at once **avenged** [$_{<Injury>}$ the death of his son].
3. [$_{<Avenger>}$ Hook] tries to **avenge** [$_{<Injured\_party>}$ himself] [$_{<Offender>}$ on Peter Pan] [$_{<Punishment>}$ by becoming a second and better father].

Each frame element contains semantic and grammatical information split into three levels of annotation. The first level is the name of the semantic role. The second and third ones describe how a frame element is realized in the sentence: the phrase syntactic category and its grammatical function. The phrase syntactic category, i.e., noun phrases, prepositional phrases, and so on, is called the phrase type (PT). FrameNet uses a small set of grammatical functions (GFs), which are specific to the target's part of speech (i.e., verbs, adjectives, prepositions, and nouns). For the verbs, FrameNet defines four GFs: Subject, Object (Obj), Complement (Comp), and Modifier (Mod), i.e., modifying adverbs ended by *-ly* or indicating manner. FrameNet renames the subjects as external arguments (Ext).

Table 13.8 shows the three-level annotation of the sentences above. Altogether, these levels form a **valence group**. Each sentence shows a **valence pattern**, a specific set of valence groups.

13.7.2 A Statistical Method to Identify Semantic Roles

We saw that it was possible to develop a case parser using manually written rules. However, such parsers require much labor, testing, and debugging, and have an unavoidably limited coverage. We introduce now a statistical technique that uses FrameNet to identify semantic roles for unrestricted text.

Gildea and Jurafsky's (2002) algorithm takes a sequence of sentences as input and consists conceptually of two steps. The first step segments the sentences to identify a target word and the constituents that will serve as frame elements. The second step labels the frame elements with their semantic roles.

Table 13.8. The valence patterns of avenge in the sentences above and their three levels of annotations: frame element (FE), phrase type (PT), and grammatical function (GF).

Sent. 1	*avenge*	FE	Avenger	Injured_party		
		PT	NP	NP		
		GF	Ext	Object		
Sent. 2	*avenge*	FE	Agent	Injury		
		PT	NP	NP		
		GF	Ext	Obj		
Sent. 3	*avenge*	FE	Avenger	Injured_party	Offender	Punishment
		PT	NP	NP	PP	PPing
		GF	Ext	Obj	Comp	Comp

We present here a simplified version of the algorithm where we first describe the two-step procedure starting from the second step: the role identification of presegmented constituents. We then outline the constituent segmentation that uses a similar probability model. Finally, we merge the two steps into a single model that yields the best results.

Two-Step Labeling. The algorithm is based on the observation that a semantic role depends both on the phrase type and on its grammatical function, for instance, noun phrase and agent, agent and subject. The idea is then to find a set of features that expresses both dependencies. While the phrase type is accessible using a parser, this is not the case for the grammatical function. Gildea and Jurafsky use a set of three parameters to capture this function:

- The simplest one is the constituent position relative to the target, before or after. This feature correlates the fact that in English, the subject is often before the verb, while the object is after.
- The second feature is the constituent governor in the parse tree. The governor is the highest node in the tree to go from the target to the constituent. It concerns only noun phrases and has two possible values: sentence (S) or verb phrase (VP). A subject will have S as typical governor and an object VP, as shown in Fig. 13.8.
- The third feature is the path from the target to the constituent. A typical object path for a subject is $VB \uparrow VP \uparrow S \downarrow NP$ and $VB \uparrow VP \downarrow NP$ for an object. This feature is used in the boundary detection of the frame elements and not in the role-labeling step.

