The previous two chapters focused on meaning representations for entire sentences. In those discussions, we made a simplifying assumption by representing word meanings as unanalyzed symbols like EAT or JOHN or RED. But representing the meaning of a word by capitalizing it is a pretty unsatisfactory model. In this chapter we introduce a richer model of the semantics of words, drawing on the linguistic study of word meaning, a field called **lexical semantics**, as well as the computational study of these meanings, known as **computational lexical semantics**.

In representing word meaning, we’ll begin with the **lemma** or **citation form** which we said in Chapter 4 is the grammatical form of a word that is used to represent a word in dictionaries and thesaurus. Thus carpet is the lemma for carpets, and sing the lemma for sing, sang, sung. In many languages the infinitive form is used as the lemma for the verb, so Spanish dormir “to sleep” is the lemma for duermes “you sleep”. The specific forms sung or carpets or sing or duermes are called **wordforms**.

But a lemma can still have many different meanings. The lemma bank can refer to a financial institution or to the sloping side of a river. We call each of these aspects of the meaning of bank a **word sense**. The fact that lemmas can be homonymous (have multiple senses) causes all sorts of problems in text processing. **Word sense disambiguation** is the task of determining which sense of a word is being used in a particular context, a task with a long history in computational linguistics and applications tasks from machine translation to question answering. We give a number of algorithms for using features from the context for deciding which sense was intended in a particular context.

We’ll also introduce **WordNet**, a widely-used thesaurus for representing word senses themselves and for representing relations between senses, like the IS-A relation between dog and mammal or the part-whole relationship between car and engine. Finally, we’ll introduce the task of computing **word similarity** and show how a sense-based thesaurus like WordNet can be used to decide whether two words have a similar meaning.

### 18.1 Word Senses

Consider the two uses of the lemma bank mentioned above, meaning something like “financial institution” and “sloping mound”, respectively:

(18.1) Instead, a bank can hold the investments in a custodial account in the client’s name.

(18.2) But as agriculture burgeons on the east bank, the river will shrink even more.
We represent this variation in usage by saying that the lemma *bank* has two senses.\(^1\) A *sense* (or *word sense*) is a discrete representation of one aspect of the meaning of a word. Loosely following lexicographic tradition, we represent each sense by placing a superscript on the orthographic form of the lemma as in *bank\(^1\)* and *bank\(^2\).*

The senses of a word might not have any particular relation between them; it may be almost coincidental that they share an orthographic form. For example, the *financial institution* and *sloping mound* senses of *bank* seem relatively unrelated. In such cases we say that the two senses are *homonyms*, and the relation between the senses is one of *homonymy*. Thus *bank\(^1\)* ("financial institution") and *bank\(^2\)* ("sloping mound") are homonyms, as are the sense of *bat* meaning 'club for hitting a ball' and the one meaning 'nocturnal flying animal'. We say that these two uses of *bank* are *homographs*, as are the two uses of *bat*, because they are written the same. Two words can be homonyms in a different way if they are spelled differently but pronounced the same, like *write* and *right* or *piece* and *peace*. We call these *homophones* and we saw in Ch. 5 that homophones are one cause of real-word spelling errors.

Homonymy causes problems in other areas of language processing as well. In question answering or information retrieval, we can do a much better job helping a user who typed "bat care" if we know whether they are vampires or just want to play baseball. And they will also have different translations; in Spanish the animal bat is a *murciélago* while the baseball bat is a *bate*. Homographs that are pronounced differently cause problems for speech synthesis (Chapter 26) such as these homographs of the word *bass*, the fish pronounced *bae s* and the instrument pronounced *be y s*.

(18.3) The expert angler from Dora, Mo., was fly-casting for *bass* rather than the traditional trout. (18.4) The curtain rises to the sound of angry dogs baying and ominous *bass* chords sounding.

Sometimes there is also some semantic connection between the senses of a word. Consider the following example:

(18.5) While some banks furnish blood only to hospitals, others are less restrictive.

Although this is clearly not a use of the "sloping mound" meaning of *bank*, it just as clearly is not a reference to a charitable giveaway by a financial institution. Rather, *bank* has a whole range of uses related to repositories for various biological entities, as in *blood bank*, *egg bank*, and *sperm bank*. So we could call this "biological repository" sense *bank\(^3\)*. Now this new sense *bank\(^3\)* has some sort of relation to *bank\(^1\)*; both *bank\(^1\)* and *bank\(^3\)* are repositories for entities that can be deposited and taken out; in *bank\(^1\)* the entity is monetary, whereas in *bank\(^3\)* the entity is biological.

When two senses are related semantically, we call the relationship between them *polysemy* rather than homonymy. In many cases of polysemy, the semantic relation between the senses is systematic and structured. For example, consider yet another sense of *bank*, exemplified in the following sentence:

(18.6) The bank is on the corner of Nassau and Witherspoon.

This sense, which we can call *bank\(^4\)*, means something like “the building belonging to a financial institution”. It turns out that these two kinds of senses (an

\(^1\) Confusingly, the word “lemma” is itself ambiguous; it is also sometimes used to mean these separate senses, rather than the citation form of the word. You should be prepared to see both uses in the literature.
organization and the building associated with an organization) occur together for many other words as well (school, university, hospital, etc.). Thus, there is a systematic relationship between senses that we might represent as

\[
\text{BUILDING} \leftrightarrow \text{ORGANIZATION}
\]

This particular subtype of polysemy relation is often called metonymy. Metonymy is the use of one aspect of a concept or entity to refer to other aspects of the entity or to the entity itself. Thus, we are performing metonymy when we use the phrase \textit{the White House} to refer to the administration whose office is in the White House. Other common examples of metonymy include the relation between the following pairings of senses:

\begin{itemize}
\item Author (\textit{Jane Austen wrote Emma}) $\leftrightarrow$ Works of Author (\textit{I really love Jane Austen})
\item Tree (\textit{Plums have beautiful blossoms}) $\leftrightarrow$ Fruit (\textit{I ate a preserved plum yesterday})
\end{itemize}

While it can be useful to distinguish polysemy from unrelated homonymy, there is no hard threshold for how related two senses must be to be considered polysemous. Thus, the difference is really one of degree. This fact can make it very difficult to decide how many senses a word has, that is, whether to make separate senses for closely related usages. There are various criteria for deciding that the differing uses of a word should be represented as distinct discrete senses. We might consider two senses discrete if they have independent truth conditions, different syntactic behavior, and independent sense relations, or if they exhibit antagonistic meanings.

Consider the following uses of the verb \textit{serve} from the WSJ corpus:

(18.7) They rarely \textit{serve} red meat, preferring to prepare seafood.

(18.8) He \textit{served} as U.S. ambassador to Norway in 1976 and 1977.

(18.9) He might have \textit{served} his time, come out and led an upstanding life.

The \textit{serve} of \textit{serving red meat} and that of \textit{serving time} clearly have different truth conditions and presuppositions; the \textit{serve} of \textit{serve as ambassador} has the distinct subcategorization structure \textit{serve as NP}. These heuristics suggest that these are probably three distinct senses of \textit{serve}. One practical technique for determining if two senses are distinct is to conjoin two uses of a word in a single sentence; this kind of conjunction of antagonistic readings is called zeugma. Consider the following ATIS examples:

(18.10) Which of those flights \textit{serve} breakfast?

(18.11) Does Midwest Express \textit{serve} Philadelphia?

(18.12) ?Does Midwest Express \textit{serve} breakfast and Philadelphia?

We use (?) to mark those examples that are semantically ill-formed. The oddness of the invented third example (a case of zeugma) indicates there is no sensible way to make a single sense of \textit{serve} work for both breakfast and Philadelphia. We can use this as evidence that \textit{serve} has two different senses in this case.

Dictionaries tend to use many fine-grained senses so as to capture subtle meaning differences, a reasonable approach given that the traditional role of dictionaries is aiding word learners. For computational purposes, we often don’t need these fine distinctions, so we may want to group or cluster the senses; we have already done this for some of the examples in this chapter.

