5

PART-OF-SPEECH

Conjunction Junction, what’s your function?
Bob Dorough, Schoolhouse Rock, 1973

There are ten parts of speech, and they are all troublesome.
Mark Twain, The Awful German Language

Dionysius Thrax of Alexandria (c. 100 B.C.), or perhaps someone else (exact authorship being understandably difficult to be sure of with texts of this vintage), wrote a grammatical sketch of Greek (a “technē”) which summarized the linguistic knowledge of his day. This work is the direct source of an astonishing proportion of our modern linguistic vocabulary, including among many other words, syntax, diphthong, elicit, and analogy. Also included are a description of eight parts-of-speech: noun, verb, pronoun, preposition, adverb, conjunction, participle, and article. Although earlier scholars (including Aristotle as well as the Stoics) had their own lists of parts-of-speech, it was Thrax’s set of eight which became the basis for practically all subsequent part-of-speech descriptions of Greek, Latin, and most European languages for the next 2000 years.

Schoolhouse Rock was a popular series of 3-minute musical animated clips first aired on television in 1973. The series was designed to inspire kids to learn multiplication tables, grammar, and basic science and history. The Grammar Rock sequence, for example, included songs about parts-of-speech, thus bringing these categories into the realm of popular culture. As it happens, Grammar Rock was remarkably traditional in its grammatical notation, including exactly eight songs about parts-of-speech. Although the list was slightly modified from Thrax’s original, substituting adjective and interjection for the original participle and article, the astonishing durability of the parts-of-speech through two millennia is an indicator of both the importance and the transparency of their role in human language.

More recent lists of parts-of-speech (or tagsets) have much larger numbers of word classes; 45 for the Penn Treebank (Marcus et al., 1993), 87 for the Brown corpus (Francis, 1979; Francis and Kučera, 1982), and 146 for the C7 tagset (Garside et al., 1997).

The significance of parts-of-speech (also known as POS, word classes, morphological classes, or lexical tags) for language processing is the large amount of information they give about a word and its neighbors. This is clearly true for major categories, (verb versus noun), but is also true for the many finer distinctions. For example, these tagsets distinguish between possessive pronouns (my, your, his, her, its) and personal pronouns (I, you, he, me). Knowing whether a word is a possessive pronoun or a personal pronoun can tell us what words are likely to occur in its vicinity (possessive pronouns are likely to be followed by a noun, personal pronouns by a verb). This can be useful in a language model for speech recognition.

A word’s part-of-speech can tell us something about how the word is pronounced. As Ch. 7 will discuss, the word content, for example, can be a noun or an adjective. They are pronounced differently (the noun is pronounced content and the adjective contains). Thus knowing the part-of-speech can produce more natural pronunciations in a speech synthesis system and more accuracy in a speech recognition system. (Other pairs like this include Object (noun) and subject (verb), Discount (noun) and Mozilla (verb); see Cutler (1986)).

Parts-of-speech can also be used in stemming for informational retrieval (IR), since knowing a word’s part-of-speech can help tell us which morphological affixes it can take, as we saw in Chapter 3. They can also enhance an IR application by selecting out nouns or other important words from a document. Automatic assignment of part-of-speech plays a role in word-sense disambiguation algorithms, and in class-based N-gram language models for speech recognition, discussed in Ch. 6. Parts-of-speech are used in shallow parsing of texts to quickly find names, times, dates, or other named entities for the information extraction applications discussed in Ch. 15. Finally, corpora that have been marked for parts-of-speech are very useful for linguistic research. For example, they can be used to help find instances or frequencies of particular constructions.

This chapter focuses on computational methods for assigning parts-of-speech to words (part-of-speech tagging). Many algorithms have been applied to this problem, including hand-written rules (rule-based tagging), probabilistic methods (HMM tagging and maximum entropy tagging), as well as other methods such as transformation-based tagging and memory-based tagging. We will introduce three of these algorithms in this chapter: rule-based tagging, HMM tagging, and...
transformation-based tagging. But before turning to the algorithms themselves, let's begin with a summary of English word classes, and of various tagsets for formally coding these classes.

### 5.1 (MOSTLY) ENGLISH WORD CLASSES

Well, every person you can know,
And every place that you can go,
And anything that you can show,
You know they're nouns.

Lynn Ahrens, Schoolhouse Rock, 1973

Until now we have been using part-of-speech terms like **noun** and **verb** rather freely. In this section we give a more complete definition of these and other classes. Traditionally the definition of parts-of-speech has been based on syntactic and morphological function; words that function similarly with respect to what can occur nearby (their "syntactic distributional properties"), or with respect to the affixes they take (their morphological properties) are grouped into classes. While word classes do have tendencies toward semantic coherence (nouns do in fact often describe "people, places or things", and adjectives often describe properties), this is not necessarily the case, and in general we don’t use semantic coherence as a definitional criterion for parts-of-speech.