We add three more features: the target, constituent headwords, and sentence voice. We have seen with selectional restrictions that semantic roles in a sentence depend on an interaction between the target and the frame elements. We express these restrictions through the constituent headword. The voice value active/passive also plays a role, because it inverts the subject/agent roles. Given the features we have described, the resulting statistical model of role identification is then:

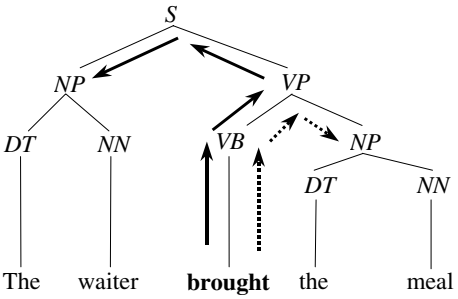

Fig. 13.8. Paths from the target to the subject (*solid lines*) and object (*dashed lines*).

$$P(r|h, pt, gov, position, voice, t),$$

where r denotes the role; h, the headword; pt, the phrase type; gov, the governor; $position$, the position of the constituent relative to the target; $voice$, the sentence voice; and t, the target.

Using the maximum likelihood, we can estimate this value from an annotated corpus:

$$P_{MLE}(r|h, pt, gov, position, voice, t) = \frac{C(r, h, pt, gov, position, voice, t)}{C(h, pt, gov, position, voice, t)}.$$

In fact, due to sparse data, it is generally not possible to compute it. Gildea and Jurafsky proposed workarounds that include linear interpolation and backoff. Both show similar performance, and here is the formula they used for linear interpolation:

$$\begin{aligned} P(r|constituent) = {} & \lambda_1 P(r|t) + \lambda_2 P(r|pt, t) \\ & + \lambda_3 P(r|pt, gov, t) + \lambda_4 P(r|pt, position, voice) \\ & + \lambda_5 P(r|pt, position, voice, t) + \lambda_6 P(r|h) \\ & + \lambda_7 P(r|h, t) + \lambda_8 P(r|h, pt, t), \end{aligned}$$

where $\sum_i \lambda_i = 1$.

The probability above does not identify the frame element boundaries. Gildea and Jurafsky proposed a model to determine if a constituent is a frame element from the path, the headword of the constituent, and the target word. The three last features appear in the condition part of the probability. To handle sparse data, they used linear interpolation:

$$P(fe|path, h, t) = \lambda_1 P(fe|path) + \lambda_2 P(fe|path, t) + \lambda_3 P(fe|h, t),$$

and set the threshold at 0.5.

Using the two-step procedure, boundary detection of frame elements and labeling of semantic roles, Gildea and Jurafsky report a precision of 67 and a recall of 48.7.

Combining Models. The valence patterns of a lexical unit, such as those shown in Table 13.7 for *avenge*: {Avenger, Injured_party}, {Agent, Injury}, and {Avenger, Injured_party, Offender, Punishment}, are not equally probable. Gildea and Jurafsky used this observation to improve the role-labeling step of their system. The most likely role assignment over a sentence of n constituents is modeled as

$$r^* = \arg\max_{r_{1\ldots n}} P(r_{1\ldots n}|t, f_{1\ldots n}),$$

where r^* is the optimal role assignment over the n constituents of a sentence; t, the target word; and f_i, the constituent features. These features are the same as in the first version of the role labeler.

Using Bayes' theorem, we rewrite the equation as

$$r^* = \arg\max_{r_{1\ldots n}} P(\{r_{1\ldots n}\}|t) \prod_i \frac{P(r_i|f_i, t)}{P(r_i|t)},$$

where $P(\{r_{1\ldots n}\}|t)$ is the probability of a valence pattern given a target word t.

Finally, Gildea and Jurafsky combined the two probability models from the frame element segmenter and the improved role labeler into a single equation:

$$r^* = \arg\max_{r_{1\ldots n}} P(\{r_{1\ldots n}\}|t) \prod_i \frac{P(r_i|f_i, fe_i, t)P(fe_i|f_i)}{P(r_i|t)},$$

where fe_i is a Boolean variable indicating whether sentence constituent i is a frame element or not. $P(fe_i|f_i)$ is computed as in the first step of the previous method. The combined equation yielded better results, with a precision of 64.6 and a recall of 61.2.

13.8 An Example of Case Grammar Application: EVAR

The EVAR application (Mast et al. 1994) is a spoken dialogue system that answers inquiries about train connections in Germany. EVAR uses a dedicated ontology to represent the domain and a case grammar to process the utterances. It is able to process exchanges like the one in Table 13.9.