How can we define the meaning of a word sense? Can we just look in a dictionary? Consider the following fragments from the definitions of right, left, red, and blood from the \textit{American Heritage Dictionary} (Morris, 1985).
Right adj. located nearer the right hand esp. being on the right when facing the same direction as the observer.  
Left adj. located nearer to this side of the body than the right.  
Red n. the color of blood or a ruby.  
Blood n. the red liquid that circulates in the heart, arteries and veins of animals.

Note the circularity in these definitions. The definition of right makes two direct references to itself, and the entry for left contains an implicit self-reference in the phrase this side of the body, which presumably means the left side. The entries for red and blood avoid this kind of direct self-reference by instead referencing each other in their definitions. Such circularity is, of course, inherent in all dictionary definitions; these examples are just extreme cases. For humans, such entries are still useful since the user of the dictionary has sufficient grasp of these other terms.

For computational purposes, one approach to defining a sense is to make use of a similar approach to these dictionary definitions; defining a sense through its relationship with other senses. For example, the above definitions make it clear that right and left are similar kinds of lemmas that stand in some kind of alternation, or opposition, to one another. Similarly, we can glean that red is a color, that it can be applied to both blood and rubies, and that blood is a liquid. Sense relations of this sort are embodied in on-line databases like WordNet. Given a sufficiently large database of such relations, many applications are quite capable of performing sophisticated semantic tasks (even if they do not really know their right from their left).

18.2 Relations Between Senses

This section explores some of the relations that hold among word senses, focusing on a few that have received significant computational investigation: synonymy, antonymy, and hypernymy, as well as a brief mention of other relations like meronymy.

### 18.2.1 Synonymy and Antonymy

When two senses of two different words (lemmas) are identical, or nearly identical, we say the two senses are 

**synonyms**. Synonyms include such pairs as

- **couch**/**sofa**  
- **vomit**/**throw up**  
- **filbert**/**hazelnut**  
- **car**/**automobile**

A more formal definition of synonymy (between words rather than senses) is that two words are synonymous if they are substitutable one for the other in any sentence without changing the truth conditions of the sentence. We often say in this case that the two words have the same **propositional meaning**.

While substitutions between some pairs of words like **car**/**automobile** or **water**/**H₂O** are truth preserving, the words are still not identical in meaning. Indeed, probably no two words are absolutely identical in meaning, and if we define synonymy as identical meanings and connotations in all contexts, there are probably no absolute synonyms. Besides propositional meaning, many other facets of meaning that distinguish these words are important. For example, **H₂O** is used in scientific contexts and would be inappropriate in a hiking guide; this difference in genre is part of the meaning of the word. In practice, the word **synonym** is therefore commonly used to describe a relationship of approximate or rough synonymy.
18.2 • Relations Between Senses

Synonymy is actually a relationship between senses rather than words. Considering the words *big* and *large*. These may seem to be synonyms in the following ATIS sentences, since we could swap *big* and *large* in either sentence and retain the same meaning:

(18.13) How big is that plane?
(18.14) Would I be flying on a large or small plane?

But note the following WSJ sentence in which we cannot substitute *large* for *big*:

(18.15) Miss Nelson, for instance, became a kind of big sister to Benjamin.
(18.16) Miss Nelson, for instance, became a kind of large sister to Benjamin.

This is because the word *big* has a sense that means being older or grown up, while *large* lacks this sense. Thus, we say that some senses of *big* and *large* are (nearly) synonymous while other ones are not.

Synonyms are words with identical or similar meanings. **Antonyms**, by contrast, are words with opposite meaning such as the following:

- long/short
- big/little
- fast/slow
- cold/hot
- dark/light
- rise/fall
- up/down
- in/out

Two senses can be antonyms if they define a binary opposition or are at opposite ends of some scale. This is the case for *long/short*, *fast/slow*, or *big/little*, which are at opposite ends of the *length* or *size* scale. Another group of antonyms, **reversives**, describe change or movement in opposite directions, such as *rise/fall* or *up/down*.

Antonyms thus differ completely with respect to one aspect of their meaning—their position on a scale or their direction—but are otherwise very similar, sharing almost all other aspects of meaning. Thus, automatically distinguishing synonyms from antonyms can be difficult.

### 18.2.2 Hyponymy

One sense is a **hyponym** of another sense if the first sense is more specific, denoting a subclass of the other. For example, *car* is a hyponym of *vehicle*, *dog* is a hyponym of *animal*, and *mango* is a hyponym of *fruit*. Conversely, we say that *vehicle* is a **hypernym** of *car*, and *animal* is a hypernym of *dog*. It is unfortunate that the two words (hypernym and hyponym) are very similar and hence easily confused; for this reason, the word **superordinate** is often used instead of **hypernym**.

<table>
<thead>
<tr>
<th>Superordinate</th>
<th>vehicle</th>
<th>fruit</th>
<th>furniture</th>
<th>mammal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyponym</td>
<td>car</td>
<td>mango</td>
<td>chair</td>
<td>dog</td>
</tr>
</tbody>
</table>

We can define hypernymy more formally by saying that the class denoted by the superordinate extensionally includes the class denoted by the hyponym. Thus, the class of animals includes as members all dogs, and the class of moving actions includes all walking actions. Hypernymy can also be defined in terms of **entailment**. Under this definition, a sense *A* is a hyponym of a sense *B* if everything that is *A* is also *B*, and hence being an *A* entails being a *B*, or $\forall x \ A(x) \Rightarrow B(x)$. Hypernymy is usually a transitive relation; if *A* is a hyponym of *B* and *B* is a hyponym of *C*, then *A* is a hyponym of *C*. Another name for the hypernym/hyponym structure is the **IS-A** hierarchy, in which we say *A* IS-A *B*, or *B* subsumes *A*.

**Meronymy**  Another common relation is **meronymy**, the part-whole relation. A *leg* is part of a *chair*; a *wheel* is part of a *car*. We say that *wheel* is a **meronym** of *car*, and *car* is a **holonym** of *wheel*. 

- meronymy
- part-whole
- meronym
- holonym
18.3 WordNet: A Database of Lexical Relations

The most commonly used resource for English sense relations is the WordNet lexical database (Fellbaum, 1998). WordNet consists of three separate databases, one each for nouns and verbs and a third for adjectives and adverbs; closed class words are not included. Each database contains a set of lemmas, each one annotated with a set of senses. The WordNet 3.0 release has 117,798 nouns, 11,529 verbs, 22,479 adjectives, and 4,481 adverbs. The average noun has 1.23 senses, and the average verb has 2.16 senses. WordNet can be accessed on the Web or downloaded and accessed locally. Figure 18.1 shows the lemma entry for the noun and adjective bass.

The noun “bass” has 8 senses in WordNet.
1. bass₁ - (the lowest part of the musical range)
2. bass², bass part₁ - (the lowest part in polyphonic music)
3. bass³, basso₁ - (an adult male singer with the lowest voice)
4. sea bass¹, bass₄ - (the lean flesh of a saltwater fish of the family Serranidae)
5. freshwater bass¹, bass₅ - (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
6. bass⁶, bass voice₁, basso² - (the lowest adult male singing voice)
7. bass⁷ - (the member with the lowest range of a family of musical instruments)
8. bass⁸ - (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

The adjective “bass” has 1 sense in WordNet.
1. bass₁, deep⁶ - (having or denoting a low vocal or instrumental range)
   “a deep voice”; “a bass voice is lower than a baritone voice”;
   “a bass clarinet”

The set of near-synonyms for a WordNet sense is called a synset (for synonym set); synsets are an important primitive in WordNet. The entry for bass includes synsets like \{bass¹, deep³\}, or \{bass⁶, bass voice¹, basso²\}. We can think of a synset as representing a concept of the type we discussed in Chapter 14. Thus, instead of representing concepts in logical terms, WordNet represents them as lists of the word senses that can be used to express the concept. Here’s another synset example:

\{chump¹, fool², gull¹, mark⁹, patsy¹, fall guy¹, sucker¹, soft touch¹, mug²\}

The gloss of this synset describes it as a person who is gullible and easy to take advantage of. Each of the lexical entries included in the synset can, therefore, be used to express this concept. Synsets like this one actually constitute the senses associated with WordNet entries, and hence it is synsets, not wordforms, lemmas, or individual senses, that participate in most of the lexical sense relations in WordNet.