Parts-of-speech can be divided into two broad supercategories: **closed class** types and **open class** types. Closed classes are those that have relatively fixed membership. For example, prepositions are a closed class because there is a fixed set of them in English; new prepositions are rarely coined. By contrast nouns and verbs are open classes because new nouns and verbs are continually coined or borrowed from other languages (e.g., the new verb to fax or the borrowed noun futon). It is likely that any given speaker or corpus will have different open class words, but all speakers of a language, and corpora that are large enough, will likely share the set of closed class words. Closed class words are also generally **function words** like of, it, and, or you, which tend to be very short, occur frequently, and often have structuring uses in grammar.

There are four major open classes that occur in the languages of the world; **nouns**, **verbs**, **adjectives**, and **adverbs**. It turns out that English has all four of these, although not every language does.

Noun is the name given to the syntactic class in which the words for most people, places, or things occur. But since syntactic classes like noun are defined syntactically and morphologically rather than semantically, some words for people, places, and things may not be nouns, and conversely some nouns may not be words for people, places, or things. Thus nouns include concrete terms like ship and chair, abstractions like bandwidth and relationship, and verb-like terms like pacing as in *His pacing to and fro became quite annoying*. What defines a noun in English, then, are things like its ability to occur with determiners (a goat, its bandwidth, Plato’s Republic), to take possessives (IBM’s annual revenue), and for most but not all nouns, to occur in the plural form (goats, abacuses).

Nouns are traditionally grouped into **proper nouns** and **common nouns**. Proper nouns, like Regina, Colorado, and IBM, are names of specific persons or entities. In English, they generally aren’t preceded by articles (e.g., *the book is upstairs*), but Regina *is* upstairs). In written English, proper nouns are usually capitalized.

In many languages, including English, common nouns are divided into **count nouns** and **mass nouns**. Count nouns are those that allow grammatical enumeration; that is, they can occur in both the singular and plural (goats/goat, relationship/relationships) and they can be counted (one goat, two goats). Mass nouns are used when something is conceptualized as a homogeneous group. So words like *snow, salt, and communism* are not counted (i.e., *two snows or *two communisms*). Mass nouns can also appear without articles where singular count nouns cannot (*Snow is white but not *Goat is white*).

The verb class includes most of the words referring to actions and processes, including main verbs like draw, provide, differ, and go. As we saw in Ch. 3, English verbs have a number of morphological forms (non-3rd-person-sg (eats), 3rd-person-sg (eats), progressive (eating)), past participle (eaten)). A subclass of English verbs called **auxiliaries** will be discussed when we turn to closed class forms.

While many researchers believe that all human languages have the categories of noun and verb, others have argued that some languages, such as Riau Indonesian and Tongan, don’t even make this distinction (Broschart, 1997; Evans, 2000; Gil, 2000).

The third open class English form is adjectives; semantically this class includes many terms that describe properties or qualities. Most languages have adjectives for the concepts of color (*white, black*), age (*old, young*), and value (*good, bad*), but there are languages without adjectives. In Korean, for example, the words corresponding to English adjectives act as a subclass of verbs, so what is in English an adjective "beautiful" acts in Korean like a verb meaning "to be beautiful" (Evans, 2000).

The final open class form, **adverbs**, is rather a hodge-podge, both semantically and formally. For example Schachter (1985) points out that in a sentence like the following, all the italicized words are adverbs:

> Unfortunately, John walked **home** extremely slowly yesterday
phrasal verbs from Thoreau:

Figure 5.2

meanings of the verb and the particle independently. Here are some examples of around beneath eastward(s), etc. on since without

around to go is 'discover', and find out means 'eliminate', rule out something like 'reject', from the separate meanings of the verb and the particle. Thus they often indicate action or process; and temporal adverbs describe the time that some action or event took place (yesterday, Monday). Because of the heterogeneous nature of this class, some adverbs (for example temporal adverbs like Monday) are tagged in some tagging schemes as nouns.