13.8.1 EVAR's Ontology and Syntactic Classes

EVAR organizes nouns as a classical ontological tree restricted to the railway domain in Germany (Fig. 13.9). For instance, *train* is linked to "transport," *Hamburg* to "location," etc.

Table 13.9. A dialogue with the EVAR system.

Turn	Utterance
EVAR	*Guten Tag. Hier ist die automatische Intercityauskunft.* 'Hello, this is the automatic Intercity information system.'
User	*Ich will nach Hamburg* 'I want to go to Hamburg'
EVAR	*Sie wollen nach Hamburg fahren?* 'You want to go to Hamburg' *Wo wollen sie abfahren?* 'Where do you want to start?'
User	*In Erlangen*
EVAR	*Sie wollen von Erlangen abfahren?* 'You want to start from Erlangen' *An welchem Tag?* 'Which day?'
User	*Morgen früh* 'Tomorrow morning'
EVAR	*Sie wollen am Donnerstag den 26. Juni zwischen 6 und 12 Uhr fahren?* 'You want to travel on Thursday 26th of June between 6 and 12 AM'
User	*genau* 'Yes'
EVAR	*Ich habe mehrere Verbindungen gefunden.* 'I found connections' *Nach Hamburg HBF fahren sie ab Erlangen um 11 Uhr 1, Ankunft in Hamburg HBF...* 'To Hamburg, you can start from Erlangen at 11.01, arrival in Hamburg...' *Soll ich die Verbindungen wiederholen?* 'Shall I repeat?'
User	*Nein Danke*
EVAR	*Möchten Sie weitere Auskünfte?* 'Would you like more information?'
User	*Nein*
EVAR	*Auf wiederhören* 'Good bye'

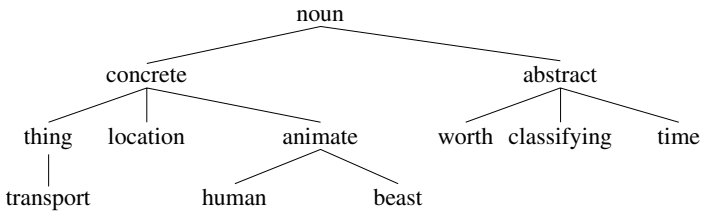

Fig. 13.9. Semantic classes in the EVAR application. After Mast et al. (1994).

13.8.2 Cases in EVAR

EVAR uses a partial parser to detect syntactic groups, notably noun groups, prepositional groups, infinitive groups, verb groups, and time and date expressions. It represents utterances with a case grammar tied to the train domain and uses the ontology in Fig. 13.9. The case system is relatively detailed – it is said to be fine grained – and consists of about 30 cases associated to verbs and also to nouns and adjectives. Table 13.10 shows some examples of case frames together with their constraints.

Table 13.10. Some verbs and nouns with their cases in the EVAR system. After Mast et al. (1994).

Word senses	Examples and cases
Fahren1.1	*Der Zug fährt von Hamburg nach München* 'The train is going from Hamburg to Munich' *Instrument*: noun group (nominative), Transport, obligatory *Source*: prepositional group (Origin), Location, optional *Goal*: prepositional group (Direction), Location, optional
Fahren1.2	*Ich fahre mit dem Zug von Hamburg nach München* 'I am going by train from Hamburg to Munich' *Agent*: noun group (nominative), Animate, obligatory *Instrument*: prepositional group (prep=mit), Transport, optional *Source*: prepositional group (Origin), Location, optional *Goal*: prepositional group (Direction), Location, optional
Abfahrt1.1	*Die Abfahrt des Zuges von Hamburg nach München* 'The departure of the train at Hamburg for Munich' *Object*: noun group (genitive), Transport, optional *Location*: prepositional group (Place), Location, optional *Time*: prepositional group (Moment), Time, optional
Verbindung1.5	*Eine Verbindung von Hamburg nach München* 'A connection from Hamburg to Munich' *Source*: prepositional group (Origin), Location, optional *Goal*: prepositional group (Direction), Location, optional

Sagerer (1990) gives the full description of the semantic cases related to EVAR.