WordNet represents all the kinds of sense relations discussed in the previous section, as illustrated in Fig. 18.2 and Fig. 18.3. WordNet hyponymy relations cor-
respond to the notion of immediate hyponymy discussed on page 5. Each synset is related to its immediately more general and more specific synsets through direct hypernym and hyponym relations. These relations can be followed to produce longer chains of more general or more specific synsets. Figure 18.4 shows hypernym chains for \textit{bass}^3 and \textit{bass}^7.

In this depiction of hyponymy, successively more general synsets are shown on successive indented lines. The first chain starts from the concept of a human bass singer. Its immediate superordinate is a synset corresponding to the generic concept of a singer. Following this chain leads eventually to concepts such as \textit{entertainer} and \textit{person}. The second chain, which starts from musical instrument, has a completely different path leading eventually to such concepts as musical instrument, device, and physical object. Both paths do eventually join at the very abstract synset \textit{whole}, \textit{unit}, and then proceed together to \textit{entity} which is the top (root) of the noun hierarchy (in WordNet this root is generally called the \textbf{unique beginner}).

### 18.4 Word Sense Disambiguation: Overview

Our discussion of compositional semantic analyzers in Chapter 15 pretty much ignored the issue of lexical ambiguity. It should be clear by now that this is an unreasonable approach. Without some means of selecting correct senses for the words in an input, the enormous amount of homonymy and polysemy in the lexicon would quickly overwhelm any approach in an avalanche of competing interpretations.
Figure 18.4  Hyponymy chains for two separate senses of the lemma bass. Note that the chains are completely distinct, only converging at the very abstract level whole, unit.

The task of selecting the correct sense for a word is called word sense disambiguation, or WSD. Disambiguating word senses has the potential to improve many natural language processing tasks, including machine translation, question answering, and information retrieval.

WSD algorithms take as input a word in context along with a fixed inventory of potential word senses and return as output the correct word sense for that use. The input and the senses depends on the task. For machine translation from English to Spanish, the sense tag inventory for an English word might be the set of different Spanish translations. If our task is automatic indexing of medical articles, the sense-tag inventory might be the set of MeSH (Medical Subject Headings) thesaurus entries.

When we are evaluating WSD in isolation, we can use the set of senses from a dictionary/thesaurus resource like WordNet. Figure 18.4 shows an example for the word bass, which can refer to a musical instrument or a kind of fish.\(^2\)

It is useful to distinguish two variants of the generic WSD task. In the lexical sample task, a small pre-selected set of target words is chosen, along with an inventory of senses for each word from some lexicicon. Since the set of words and

\(^2\) The WordNet database includes eight senses; we have arbitrarily selected two for this example; we have also arbitrarily selected one of the many Spanish fishes that could translate English sea bass.
the set of senses are small, supervised machine learning approaches are often used to handle lexical sample tasks. For each word, a number of corpus instances (context sentences) can be selected and hand-labeled with the correct sense of the target word in each. Classifier systems can then be trained with these labeled examples. Unlabeled target words in context can then be labeled using such a trained classifier. Early work in word sense disambiguation focused solely on lexical sample tasks of this sort, building word-specific algorithms for disambiguating single words like line, interest, or plant.

In contrast, in the all-words task, systems are given entire texts and a lexicon with an inventory of senses for each entry and are required to disambiguate every content word in the text. The all-words task is similar to part-of-speech tagging, except with a much larger set of tags since each lemma has its own set. A consequence of this larger set of tags is a serious data sparseness problem; it is unlikely that adequate training data for every word in the test set will be available. Moreover, given the number of polysemous words in reasonably sized lexicons, approaches based on training one classifier per term are unlikely to be practical.

In the following sections we explore the application of various machine learning paradigms to word sense disambiguation.

18.5 Supervised Word Sense Disambiguation

If we have data that has been hand-labeled with correct word senses, we can use a supervised learning approach to the problem of sense disambiguation—extracting features from the text and training a classifier to assign the correct sense given these features. The output of training is thus a classifier system capable of assigning sense labels to unlabeled words in context.

For lexical sample tasks, there are various labeled corpora for individual words; these corpora consist of context sentences labeled with the correct sense for the target word. These include the line-hard-serve corpus containing 4,000 sense-tagged examples of line as a noun, hard as an adjective and serve as a verb (Leacock et al., 1993), and the interest corpus with 2,369 sense-tagged examples of interest as a noun (Bruce and Wiebe, 1994). The SENSEVAL project has also produced a number of such sense-labeled lexical sample corpora (SENSEVAL-1 with 34 words from the HECTOR lexicon and corpus (Kilgarriff and Rosenzweig 2000, Atkins 1993), SENSEVAL-2 and -3 with 73 and 57 target words, respectively (Palmer et al. 2001, Kilgarriff 2001).

For training all-word disambiguation tasks we use a semantic concordance, a corpus in which each open-class word in each sentence is labeled with its word sense from a specific dictionary or thesaurus. One commonly used corpus is SemCor, a subset of the Brown Corpus consisting of over 234,000 words that were man-

---

<table>
<thead>
<tr>
<th>WordNet Sense</th>
<th>Spanish Translation</th>
<th>Roget Category</th>
<th>Target Word in Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>bass</td>
<td>lubina</td>
<td>FISH/INSECT</td>
<td>. . . fish as Pacific salmon and striped bass and...</td>
</tr>
<tr>
<td>bass</td>
<td>lubina</td>
<td>FISH/INSECT</td>
<td>. . . produce filets of smoked bass or sturgeon...</td>
</tr>
<tr>
<td>bass</td>
<td>bajo</td>
<td>MUSIC</td>
<td>. . . exciting jazz bass player since Ray Brown...</td>
</tr>
<tr>
<td>bass</td>
<td>bajo</td>
<td>MUSIC</td>
<td>. . . play bass because he doesn’t have to solo...</td>
</tr>
</tbody>
</table>
usually tagged with WordNet senses (Miller et al. 1993, Landes et al. 1998). In addition, sense-tagged corpora have been built for the SENSEVAL all-word tasks. The SENSEVAL-3 English all-words test data consisted of 2081 tagged content word tokens, from 5,000 total running words of English from the WSJ and Brown corpora (Palmer et al., 2001).

The first step in supervised training is to extract features that are predictive of word senses. The insight that underlies all modern algorithms for word sense disambiguation was famously first articulated by Weaver (1955) in the context of machine translation:

> 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?”

We first perform some processing on the sentence containing the window, typically including part-of-speech tagging, lemmatization, and, in some cases, syntactic parsing to reveal headwords and dependency relations. Context features relevant to the target word can then be extracted from this enriched input. A feature vector consisting of numeric or nominal values encodes this linguistic information as an input to most machine learning algorithms.

Two classes of features are generally extracted from these neighboring contexts, both of which we have seen previously in part-of-speech tagging: collocational features and bag-of-words features. A collocation is a word or series of words in a position-specific relationship to a target word (i.e., exactly one word to the right, or the two words starting 3 words to the left, and so on). Thus, collocational features encode information about specific positions located to the left or right of the target word. Typical features extracted for these context words include the word itself, the root form of the word, and the word’s part-of-speech. Such features are effective at encoding local lexical and grammatical information that can often accurately isolate a given sense.

For example consider the ambiguous word *bass* in the following WSJ sentence:

> (18.17) An electric guitar and *bass* player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.