The closed classes differ more from language to language than do the open classes. Here’s a quick overview of some of the more important closed classes in English, with a few examples of each:

- **prepositions**: on, under, over, near, by, at, from, to, with
- **determiners**: a, an, the
- **conjunctions**: and, but, or, as, if, when
- **auxiliary verbs**: can, may, should, are
- **particles**: up, down, on, off, in, out, at, by,
- **numerals**: one, two, three, first, second, third

Prepositions occur before noun phrases; semantically they are relational, often indicating spatial or temporal relations, whether literal (on it, before then, by the house) or metaphorical (on time, with gusto, beside herself). But they often indicate other relations as well (Hamlet was written by Shakespeare, and [from Shakespeare] “And I did laugh sans intermission an hour by his dial”). Fig. 5.1 shows the prepositions of English according to the CELEX on-line dictionary (Baayen et al., 1995), sorted by their frequency in the COBUILD 16 million word corpus of English. Fig. 5.1 should not be considered a definitive list, since different dictionaries and tagset label word classes differently. Furthermore, this list combines prepositions and particles.

A particle is a word that resembles a preposition or an adverb, and is used in combination with a verb. When a verb and a particle behave as a single syntactic and/or semantic unit, we call the combination a phrasal verb. Phrasal verbs can behave as a semantic unit; thus they often have a meaning that is not predictable from the separate meanings of the verb and the particle. Thus turn down means something like ‘reject’, rule out means ‘eliminate’, and go on is ‘continue’. These are not meanings that could have been predicted from the meanings of the verb and the particle independently. Here are some examples of phrasal verbs from Thoreau:

<table>
<thead>
<tr>
<th>Common phrasal verbs</th>
<th>Preposition</th>
<th>Particle</th>
</tr>
</thead>
<tbody>
<tr>
<td>go up, go down</td>
<td>up, down</td>
<td></td>
</tr>
<tr>
<td>come in, come out</td>
<td>in, out</td>
<td></td>
</tr>
<tr>
<td>go around, go round</td>
<td>around, round</td>
<td></td>
</tr>
</tbody>
</table>

So I went on for some days cutting and hewing timber… Moral reform is the effort to throw off sleep…” Particles don’t always occur with idiomatic phrasal verb semantics; here are more examples of particles from the Brown corpus:

- …she had turned the paper over.
- He arose slowly and brushed himself off.
- He packed up his clothes.

We show in Fig. 5.2 a list of single-word particles from Quirk et al. (1985). Since it is extremely hard to automatically distinguish particles from prepositions, some tagsets (like the one used for CELEX) do not distinguish them, and even in corpora that do (like the Penn Treebank) the distinction is very difficult to make reliably in an automatic process, so we do not give counts.
Section 5.1. (Mostly) English Word Classes

A closed class that occurs with nouns, often marking the beginning of a noun phrase, is the determiners. One small subtype of determiners is the articles: English has three articles: *a*, *an*, and *the*. Other determiners include *this* (as in this chapter) and *that* (as in that page). *A* and *an* mark a noun phrase as indefinite, while *the* can mark it as definite; definiteness is a discourse and semantic property that will be discussed in Ch. 18. Articles are quite frequent in English; indeed the is the most frequently occurring word in most corpora of written English. Here are COBUILD statistics, again out of 16 million words:

the: 1,071,676 a: 413,887 an: 59,359

Conjunctions are used to join two phrases, clauses, or sentences. Coordinating conjunctions like *and*, *or*, and *but*, join two elements of equal status. Subordinating conjunctions are used when one of the elements is of some sort of embedded status. For example that in *"I thought that you might like some milk"* is a subordinating conjunction that links the main clause *I thought* with the subordinate clause *you might like some milk*. This clause is called subordinate because this entire clause is the “content” of the main verb thought. Subordinating conjunctions like *that* which link a verb to its argument in this way are also called complementizers. Ch. 9 and Ch. 11 will discuss complementation in more detail. Table 5.3 lists English conjunctions.

<table>
<thead>
<tr>
<th>Conjunctions</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>and</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>as</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>because</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>but</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>if</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>since</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>than</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>that</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>yet</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>lest</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>nor</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>or</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>so</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>therefore</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>unless</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>although</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>but</em></td>
<td>514,946</td>
</tr>
<tr>
<td><em>consequently</em></td>
<td>174</td>
</tr>
<tr>
<td><em>considering</em></td>
<td>174</td>
</tr>
<tr>
<td><em>forasmuch</em></td>
<td>174</td>
</tr>
<tr>
<td><em>getting</em></td>
<td>174</td>
</tr>
<tr>
<td><em>thereupon</em></td>
<td>174</td>
</tr>
<tr>
<td><em>whereas</em></td>
<td>174</td>
</tr>
<tr>
<td><em>as</em></td>
<td>174</td>
</tr>
<tr>
<td><em>although</em></td>
<td>174</td>
</tr>
<tr>
<td><em>but</em></td>
<td>174</td>
</tr>
<tr>
<td><em>than</em></td>
<td>174</td>
</tr>
</tbody>
</table>

Figure 5.3: Coordinating and subordinating conjunctions of English from CELEX.