13.9 Further Reading

Although it may not serve for immediate applications, Saussure's *Cours de linguistique générale* (1916) offers a fundamental background introduction on links between language and though. Also of interest, is Hjelmslev's *Prolegomena to a Theory of Language* (1943), which provides a complement to Saussure's views. A good introduction to the classical texts is the historical presentation on linguistics by Harris and Taylor (1997).

Electric Words by Wilks et al. (1996) provides an account on semantics that focuses on computerized dictionaries. It contains many mistakes however. Boguraev and Pustejovsky (1996) have similar concerns but address more specific points of lexical acquisition. Mel'cuk et al. (1995) propose a detailed dictionary model for French. Fellbaum (1998) gives an in-depth description of WordNet and its design. The WordNet lexical database is regularly updated and its content is available for download from http://wordnet.princeton.edu. Pustejovsky (1995) presents an alternate viewpoint on lexical structure. Dutoit (1992) describes another model that governed the implementation of Dicologique, a lexical database in French of size comparable to WordNet.

Literature on word sense disambiguation is countless. The reader can find a starting point in a dedicated special issue of *Computational Linguistics*, especially in its Introduction (Ide and Véronis 1998), which lists more than 300 references! Word sense disambiguation has made considerable progress recently. The SENSEVAL workshops benchmark competing systems for a variety of languages. Proceedings are available from the ACL Anthology (http://www.aclweb.org/anthology).

In a classical text, Fillmore (1968) gives the rationale behind case grammars. Jackendoff (1990) gives another detailed description of cases for English. Later, Fillmore started the FrameNet project that itemizes the frame elements of all the verbs. Although it is an ongoing project, the FrameNet database is well underway. Its description and content is available for download from http://framenet.icsi.berkeley.edu. Propbank or Proposition bank is a similar project aimed at annotating the Penn Treebank with semantic data (Kingsbury et al. 2002).

Automatic role labeling using statistical techniques has received considerable interest recently and was the theme of two conferences on Computational Natural Language Learning (CoNLL-2004 and CoNLL-2005). Annotated data, descriptions, and performance of the competing systems are available from the conference Web pages (http://www.cnts.ua.ac.be/conll2004/ and http://www.cnts.ua.ac.be/conll2005/).

Exercises

13.1. Implement the complete semantic net of Fig. 13.3.

13.2. Implement a graph search that finds entities linked by properties using inheritance, and test it using the `eat/2` relation.

13.3. Annotate each word of the following sentences with their possible senses:
The waiter brought the starter to the customers.
Le serveur a apporté l'entrée aux clients
Der Ober hat die Vorspeise zum Kunden gebracht.
You may use any dictionary.

13.4. Write verb syntactical models corresponding to senses of *order* in Table 13.1.

13.5. Write selectional restrictions corresponding to senses of *order* in Table 13.1.

13.6. Take a dozen or so words and, using their definition, build the corresponding ontological tree.

13.7. According to WordNet, *bring* entails *come, come up* (move toward, travel toward something or somebody or approach something or somebody). Classify this type of entailment as coextensiveness, proper inclusion, backward presupposition, or cause.

13.8. In this exercise, you will implement the word sense disambiguation algorithm outlined in Sect. 13.5.3.

- Write a Prolog program that produces all sense sequences of a given sentence. Implement the lexical database representing possible senses of *patron*, *ordered*, and *meal*, and test the program with *The patron ordered the meal*.
- Find topics associated with senses of words *patron*, *order*, and *meal*. Set these topics under the form of Prolog facts.
- Write a Prolog program that collects all the topics associated with a sense sequence.
- What is the main condition for the algorithm to produce good results?

13.9. Disambiguate by hand the senses of words in the sentence: *the patron ordered the meal* using word definitions and the algorithm of Sect. 13.5.6. You may use any dictionary.

13.10. Program the algorithm of the Exercise 13.9.