A collocational feature vector, extracted from a window of two words to the right and left of the target word, made up of the words themselves, their respective parts-of-speech, and pairs of words, that is,

\[
\begin{align*}
[w_{i-2}, \text{POS}_{i-2}, w_{i-1}, \text{POS}_{i-1}, w_{i+1}, \text{POS}_{i+1}, w_{i+2}, \text{POS}_{i+2}, w_{i-2}^t, w_{i+1}^t] \\
\end{align*}
\]

would yield the following vector:

\[
\begin{align*}
\text{[guitar, NN, and, CC, player, NN, stand, VB, and guitar, player stand]} \\
\end{align*}
\]

High performing systems generally use POS tags and word collocations of length 1, 2, and 3 from a window of words 3 to the left and 3 to the right (Zhong and Ng, 2010).

The second type of feature consists of bag-of-words information about neighboring words. A bag-of-words means an unordered set of words, with their exact
position ignored. The simplest bag-of-words approach represents the context of a target word by a vector of features, each binary feature indicating whether a vocabulary word \( w \) does or doesn’t occur in the context.

This vocabulary is typically pre-selected as some useful subset of words in a training corpus. In most WSD applications, the context region surrounding the target word is generally a small, symmetric, fixed-size window with the target word at the center. Bag-of-word features are effective at capturing the general topic of the discourse in which the target word has occurred. This, in turn, tends to identify senses of a word that are specific to certain domains. We generally don’t use stopwords, punctuation, or number as features, and words are lemmatized and lower-cased. In some cases we may also limit the bag-of-words to consider only frequently used words. For example, a bag-of-words vector consisting of the 12 most frequent content words from a collection of \( \textit{bass} \) sentences drawn from the WSJ corpus would have the following ordered word feature set:

\[
\text{[fishing, big, sound, player, fly, rod, pound, double, runs, playing, guitar, band]}
\]

Using these word features with a window size of 10, (18.17) would be represented by the following binary vector:

\[
[0,0,0,1,0,0,0,0,0,1,0]
\]

Given training data together with the extracted features, any supervised machine learning paradigm can be used to train a sense classifier.

### 18.5.1 Wikipedia as a source of training data

Supervised methods for WSD are very dependent on the amount of training data, especially because of their reliance on sparse lexical and collocation features. One way to increase the amount of training data is to use Wikipedia as a source of sense-labeled data. When a concept is mentioned in a Wikipedia article, the article text may contain an explicit link to the concepts’ Wikipedia page, which is named by a unique identifier. This link can be used as a sense annotation. For example, the ambiguous word \( \textit{bar} \) is linked to a different Wikipedia article depending on its meaning in context, including the page \( \textit{BAR (LAW)} \), the page \( \textit{BAR (MUSIC)} \), and so on, as in the following Wikipedia examples (Mihalcea, 2007).

In 1834, Sumner was admitted to the \([\textit{bar (law)}]/\textit{bar}]\) at the age of twenty-three, and entered private practice in Boston.

It is danced in 3/4 time (like most waltzes), with the couple turning approx. 180 degrees every \([\textit{bar (music)}]/\textit{bar}]\).

Jenga is a popular beer in the \([\textit{bar (establishment)}]/\textit{bar}]\)s of Thailand.

These sentences can then be added to the training data for a supervised system. In order to use Wikipedia in this way, however, it is necessary to map from Wikipedia concepts to whatever inventory of senses is relevant for the WSD application. Automatic algorithms that map from Wikipedia to WordNet, for example, involve finding the WordNet sense that has the greatest lexical overlap with the Wikipedia sense, by comparing the vector of words in the WordNet synset, gloss, and related senses with the vector of words in the Wikipedia page title, outgoing links, and page category (Ponzetto andNavigli, 2010).

### 18.5.2 Evaluation

To evaluate WSD algorithms, it’s better to consider \textit{extrinsic}, \textit{task-based}, or \textit{end-}
to-end evaluation, in which we see whether some new WSD idea actually improves performance in some end-to-end application like question answering or machine translation. Nonetheless, because extrinsic evaluations are difficult and slow, WSD systems are typically evaluated with intrinsic evaluation, in which a WSD component is treated as an independent system. Common intrinsic evaluations are either exact-match sense accuracy—the percentage of words that are tagged identically with the hand-labeled sense tags in a test set—or with precision and recall if systems are permitted to pass on the labeling of some instances. In general, we evaluate by using held-out data from the same sense-tagged corpora that we used for training, such as the SemCor corpus discussed above or the various corpora produced by the SENSEVAL effort.

Many aspects of sense evaluation have been standardized by the SENSEVAL and SEMEVAL efforts (Palmer et al. 2006, Kilgarriff and Palmer 2000). This framework provides a shared task with training and testing materials along with sense inventories for all-words and lexical sample tasks in a variety of languages.

The normal baseline is to choose the most frequent sense for each word from the senses in a labeled corpus (Gale et al., 1992a). For WordNet, this corresponds to the first sense, since senses in WordNet are generally ordered from most frequent to least frequent. WordNet sense frequencies come from the SemCor sense-tagged corpus described above– WordNet senses that don’t occur in SemCor are ordered arbitrarily after those that do. The most frequent sense baseline can be quite accurate, and is therefore often used as a default, to supply a word sense when a supervised algorithm has insufficient training data.

18.6 WSD: Dictionary and Thesaurus Methods

Supervised algorithms based on sense-labeled corpora are the best-performing algorithms for sense disambiguation. However, such labeled training data is expensive and limited. One alternative is to get indirect supervision from dictionaries and thesauruses or similar knowledge bases and so this method is also called knowledge-based WSD. Methods like this that do not use texts that have been hand-labeled with senses are also called weakly supervised.

18.6.1 The Lesk Algorithm

The most well-studied dictionary-based algorithm for sense disambiguation is the Lesk algorithm, really a family of algorithms that choose the sense whose dictionary gloss or definition shares the most words with the target word’s neighborhood. Figure 18.6 shows the simplest version of the algorithm, often called the Simplified Lesk algorithm (Kilgarriff and Rosenzweig, 2000).

As an example of the Lesk algorithm at work, consider disambiguating the word bank in the following context:

(18.19) The bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

given the following two WordNet senses:
function SIMPLIFIED LESK(word, sentence) returns best sense of word

best-sense ← most frequent sense for word
max-overlap ← 0
context ← set of words in sentence

for each sense in senses of word do
  signature ← set of words in the gloss and examples of sense
  overlap ← COMPUTE_OVERLAP(signature, context)
  if overlap > max-overlap then
    max-overlap ← overlap
    best-sense ← sense
  end
return(best-sense)

Figure 18.6  The Simplified Lesk algorithm. The COMPUTE_OVERLAP function returns the number of words in common between two sets, ignoring function words or other words on a stop list. The original Lesk algorithm defines the context in a more complex way. The Corpus Lesk algorithm weights each overlapping word \( w \) by its \( -\log P(w) \) and includes labeled training corpus data in the signature.

| bank₁ | Gloss: a financial institution that accepts deposits and channels the money into lending activities |
|       | Examples: “he cashed a check at the bank”, “that bank holds the mortgage on my home” |
| bank₂ | Gloss: sloping land (especially the slope beside a body of water) |
|       | Examples: “they pulled the canoe up on the bank”, “he sat on the bank of the river and watched the currents” |

Sense bank₁ has two non-stopwords overlapping with the context in (18.19): deposits and mortgage, while sense bank₂ has zero words, so sense bank₁ is chosen.

There are many obvious extensions to Simplified Lesk. The original Lesk algorithm (Lesk, 1986) is slightly more indirect. Instead of comparing a target word’s signature with the context words, the target signature is compared with the signatures of each of the context words. For example, consider Lesk’s example of selecting the appropriate sense of cone in the phrase pine cone given the following definitions for pine and cone.

| pine | 1 kinds of evergreen tree with needle-shaped leaves | 2 waste away through sorrow or illness |
| cone | 1 solid body which narrows to a point | 2 something of this shape whether solid or hollow | 3 fruit of certain evergreen trees |

In this example, Lesk’s method would select cone³ as the correct sense since two of the words in its entry, evergreen and tree, overlap with words in the entry for pine, whereas neither of the other entries has any overlap with words in the definition of pine. In general Simplified Lesk seems to work better than original Lesk.