Pronouns are forms of speech that can also be a kind of shorthand for referring to some noun phrase or entity or event. *Personal pronouns* refer to persons or entities (you, she, I, it, me, etc.). *Possessive pronouns* are forms of personal pronouns that indicate either actual possession or more often just an abstract relation between the person and some object (my, your, his, her, its, one’s, our, their). Wh-pronouns (what, who, whom, whoever) are used in certain question forms, or may also act as complementizers (Frieda, who I met five years ago . . .). Table 5.4 shows English pronouns, again from CELEX.

<table>
<thead>
<tr>
<th>Pronouns</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>it</em></td>
<td>199,920</td>
</tr>
<tr>
<td><em>he</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>she</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>you</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>we</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>they</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>his</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>her</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>our</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>their</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>my</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>her</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>our</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>their</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>yours</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>which</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>all</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>both</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>each</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>every</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>any</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>some</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>what</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>whom</em></td>
<td>198,139</td>
</tr>
<tr>
<td><em>whose</em></td>
<td>198,139</td>
</tr>
</tbody>
</table>

Figure 5.4: Pronouns of English from the CELEX on-line dictionary. Frequency counts are from the COBUILD 16 million word corpus.

A closed class subtype of English verbs are the auxiliary verbs. Crosslinguistically, auxiliaries are words (usually verbs) that mark certain semantic features of a main verb, including whether an action takes place in the present, past or future (tense), whether it is completed (aspect), whether it is negated (polarity), and whether an action is necessary, possible, suggested, desired, etc. (mood). English auxiliaries include the copula verb be, the two verbs do and have,
Section 5.2. Tagsets for English

The previous section gave broad descriptions of the kinds of syntactic classes that English words fall into. This section fleshes out that sketch by describing the actual tagsets used in part-of-speech tagging, in preparation for the various tagging algorithms to be described in the following sections.

### 5.2 Tagsets for English

Along with their inflected forms, as well as a class of modal verbs, *be* is called a copula because it connects subjects with certain kinds of predicate nominals and adjectives (*He is a duck*). The verb *have* is used for example to mark the perfect tenses (*I have gone, I had gone*), while *be* is used as part of the passive (*We were robbed*, or progressive (*We are leaving*)) constructions. The modals are used to mark the mood associated with the event or action depicted by the main verb. So *can* indicates ability or possibility, *may* indicates permission or possibility, *must* indicates necessity, and so on. Fig. 5.5 gives counts for the frequencies of the modals in English. In addition to the perfect tenses mentioned above, there is a modal verb *have* (*e.g.*, *I have to go*), which is very common in spoken English. Neither it nor the modal verb *dare*, which is very rare, have frequency counts because the CELEX dictionary does not distinguish the main verb sense (*I have three oranges, He dared me to eat them*), from the modal sense (*There has to be some mistake, Dare I confront him*?), from the non-modal auxiliary verb sense (*I have never seen that*).

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Coordn. Conjunction</td>
<td>and, but, or</td>
</tr>
<tr>
<td>CD</td>
<td>Cardinal number</td>
<td>one, two, three</td>
</tr>
<tr>
<td>DT</td>
<td>Determiner</td>
<td>a, the</td>
</tr>
<tr>
<td>EX</td>
<td>Existential ‘there’</td>
<td>there</td>
</tr>
<tr>
<td>FW</td>
<td>Foreign word</td>
<td>mea culpa</td>
</tr>
<tr>
<td>IN</td>
<td>Preposition/sub-conj</td>
<td>of, in, by</td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
<td>yellow</td>
</tr>
<tr>
<td>JJR</td>
<td>Adj., comparative</td>
<td>bigger</td>
</tr>
<tr>
<td>JJT</td>
<td>Adj., superlative</td>
<td>wildest</td>
</tr>
<tr>
<td>LS</td>
<td>List item marker</td>
<td>1, 2, One</td>
</tr>
<tr>
<td>MD</td>
<td>Modal</td>
<td>can, should</td>
</tr>
<tr>
<td>NN</td>
<td>Noun, singular</td>
<td>llamas</td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
<td>llamas</td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
<td>IBM</td>
</tr>
<tr>
<td>NNPS</td>
<td>Proper noun, plural</td>
<td>Carolinas</td>
</tr>
<tr>
<td>PDT</td>
<td>Preposition/possessive</td>
<td>all, both</td>
</tr>
<tr>
<td>POS</td>
<td>Possessive ending</td>
<td>’s</td>
</tr>
<tr>
<td>PRP</td>
<td>Personal pronoun</td>
<td>I, you, he</td>
</tr>
<tr>
<td>PRPS</td>
<td>Possessive pronoun</td>
<td>your, one’s</td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
<td>quietly, never</td>
</tr>
<tr>
<td>RBR</td>
<td>Adverb, comparative</td>
<td>faster</td>
</tr>
<tr>
<td>RBS</td>
<td>Adverb, superlative</td>
<td>fastest</td>
</tr>
<tr>
<td>RP</td>
<td>Particle</td>
<td>up, off</td>
</tr>
<tr>
<td>SYM</td>
<td>Symbol</td>
<td>$, %, &amp;</td>
</tr>
</tbody>
</table>

**FIGURE 5.5** English modal verbs from the CELEX on-line dictionary. Frequency counts are from the COBUILD 16 million word corpus.

English also has many words of more or less unique function, including interjections (*oh, ah, hey, man, alas, uh, um*), negatives (*no, nor, not*), politeness markers (*please, thank you*), greetings (*hello, goodbye*), and the existential *there* (*there are two on the table*) among others. Whether these classes are assigned particular names or lumped together (as interjections or even adverbs) depends on the purpose of the labeling.

### 5.2.1 Tagsets for English

The Penn Treebank tagset, shown in Fig. 5.6, has been applied to the Brown corpus, focusing on the smallest, the Penn Treebank set, and discuss difficult tagging decisions in that tag set and some useful distinctions made in the larger tagsets.

There are a small number of popular tagsets for English, many of which evolved from the 87-tag tagset used for the Brown corpus (Francis, 1979; Francis and Kucera, 1982). The Brown corpus is a 1 million word collection of samples from 500 written texts from different genres (newspaper, novels, non-fiction, academic, etc.) which was assembled at Brown University in 1963–1964 (Kucera and Francis, 1967; Francis, 1979; Francis and Kucera, 1982). This corpus was evolved from the 87-tag tagset used for the Brown corpus (Francis, 1979; Francis and Kucera, 1982). The Brown corpus is a 1 million word collection of samples from 500 written texts from different genres (newspaper, novels, non-fiction, academic, etc.) which was assembled at Brown University in 1963–1964 (Kucera and Francis, 1967; Francis, 1979; Francis and Kucera, 1982). This corpus was tagged with parts-of-speech by first applying the TAGGIT program and then hand-correcting the tags.

Besides this original Brown tagset, two of the most commonly used tagsets are the small 45-tag Penn Treebank tagset (Marcus et al., 1993), and the medium-sized 61 tag C5 tagset used by the Lancaster UCREL project’s CLAWS (the Continent Likelihood Automatic Word-tagging System) tagger to tag the British National Corpus (BNC) (Gaswade et al., 1997). We give all three of these tagsets here, focusing on the smallest, the Penn Treebank set, and discuss difficult tagging decisions in that tag set and some useful distinctions made in the larger tagsets.

**FIGURE 5.6** Penn Treebank part-of-speech tags (including punctuation).
corpus, the Wall Street Journal corpus, and the Switchboard corpus among others; indeed, perhaps partly because of its small size, it is one of the most widely used tagsets. Here are some examples of tagged sentences from the Penn Treebank version of the Brown corpus (we will represent a tagged word by placing the tag after each word, delimited by a slash):

(5.1) The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

(5.2) There/EX are/VBP 70/CD children/NNS there/RB

(5.3) Although/IN preliminary/JJ findings/NNS were/VBD reported/VBN more/RBR than/IN a/DT year/NN ago/IN ./, the/DT latest/OJS results/NNS appear/VBP in/IN today/NN’s/POS New/NNP England/NNP Journal/NNP of/IN Medicine/NNP ./.

Example (5.1) shows phenomena that we discussed in the previous section; the determiners the and a, the adjectives grand and other, the common nouns jury, number, and topics, the past tense verb commented. Example (5.2) shows the use of the EX tag to mark the existential there construction in English, and, for comparison, another use of there which is tagged as an adverb (RB). Example (5.3) shows the segmentation of the possessive morpheme ’s, and shows an example of a passive construction, ‘were reported’, in which the verb reported is marked as a past participle (VBN), rather than a simple past (VBD). Note also that the proper noun New England is tagged NNP. Finally, note that since New England Journal of Medicine is a proper noun, the Treebank tagging chooses to mark each noun in it separately as NNP, including journal and medicine, which might otherwise be labeled as common nouns (NN).