The primary problem with either the original or simplified approaches, however, is that the dictionary entries for the target words are short and may not provide enough chance of overlap with the context.³ One remedy is to expand the list of words used in the classifier to include words related to, but not contained in, their

³ Indeed, Lesk (1986) notes that the performance of his system seems to roughly correlate with the length of the dictionary entries.
individual sense definitions. But the best solution, if any sense-tagged corpus data like SemCor is available, is to add all the words in the labeled corpus sentences for a word sense into the signature for that sense. This version of the algorithm, the Corpus Lesk algorithm, is the best-performing of all the Lesk variants (Kilgarriff and Rosenzweig 2000, Vasilescu et al. 2004) and is used as a baseline in the SENSEVAL competitions. Instead of just counting up the overlapping words, the Corpus Lesk algorithm also applies a weight to each overlapping word. The weight is the inverse document frequency or IDF, a standard information-retrieval measure introduced in Chapter 17. IDF measures how many different “documents” (in this case, glosses and examples) a word occurs in and is thus a way of discounting function words. Since function words like the, of, etc., occur in many documents, their IDF is very low, while the IDF of content words is high. Corpus Lesk thus uses IDF instead of a stop list.

Formally, the IDF for a word \( i \) can be defined as

\[
\text{idf}_i = \log \left( \frac{N_{doc}}{n_{d_i}} \right)
\]  

(18.20)

where \( N_{doc} \) is the total number of “documents” (glosses and examples) and \( n_{d_i} \) is the number of these documents containing word \( i \).

Finally, we can combine the Lesk and supervised approaches by adding new Lesk-like bag-of-words features. For example, the glosses and example sentences for the target sense in WordNet could be used to compute the supervised bag-of-words features in addition to the words in the SemCor context sentence for the sense (Yuret, 2004).

18.6.2 Graph-based Methods

Another way to use a thesaurus like WordNet is to make use of the fact that WordNet can be construed as a graph, with senses as nodes and relations between senses as edges. In addition to the hyponymy and other relations, it’s possible to create links between senses and those words in the gloss that are unambiguous (have only one sense). Often the relations are treated as undirected edges, creating a large undirected WordNet graph. Fig. 18.7 shows a portion of the graph around the word \( \text{drink}_1 \).

There are various ways to use the graph for disambiguation, some using the whole graph, some using only a subpart. For example the target word and the words in its sentential context sentence can all be inserted as nodes in the graph via a directed edge to each of its senses. If we consider the sentence \( \text{She drank some milk} \),
Fig. 18.8 shows a portion of the WordNet graph between the senses for between drink\(^1\) and milk\(^1\).

![WordNet graph](diagram)

Figure 18.8 Part of the WordNet graph between drink\(^1\) and milk\(^1\), for disambiguating a sentence like *She drank some milk*, adapted from Navigli and Lapata (2010).

The correct sense is then the one which is the most important or *central* in some way in this graph. There are many different methods for deciding centrality. The simplest is *degree*, the number of edges into the node, which tends to correlate with the most frequent sense. Another algorithm for assigning probabilities across nodes is *personalized page rank*, a version of the well-known pagerank algorithm which uses some seed nodes. By inserting a uniform probability across the word nodes (*drink* and *milk* in the example) and computing the personalized page rank of the graph, the result will be a pagerank value for each node in the graph, and the sense with the maximum pagerank can then be chosen. See Agirre et al. (2014) and Navigli and Lapata (2010) for details.

### 18.7 Semi-Supervised WSD: Bootstrapping

Both the supervised approach and the dictionary-based approaches to WSD require large hand-built resources: supervised training sets in one case, large dictionaries in the other. We can instead use *bootstrapping* or *semi-supervised learning*, which needs only a very small hand-labeled training set.

A classic bootstrapping algorithm for WSD is the *Yarowsky algorithm* for learning a classifier for a target word (in a lexical-sample task) (Yarowsky, 1995). The algorithm is given a small seedset \( \Lambda_0 \) of labeled instances of each sense and a much larger unlabeled corpus \( V_0 \). The algorithm first trains an initial classifier on the seedset \( \Lambda_0 \). It then uses this classifier to label the unlabeled corpus \( V_0 \). The algorithm then selects the examples in \( V_0 \) that it is most confident about, removes them, and adds them to the training set (call it now \( \Lambda_1 \)). The algorithm then trains a new classifier (a new set of rules) on \( \Lambda_1 \), and iterates by applying the classifier to the now-smaller unlabeled set \( V_1 \), extracting a new training set \( \Lambda_2 \), and so on. With each iteration of this process, the training corpus grows and the untagged corpus shrinks. The process is repeated until some sufficiently low error-rate on the training set is reached or until no further examples from the untagged corpus are above threshold.

Initial seeds can be selected by hand-labeling a small set of examples (Hearst, 1991), or by using the help of a heuristic. Yarowsky (1995) used the *one sense per collocation* heuristic, which relies on the intuition that certain words or phrases...
We need more good teachers – right now, there are only a half a dozen who can play the free bass with ease.

An electric guitar and bass player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.

The researchers said the worms spend part of their life cycle in such fish as Pacific salmon and striped bass and Pacific rockfish or snapper.

And it all started when fishermen decided the striped bass in Lake Mead were too skinny.

We need more good teachers – right now, there are only a half a dozen who can play the free bass with ease.

An electric guitar and bass player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.

The researchers said the worms spend part of their life cycle in such fish as Pacific salmon and striped bass and Pacific rockfish or snapper.

And it all started when fishermen decided the striped bass in Lake Mead were too skinny.

Figure 18.10 Samples of bass sentences extracted from the WSJ by using the simple correlates play and fish.

strongly associated with the target senses tend not to occur with the other sense. Yarowsky defines his seedset by choosing a single collocation for each sense.

For example, to generate seed sentences for the fish and musical musical senses of bass, we might come up with fish as a reasonable indicator of bass1 and play as a reasonable indicator of bass2. Figure 18.10 shows a partial result of such a search for the strings “fish” and “play” in a corpus of bass examples drawn from the WSJ.

The original Yarowsky algorithm also makes use of a second heuristic, called one sense per discourse, based on the work of Gale et al. (1992b), who noticed that a particular word appearing multiple times in a text or discourse often appeared with the same sense. This heuristic seems to hold better for coarse-grained senses and particularly for cases of homonymy rather than polysemy (Krovetz, 1998).

Nonetheless, it is still useful in a number of sense disambiguation situations. In fact, the one sense per discourse heuristic is an important one throughout language processing as it seems that many disambiguation tasks may be improved by a bias toward resolving an ambiguity the same way inside a discourse segment.
18.8 Unsupervised Word Sense Induction

It is expensive and difficult to build large corpora in which each word is labeled for its word sense. For this reason, an unsupervised approach to sense disambiguation, often called word sense induction or WSI, is an exciting and important research area. In unsupervised approaches, we don’t use human-defined word senses. Instead, the set of “senses” of each word is created automatically from the instances of each word in the training set.

Most algorithms for word sense induction use some sort of clustering. For example, the early algorithm of Schütze (Schütze 1992, Schütze 1998) represented each word as a context vector of bag-of-words features \( \vec{c} \). (See Chapter 17 for a more complete introduction to such vector models of meaning.) Then in training, we use three steps.

1. For each token \( w_i \) of word \( w \) in a corpus, compute a context vector \( \vec{c} \).
2. Use a clustering algorithm to cluster these word-token context vectors \( \vec{c} \) into a predefined number of groups or clusters. Each cluster defines a sense of \( w \).
3. Compute the vector centroid of each cluster. Each vector centroid \( \vec{s}_j \) is a sense vector representing that sense of \( w \).

Since this is an unsupervised algorithm, we don’t have names for each of these “senses” of \( w \); we just refer to the \( j \)th sense of \( w \).