Some tagging distinctions are quite hard for both humans and machines to make. For example prepositions (IN), particles (RP), and adverbs (RB) can have a large overlap. Words like around can be all three:

(5.4) Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG
(5.5) All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN
(5.6) Chateau/NNP Petrus/NNP costs/VBZ around/RB 250/CD

Making these decisions requires sophisticated knowledge of syntax; tagging manuals (Santorini, 1990) give various heuristics that can help human coders make these decisions, and that can also provide useful features for automatic taggers. For example two heuristics from Santorini (1990) are that prepositions generally are associated with a following noun phrase (although they also may be followed by prepositional phrases), and that the word around is tagged as an adverb when it means ‘approximately’. Furthermore, while particles often can either precede or follow a noun phrase object, as in the following examples:

(5.7) She told off/RP her friends
(5.8) She told her friends off/RP.

prepositions cannot follow their noun phrase (* is used here to mark an ungrammatical sentence, a concept which we will return to in Ch. 9):

(5.9) She stepped off/IN the train
(5.10) *She stepped the train off/IN.

Another difficulty is labeling the words that can modify nouns. Sometimes the modifiers preceding nouns are common nouns like cotton below, other times the Treebank tagging manual specifies that modifiers be tagged as adjectives (for example if the modifier is a hyphenated common noun like income-tax) and other times as proper nouns (for modifiers which are hyphenated proper nouns like Gramm-Rudman):

(5.11) cotton/NN sweater/NN
(5.12) income-tax/JJ return/NN
(5.13) the/DT Gramm-Rudman/NP Act/NP

Some words that can be adjectives, common nouns, or proper nouns, are tagged in the Treebank as common nouns when acting as modifiers:

(5.14) Chinese/NN cooking/NN
(5.15) Pacific/NN waters/NNS

A third known difficulty in tagging is distinguishing past participles (VBN) from adjectives (JJ). A word like married is a past participle when it is being used in an eventive, verbal way, as in (5.16) below, and is an adjective when it is being used to express a property, as in (5.17):

(5.16) They were married/VBN by the Justice of the Peace yesterday at 5:00.
(5.17) At the time, she was already married/JJ.

Tagging manuals like Santorini (1990) give various helpful criteria for deciding how ‘verb-like’ or ‘eventive’ a particular word is in a specific context.

The Penn Treebank tagset was culled from the original 87-tag tagset for the Brown corpus. This reduced set leaves out information that can be recovered from the identity of the lexical item. For example the original Brown and C5 tagsets include a separate tag for each of the different forms of the verbs do (e.g. C5 tag “VDD” for did and “VDG” for doing), be, and have. These were omitted from the Treebank set.

Certain syntactic distinctions were not marked in the Penn Treebank tagset because Treebank sentences were parsed, not merely tagged, and so some syntactic information is represented in the phrase structure. For example, the single tag
Section 5.3. Part-of-Speech Tagging

IN is used for both prepositions and subordinating conjunctions since the tree-structure of the sentence disambiguates them (subordinating conjunctions always precede clauses, prepositions precede noun phrases or prepositional phrases). Most tagging situations, however, do not involve parsed corpora; for this reason the Penn Treebank set is not specific enough for many uses. The original Brown and C5 tagsets, for example, distinguish prepositions (IN) from subordinating conjunctions (CS), as in the following examples:

(4.18) after/CS spending/VBG a/AT few/AP days/NNS at/AT Brown/NP Palace/NN Hotel/NN

(4.19) after/IN a/AT wedding/NN trip/NN to/IN Corpus/NP Christi/NP /

The original Brown and C5 tagsets also have two tags for the word to; in Brown the infinitive use is tagged TO, while the prepositional use as IN:

(4.20) to/TO give/VB priority/NN to/IN teacher/NN pay/NN raises/NNS

Brown also has the tag NR for adverbial nouns like home, west, Monday, and tomorrow. Because the Treebank lacks this tag, it has a much less consistent policy for adverbial nouns: Monday, Tuesday, and other days of the week are marked NNP, tomorrow, west, and home are marked sometimes as NN, sometimes as RB. This makes the Treebank tagset less useful for high-level NLP tasks like the detection of time phrases.

Nonetheless, the Treebank tagset has been the most widely used in evaluating tagging algorithms, and so many of the algorithms we describe below have been evaluated mainly on this tagset. Of course whether a tagset is useful for a particular application depends on how much information the application needs.