Now how do we disambiguate a particular token \( t \) of \( w \)? Again, we have three steps:

1. Compute a context vector \( \vec{c} \) for \( t \).
2. Retrieve all sense vectors \( \vec{s}_j \) for \( w \).
3. Assign \( t \) to the sense represented by the sense vector \( \vec{s}_j \) that is closest to \( t \).

All we need is a clustering algorithm and a distance metric between vectors. Clustering is a well-studied problem with a wide number of standard algorithms that can be applied to inputs structured as vectors of numerical values (Duda and Hart, 1973). A frequently used technique in language applications is known as agglomerative clustering. In this technique, each of the \( N \) training instances is initially assigned to its own cluster. New clusters are then formed in a bottom-up fashion by the successive merging of the two clusters that are most similar. This process continues until either a specified number of clusters is reached, or some global goodness measure among the clusters is achieved. In cases in which the number of training instances makes this method too expensive, random sampling can be used on the original training set to achieve similar results.

Recent algorithms have also used topic modeling algorithms like Latent Dirichlet Allocation (LDA), another way to learn clusters of words based on their distributions (Lau et al., 2012).

How can we evaluate unsupervised sense disambiguation approaches? As usual, the best way is to do extrinsic evaluation embedded in some end-to-end system; one example used in a SemEval bakeoff is to improve search result clustering and diversification (Navigli and Vannella, 2013). Intrinsic evaluation requires a way to map the automatically derived sense classes into a hand-labeled gold-standard set so that we can compare a hand-labeled test set with a set labeled by our unsupervised classifier. Various such metrics have been tested, for example in the SemEval tasks (Manandhar et al. 2010, Navigli and Vannella 2013, Jurgens and Klapaftis 2013),
including cluster overlap metrics, or methods that map each sense cluster to a pre-defined sense by choosing the sense that (in some training set) has the most overlap with the cluster. However it is fair to say that no evaluation metric for this task has yet become standard.

### 18.9 Word Similarity: Thesaurus Methods

We turn now to the computation of various semantic relations that hold between words. We saw in Section 18.2 that such relations include synonymy, antonymy, hyponymy, hypernymy, and meronymy. Of these, the one that has been most computationally developed and has the greatest number of applications is the idea of word **synonymy** and **similarity**.

Synonymy is a binary relation between words; two words are either synonyms or not. For most computational purposes, we use instead a looser metric of word **similarity** or **semantic distance**. Two words are more similar if they share more features of meaning or are near-synonyms. Two words are less similar or have greater semantic distance, if they have fewer common meaning elements. Although we have described them as relations between words, synonymy, similarity, and distance are actually relations between word **senses**. For example, of the two senses of *bank*, we might say that the financial sense is similar to one of the senses of *fund* and the riparian sense is more similar to one of the senses of *slope*. In the next few sections of this chapter, we will compute these relations over both words and senses.

The ability to compute word similarity is a useful part of many language understanding applications. In information retrieval or question answering, we might want to retrieve documents whose words have meanings similar to the query words. In summarization, generation, and machine translation, we need to know whether two words are similar to know if we can substitute one for the other in particular contexts. In language modeling, we can use semantic similarity to cluster words for class-based models. One interesting class of applications for word similarity is automatic grading of student responses. For example, algorithms for automatic essay grading use word similarity to determine if an essay is similar in meaning to a correct answer. We can also use word similarity as part of an algorithm to take an exam, such as a multiple-choice vocabulary test. Automatically taking exams is useful in test designs in order to see how easy or hard a particular multiple-choice question or exam is.

Two classes of algorithms can be used to measure word similarity. This chapter focuses on thesaurus-based algorithms, in which we measure the distance between two senses in an on-line thesaurus like WordNet or MeSH. The next chapter focuses on distributional algorithms, in which we estimate word similarity by finding words that have similar distributions in a corpus.

The thesaurus-based algorithms use the structure of the thesaurus to define word similarity. In principle, we could measure similarity by using any information available in a thesaurus (meronymy, glosses, etc.). In practice, however, thesaurus-based word similarity algorithms generally use only the hypernym/hyponym (is-a or subsumption) hierarchy. In WordNet, verbs and nouns are in separate hypernym hierarchies, so a thesaurus-based algorithm for WordNet can thus compute only noun-noun similarity, or verb-verb similarity; we can’t compare nouns to verbs or do anything with adjectives or other parts of speech.

We can distinguish **word similarity** from **word relatedness**. Two words are
similar if they are near-synonyms or roughly substitutable in context. Word relatedness characterizes a larger set of potential relationships between words; antonyms, for example, have high relatedness but low similarity. The words car and gasoline are closely related but not similar, while car and bicycle are similar. Word similarity is thus a subcase of word relatedness. In general, the five algorithms we describe in this section do not attempt to distinguish between similarity and semantic relatedness; for convenience, we will call them similarity measures, although some would be more appropriately described as relatedness measures; we return to this question in Section ??.

Figure 18.11 A fragment of the WordNet hypernym hierarchy, showing path lengths (number of edges plus 1) from nickel to coin (2), dime (3), money (6), and Richter scale (8).

The simplest thesaurus-based algorithms are based on the intuition that words or senses are more similar if there is a shorter path between them in the thesaurus graph, an intuition dating back to Quillian (1969). A word/sense is most similar to itself, then to its parents or siblings, and least similar to words that are far away. We make this notion operational by measuring the number of edges between the two concept nodes in the thesaurus graph and adding one. Figure 18.11 shows an intuition; the concept dime is most similar to nickel and coin, less similar to money, and even less similar to Richter scale. A formal definition:

\[
\text{pathlen}(c_1, c_2) = 1 + \text{the number of edges in the shortest path in the thesaurus graph between the sense nodes } c_1 \text{ and } c_2
\]

Path-based similarity can be defined as just the path length, transformed either by log (Leacock and Chodorow, 1998) or, more often, by an inverse, resulting in the following common definition of path-length based similarity:

\[
\text{sim}_{\text{path}}(c_1, c_2) = \frac{1}{\text{pathlen}(c_1, c_2)}
\]  

(18.21)

For most applications, we don’t have sense-tagged data, and thus we need our algorithm to give us the similarity between words rather than between senses or concepts. For any of the thesaurus-based algorithms, following Resnik (1995), we can approximate the correct similarity (which would require sense disambiguation) by just using the pair of senses for the two words that results in maximum sense similarity. Thus, based on sense similarity, we can define word similarity as follows:

\[
\text{wordsim}(w_1, w_2) = \max_{c_1 \in \text{Senses}(w_1)} \max_{c_2 \in \text{Senses}(w_2)} \text{sim}(c_1, c_2)
\]

(18.22)
Chapter 18 • Computing with Word Senses

The basic path-length algorithm makes the implicit assumption that each link in the network represents a uniform distance. In practice, this assumption is not appropriate. Some links (e.g., those that are deep in the WordNet hierarchy) often seem to represent an intuitively narrow distance, while other links (e.g., higher up in the WordNet hierarchy) represent an intuitively wider distance. For example, in Fig. 18.11, the distance from *nickel* to *money* (5) seems intuitively much shorter than the distance from *nickel* to an abstract word *standard*; the link between *medium of exchange* and *standard* seems wider than that between, say, *coin* and *coinage*.

It is possible to refine path-based algorithms with normalizations based on depth in the hierarchy (Wu and Palmer, 1994), but in general we’d like an approach that lets us independently represent the distance associated with each edge.

A second class of thesaurus-based similarity algorithms attempts to offer just such a fine-grained metric. These information-content word-similarity algorithms still rely on the structure of the thesaurus but also add probabilistic information derived from a corpus.

Following Resnik (1995) we’ll define \( P(c) \) as the probability that a randomly selected word in a corpus is an instance of concept \( c \) (i.e., a separate random variable, ranging over words, associated with each concept). This implies that \( P(root) = 1 \) since any word is subsumed by the root concept. Intuitively, the lower a concept in the hierarchy, the lower its probability. We train these probabilities by counting in a corpus; each word in the corpus counts as an occurrence of each concept that contains it. For example, in Fig. 18.11 above, an occurrence of the word *dime* would count toward the frequency of *coin*, *currency*, *standard*, etc. More formally, Resnik computes \( P(c) \) as follows:

\[
P(c) = \frac{\sum_{w \in \text{words}(c)} \text{count}(w)}{N}
\]  

(18.23)

where \( \text{words}(c) \) is the set of words subsumed by concept \( c \), and \( N \) is the total number of words in the corpus that are also present in the thesaurus.

Figure 18.12, from Lin (1998), shows a fragment of the WordNet concept hierarchy augmented with the probabilities \( P(c) \).

<table>
<thead>
<tr>
<th>entity 0.395</th>
</tr>
</thead>
<tbody>
<tr>
<td>inanimate-object 0.167</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>natural-object 0.0163</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>geological-formation 0.00176</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>natural-elevation 0.000113 shore 0.0000836</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>0.000189 hill coast 0.0000216</td>
</tr>
</tbody>
</table>

Figure 18.12 A fragment of the WordNet hierarchy, showing the probability \( P(c) \) attached to each content, adapted from a figure from Lin (1998).

We now need two additional definitions. First, following basic information theory, we define the information content (IC) of a concept \( c \) as

\[
\text{IC}(c) = -\log P(c)
\]  

(18.24)
Second, we define the **lowest common subsumer** or LCS of two concepts:

\[
\text{LCS}(c_1, c_2) = \text{the lowest common subsumer, that is, the lowest node in the hierarchy that subsumes (is a hypernym of) both } c_1 \text{ and } c_2.
\]

There are now a number of ways to use the information content of a node in a word similarity metric. The simplest way was first proposed by Resnik (1995). We think of the similarity between two words as related to their common information; the more two words have in common, the more similar they are. Resnik proposes to estimate the common amount of information by the **information content of the lowest common subsumer of the two nodes**. More formally, the **Resnik similarity** measure is

\[
\text{sim}_{\text{resnik}}(c_1, c_2) = -\log P(\text{LCS}(c_1, c_2)) \tag{18.25}
\]

Lin (1998) extended the Resnik intuition by pointing out that a similarity metric between objects A and B needs to do more than measure the amount of information in common between A and B. For example, he additionally pointed out that the more **differences** between A and B, the less similar they are. In summary:

- **Commonality**: the more information A and B have in common, the more similar they are.
- **Difference**: the more differences between the information in A and B, the less similar they are.

Lin measures the commonality between A and B as the information content of the proposition that states the commonality between A and B:

\[
\text{IC}(\text{common}(A,B)) \tag{18.26}
\]

He measures the difference between A and B as

\[
\text{IC}(\text{description}(A,B)) - \text{IC}(\text{common}(A,B)) \tag{18.27}
\]

where description(A,B) describes A and B. Given a few additional assumptions about similarity, Lin proves the following theorem:

**Similarity Theorem**: The similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe what A and B are.

\[
\text{sim}_{\text{Lin}}(A,B) = \frac{\text{common}(A,B)}{\text{description}(A,B)} \tag{18.28}
\]

Applying this idea to the thesaurus domain, Lin shows (in a slight modification of Resnik’s assumption) that the information in common between two concepts is twice the information in the lowest common subsumer LCS\((c_1, c_2)\). Adding in the above definitions of the information content of thesaurus concepts, the final **Lin similarity** function is

\[
\text{sim}_{\text{Lin}}(c_1, c_2) = \frac{2 \times \log P(\text{LCS}(c_1, c_2))}{\log P(c_1) + \log P(c_2)} \tag{18.29}
\]

For example, using sim\(_{\text{Lin}}\), Lin (1998) shows that the similarity between the concepts of *hill* and *coast* from Fig. 18.12 is
A similar formula, **Jiang-Conrath distance** (Jiang and Conrath, 1997), although derived in a completely different way from Lin and expressed as a distance rather than similarity function, has been shown to work as well as or better than all the other thesaurus-based methods:

\[
dist_{JC}(c_1, c_2) = 2 \times \log P(LCS(c_1, c_2)) - (\log P(c_1) + \log P(c_2)) \quad (18.31)
\]

We can transform \( \text{dist}_{JC} \) into a similarity by taking the reciprocal.

Finally, we describe a **dictionary-based** method, an extension of the Lesk algorithm for word sense disambiguation described in Section 18.6.1. We call this a dictionary rather than a thesaurus method because it makes use of glosses, which are, in general, a property of dictionaries rather than thesauruses (although WordNet does have glosses). Like the Lesk algorithm, the intuition of this **extended gloss overlap**, or **Extended Lesk** measure (Banerjee and Pedersen, 2003) is that two concepts/senses in a thesaurus are similar if their glosses contain overlapping words. We’ll begin by sketching an overlap function for two glosses. Consider these two concepts, with their glosses:

- **drawing paper**: paper that is specially prepared for use in drafting
- **decal**: the art of transferring designs from specially prepared paper to a wood or glass or metal surface.

For each \( n \)-word phrase that occurs in both glosses, Extended Lesk adds in a score of \( n^2 \) (the relation is non-linear because of the Zipfian relationship between lengths of phrases and their corpus frequencies; longer overlaps are rare, so they should be weighted more heavily). Here, the overlapping phrases are **paper** and **specially prepared**, for a total similarity score of \( 1^2 + 2^2 = 5 \).

Given such an overlap function, when comparing two concepts (synsets), Extended Lesk not only looks for overlap between their glosses but also between the glosses of the senses that are hypernyms, hyponyms, meronyms, and other relations of the two concepts. For example, if we just considered hyponyms and defined \( \text{gloss}(\text{hypo}(A)) \) as the concatenation of all the glosses of all the hyponym senses of \( A \), the total relatedness between two concepts \( A \) and \( B \) might be

\[
similarity(A,B) = \text{overlap}(\text{gloss}(A), \text{gloss}(B)) + \text{overlap}(\text{gloss}(\text{hypo}(A)), \text{gloss}(\text{hypo}(B))) + \text{overlap}(\text{gloss}(A), \text{gloss}(\text{hypo}(B))) + \text{overlap}(\text{gloss}(\text{hypo}(A)), \text{gloss}(B))
\]

Let \( \text{RELS} \) be the set of possible WordNet relations whose glosses we compare; assuming a basic overlap measure as sketched above, we can then define the **Extended Lesk** overlap measure as

\[
sim_{eLesk}(c_1, c_2) = \sum_{r,q \in \text{RELS}} \text{overlap}(\text{gloss}(r(c_1)), \text{gloss}(q(c_2))) \quad (18.32)
\]

Figure 18.13 summarizes the five similarity measures we have described in this section.
Evaluating Thesaurus-Based Similarity

Which of these similarity measures is best? Word similarity measures have been evaluated in two ways. The most common intrinsic evaluation metric computes the correlation coefficient between an algorithm’s word similarity scores and word similarity ratings assigned by humans. There are a variety of such human-labeled datasets: the RG-65 dataset of human similarity ratings on 65 word pairs (Rubenstein and Goodenough, 1965), the MC-30 dataset of 30 word pairs (Miller and Charles, 1991). The WordSim-353 (Finkelstein et al., 2002) is a commonly used set of of ratings from 0 to 10 for 353 noun pairs; for example (plane, car) had an average score of 5.77. SimLex-999 (Hill et al., 2015) is a more difficult dataset that quantifies similarity (cup, mug) rather than relatedness (cup, coffee), and including both concrete and abstract adjective, noun and verb pairs. Another common intrinsic similarity measure is the TOEFL dataset, a set of 80 questions, each consisting of a target word with 4 additional word choices; the task is to choose which is the correct synonym, as in the example: *Levied is closest in meaning to:* imposed, believed, requested, correlated (Landauer and Dumais, 1997). All of these datasets present words without context.