5.3 Part-of-Speech Tagging

Part-of-speech tagging (or just tagging for short) is the process of assigning a part-of-speech or other syntactic class marker to each word in a corpus. Because tags are generally also applied to punctuation, tagging requires that the punctuation marks (period, comma, etc) be separated off of the words. Thus tokenization of the sort described in Ch. 5 is usually performed before, or as part of, the tagging process, separating commas, quotation marks, etc., from words, and disambiguating end-of-sentence punctuation (period, question mark, etc) from part-of-word punctuation (such as in abbreviations like e.g. and etc.).

The input to a tagging algorithm is a string of words and a specified tagset of the kind described in the previous section. The output is a single best tag for each word. For example, here are some sample sentences from the ATIS corpus of dialogues about air-travel reservations that we will discuss in Ch. 9. For each

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>opening parenthesis</td>
<td>(</td>
</tr>
<tr>
<td>1</td>
<td>closing parenthesis</td>
<td>)</td>
</tr>
<tr>
<td>*</td>
<td>negator</td>
<td>!’</td>
</tr>
<tr>
<td>-</td>
<td>comma</td>
<td>,</td>
</tr>
<tr>
<td>-</td>
<td>dash</td>
<td>:</td>
</tr>
<tr>
<td>-</td>
<td>sentence terminator</td>
<td>?</td>
</tr>
<tr>
<td>-</td>
<td>colon</td>
<td>!</td>
</tr>
<tr>
<td>ABP</td>
<td>pre-qualifier</td>
<td>quite, rather, such</td>
</tr>
<tr>
<td>ABN</td>
<td>pre-quantertifier</td>
<td>half, all</td>
</tr>
<tr>
<td>AP</td>
<td>post-determiner</td>
<td>many, next, several, last</td>
</tr>
<tr>
<td>AT</td>
<td>article</td>
<td>a/the an an a every</td>
</tr>
<tr>
<td>BES</td>
<td>BES/BER/BER/BER/BES</td>
<td>between/between/between/between/between</td>
</tr>
</tbody>
</table>
we have shown a potential tagged output using the Penn Treebank tagset defined in Fig. 5.6 on page 10:

(5.21) Book/VB that/DT flight/NN ./.

(5.22) Does/VBZ you/PRP serve/VB dinner/NN ./.

The previous section discussed some tagging decisions that are difficult to make for humans. Even in these simple examples, automatically assigning a tag to each word is not trivial. For example, *book* is ambiguous. That is, it has more than one possible usage and part-of-speech. It can be a verb (as in *book that flight* or *to book the suspect*) or a noun (as in *book of matches* or *a book of matches*). Similarly that can be a determiner (as in *Does that flight serve dinner*), or a complementizer (as in *I thought that your flight was earlier*). The problem of POS-tagging is to resolve these ambiguities, choosing the proper tag for the context. Part-of-speech tagging is thus one of the many disambiguation tasks we will see in this book.

How hard is the tagging problem? The previous section described some difficult tagging decisions; how common is tag ambiguity? It turns out that most words in English are unambiguous; i.e., they have only a single tag. But many of the most common words of English are ambiguous (for example *can* can be an auxiliary (‘to be able’), a noun (‘a metal container’), or a verb (‘to put something in such a metal container’)). In fact, DeRose (1988) reports that while only 11.5% of English word types in the Brown corpus are ambiguous, over 40% of Brown tokens are ambiguous. Fig. 5.10 shows the number of word types with different levels of part-of-speech ambiguity from the Brown corpus. We show these computations from two versions of the tagged Brown corpus, the original tagging done at Brown by Francis and Kučera (1982), and the Treebank-3 tagging done at the University of Pennsylvania. Note that despite having more coarse-grained tags, the
45-tag corpus unexpectedly has more ambiguity than the 87-tag corpus.

<table>
<thead>
<tr>
<th>Unambiguous (1 tag)</th>
<th>Original 87-tag corpus</th>
<th>Treebank 45-tag corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular terms</td>
<td>44,019</td>
<td>38,857</td>
</tr>
<tr>
<td>Ambiguous (2–7 tags)</td>
<td>5,490</td>
<td>8844</td>
</tr>
<tr>
<td>Details</td>
<td>2 tags</td>
<td>4,967</td>
</tr>
<tr>
<td></td>
<td>3 tags</td>
<td>411</td>
</tr>
<tr>
<td></td>
<td>4 tags</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>5 tags</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>6 tags</td>
<td>2 (well, beat)</td>
</tr>
<tr>
<td></td>
<td>7 tags</td>
<td>2 (still, down)</td>
</tr>
<tr>
<td></td>
<td>8 tags</td>
<td>4 (’s, half, back, a)</td>
</tr>
<tr>
<td></td>
<td>9 tags</td>
<td>3 (that, more, in)</td>
</tr>
</tbody>
</table>

Figure 5.10 The amount of tag ambiguity for word types in the Brown corpus, from the ICAME release of the original (87-tag) tagging and the Treebank-3 (45-tag) tagging. Numbers are not strictly comparable because only the Treebank segments ‘a’ An earlier estimate of some of these numbers is reported in DeRose (1988).