Slightly more realistic are intrinsic similarity tasks that include context. The Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012) offers a richer evaluation scenario, giving human judgments on 2,003 pairs of words in their sentential context, including nouns, verbs, and adjectives. This dataset enables the evaluation of word similarity algorithms that can make use of context words. The *semantic textual similarity* task (Agirre et al. 2012, Agirre et al. 2015) evaluates the performance of sentence-level similarity algorithms, consisting of a set of pairs of sentences, each pair with human-labeled similarity scores.

Alternatively, the similarity measure can be embedded in some end-application, such as question answering (Surdeanu et al., 2011), spell-checking (Jones and Martin 1997, Budanitsky and Hirst 2006, Hirst and Budanitsky 2005), web search result clustering (Di Marco andNavigli, 2013), or text simplification (Biran et al., 2011), and different measures can be evaluated by how much they improve the end application.

We’ll return to evaluation metrics in the next chapter when we consider distributional semantics and similarity.
18.10 Summary

This chapter has covered a wide range of issues concerning the meanings associated with lexical items. The following are among the highlights:

- **Lexical semantics** is the study of the meaning of words and the systematic meaning-related connections between words.
- A **word sense** is the locus of word meaning; definitions and meaning relations are defined at the level of the word sense rather than wordforms.
- **Homonymy** is the relation between unrelated senses that share a form, and **polysemy** is the relation between related senses that share a form.
- **Synonymy** holds between different words with the same meaning.
- **Hyponymy** and **hypernymy** relations hold between words that are in a class-inclusion relationship.
- **WordNet** is a large database of lexical relations for English.
- **Word-sense disambiguation (WSD)** is the task of determining the correct sense of a word in context. Supervised approaches make use of sentences in which individual words (**lexical sample task**) or all words (**all-words task**) are hand-labeled with senses from a resource like WordNet. Classifiers for supervised WSD are generally trained on collocational and bag-of-words features that describe the surrounding words.
- An important baseline for WSD is the **most frequent sense**, equivalent, in WordNet, to **take the first sense**.
- The **Lesk algorithm** chooses the sense whose dictionary definition shares the most words with the target word’s neighborhood.
- Graph-based algorithms view the thesaurus as a graph and choose the sense that is most central in some way.
- **Word similarity** can be computed by measuring the **link distance** in a thesaurus or by various measure of the **information content** of the two nodes.

Bibliographical and Historical Notes

Word sense disambiguation traces its roots to some of the earliest applications of digital computers. We saw above Warren Weaver’s (1955) suggestion to disambiguate a word by looking at a small window around it, in the context of machine translation. Other notions first proposed in this early period include the use of a thesaurus for disambiguation (Masterman, 1957), supervised training of Bayesian models for disambiguation (Madhu and Lytel, 1965), and the use of clustering in word sense analysis (Sparck Jones, 1986).

An enormous amount of work on disambiguation was conducted within the context of early AI-oriented natural language processing systems. Quillian (1968) and Quillian (1969) proposed a graph-based approach to language understanding, in which the dictionary definition of words was represented by a network of word nodes connected by syntactic and semantic relations. He then proposed to do sense disambiguation by finding the shortest path between senses in the conceptual graph. Simmons (1973) is another influential early semantic network approach. Wilks proposed
one of the earliest non-discrete models with his *Preference Semantics* (Wilks 1975c, Wilks 1975b, Wilks 1975a), and Small and Rieger (1982) and Riesbeck (1975) proposed understanding systems based on modeling rich procedural information for each word. Hirst’s *ABSITY* system (Hirst and Charniak 1982, Hirst 1987, Hirst 1988), which used a technique called marker passing based on semantic networks, represents the most advanced system of this type. As with these largely symbolic approaches, early neural network (often called ‘connectionist’) approaches to word sense disambiguation relied on small lexicons with hand-coded representations (Cottrell 1985, Kawamoto 1988).

Considerable work on sense disambiguation has been conducted in the areas of cognitive science and psycholinguistics. Appropriately enough, this work is generally described by a different name: lexical ambiguity resolution. Small et al. (1988) present a variety of papers from this perspective.

The earliest implementation of a robust empirical approach to sense disambiguation is due to Kelly and Stone (1975), who directed a team that hand-crafted a set of disambiguation rules for 1790 ambiguous English words. Lesk (1986) was the first to use a machine-readable dictionary for word sense disambiguation. The problem of dictionary senses being too fine-grained or lacking an appropriate organization has been addressed with models of clustering word senses (Dolan 1994, Chen and Chang 1998, Mihalcea and Moldovan 2001, Agirre and de Lacalle 2003, Chklovski and Mihalcea 2003, Palmer et al. 2004,Navigli 2006, Snow et al. 2007). Clustered senses are often called *coarse senses*. Corpora with clustered word senses for training clustering algorithms include Palmer et al. (2006) and *OntoNotes* (Hovy et al., 2006).

Modern interest in supervised machine learning approaches to disambiguation began with Black (1988), who applied decision tree learning to the task. The need for large amounts of annotated text in these methods led to investigations into the use of bootstrapping methods (Hearst 1991, Yarowsky 1995).

Diab and Resnik (2002) give a semi-supervised algorithm for sense disambiguation based on aligned parallel corpora in two languages. For example, the fact that the French word *catastrophe* might be translated as English *disaster* in one instance and *tragedy* in another instance can be used to disambiguate the senses of the two English words (i.e., to choose senses of *disaster* and *tragedy* that are similar). Abney (2002) and Abney (2004) explore the mathematical foundations of the Yarowsky algorithm and its relation to co-training. The most-frequent-sense heuristic is an extremely powerful one but requires large amounts of supervised training data.

The earliest use of clustering in the study of word senses was by Sparck Jones (1986). Zernik (1991) applied a standard information retrieval clustering algorithm to the problem and evaluated it according to improvements in retrieval performance and Pedersen and Bruce (1997), Schütze (1997), and Schütze (1998) applied distributional methods. Recent work on word sense induction has applied Latent Dirichelet Allocation (LDA) (Boyd-Graber et al. 2007, Brody and Lapata 2009, Lau et al. 2012), and large co-occurrence graphs (Di Marco and Navigli, 2013).

Cruse (2004) is a useful introductory linguistic text on lexical semantics. A collection of work concerning WordNet can be found in Fellbaum (1998). Many efforts have been made to use existing dictionaries as lexical resources. One of the earliest was Amsler’s (1981) use of the Merriam Webster dictionary. The machine-readable version of Longman’s *Dictionary of Contemporary English* has also been used (Boguraev and Briscoe, 1989).

Navigli (2009) is a comprehensive survey article on WSD, Agirre and Edmonds...

See Pustejovsky (1995), Pustejovsky and Boguraev (1996), Martin (1986), and Copestake and Briscoe (1995), inter alia, for computational approaches to the representation of polysemy. Pustejovsky’s theory of the generative lexicon, and in particular his theory of the qualia structure of words, is another way of accounting for the dynamic systematic polysemy of words in context.

Another important recent direction is the addition of sentiment and connotation to knowledge bases (Wiebe et al. 2005, Qiu et al. 2009, Velikovich et al. 2010) including SentiWordNet (Baccianella et al., 2010) and ConnotationWordNet (Kang et al., 2014).

Exercises

18.1 Collect a small corpus of example sentences of varying lengths from any newspaper or magazine. Using WordNet or any standard dictionary, determine how many senses there are for each of the open-class words in each sentence. How many distinct combinations of senses are there for each sentence? How does this number seem to vary with sentence length?

18.2 Using WordNet or a standard reference dictionary, tag each open-class word in your corpus with its correct tag. Was choosing the correct sense always a straightforward task? Report on any difficulties you encountered.

18.3 Using your favorite dictionary, simulate the original Lesk word overlap disambiguation algorithm described on page 13 on the phrase *Time flies like an arrow*. Assume that the words are to be disambiguated one at a time, from left to right, and that the results from earlier decisions are used later in the process.

18.4 Build an implementation of your solution to the previous exercise. Using WordNet, implement the original Lesk word overlap disambiguation algorithm described on page 13 on the phrase *Time flies like an arrow*. 