Most tagging algorithms fall into one of two classes: rule-based taggers and stochastic taggers. Rule-based taggers generally involve a large database of hand-written disambiguation rules which specify, for example, that an ambiguous word is a noun rather than a verb if it follows a determiner. The next section will describe a sample rule-based tagger, EngCG, based on the Constraint Grammar approach of Karlsson et al. (1995b). In this section we describe a tagger based on this approach, the EngCG tagger (Voutilainen, 1995, 1999).

The EngCG ENGTWOL lexicon is based on the two-level morphology described in Ch. 3, and has about 36,000 entries for English word stems (Heikkilä, 1995), counting a word with multiple parts-of-speech (e.g., nominal and verbal senses of hit) as separate entries, and not counting inflected and many derived forms. Each entry is annotated with a set of morphological and syntactic features. Fig. 5.11 shows some selected words, together with a slightly simplified listing of their features; these features are used in rule writing.

<table>
<thead>
<tr>
<th>Word</th>
<th>POS</th>
<th>Additional POS features</th>
</tr>
</thead>
<tbody>
<tr>
<td>smaller</td>
<td>ADJ</td>
<td>COMPARATIVE</td>
</tr>
<tr>
<td>entire</td>
<td>ADJ</td>
<td>ABSOLUTE ATTRIBUTIVE</td>
</tr>
<tr>
<td>fast</td>
<td>ADV</td>
<td>SUPERLATIVE</td>
</tr>
<tr>
<td>that</td>
<td>DET</td>
<td>CENTRAL DEMONSTRATIVE SG</td>
</tr>
<tr>
<td>all</td>
<td>DET</td>
<td>PREDETERMINER SG/PL QUANTIFIER</td>
</tr>
<tr>
<td>dog's</td>
<td>N</td>
<td>GENITIVE SG</td>
</tr>
<tr>
<td>furniture</td>
<td>N</td>
<td>NOMINATIVE SG NONDETERMINER</td>
</tr>
<tr>
<td>one-third</td>
<td>NUM</td>
<td>SG</td>
</tr>
<tr>
<td>she</td>
<td>PRON</td>
<td>PERSONAL FEMININE NOMINATIVE SG3</td>
</tr>
<tr>
<td>show</td>
<td>V</td>
<td>PRESENT -SG VFIN</td>
</tr>
<tr>
<td>shown</td>
<td>N</td>
<td>NOMINATIVE SG</td>
</tr>
<tr>
<td>occurred</td>
<td>PCP2</td>
<td>SV,SOO,SOV SV</td>
</tr>
<tr>
<td>occurred</td>
<td>PCP2</td>
<td>V</td>
</tr>
</tbody>
</table>

Figure 5.11 Sample lexical entries from the ENGTWOL lexicon described in Voutilainen (1995) and Heikkilä (1995).

Most of the features in Fig. 5.11 are relatively self-explanatory; SG for singular, -SG for other than third-person-singular. ABSOLUTE means non-comparative and non-superlative for an adjective, NOMINATIVE just means non-genitive, and PCP2 means past participle. PRE, CENTRAL, and POST are ordering slots for determiners (predeterminers (all) come before determiners (the): all the president’s
Section 5.4. Rule-Based Part-of-Speech Tagging

Men). NOINDEFDETERMINER means that words like furniture do not appear with the indefinite determiner a. SV, SVO, and SVOO specify the subcategorization or complementation pattern for the verb. Subcategorization will be discussed in Ch. 9 and Ch. 11, but briefly SV means the verb appears solely with a subject (nothing occurred); SVO with a subject and an object (I showed the film); SVOO with a subject and two complements: She showed her the ball.

In the first stage of the tagger, each word is run through the two-level lexicon transducer and the entries for all possible parts-of-speech are returned. For example the phrase Pavlov had shown that salivation . . . would return the following list (one line per possible tag, with the correct tag shown in boldface):

Pavlov  PA VLOV  N  NOM  SG  PROPER
had  HA VE  V  PAST  VFIN  SVO
shown  SHOW  PCP2  SVOO  SVO  SV
that  ADV
salivation  NN  O  MS  G
...

EngCG then applies a large set of constraints (as many as 3,744 constraints in the EngCG-2 system) to the input sentence to rule out incorrect parts-of-speech. The boldfaced entries in the table above show the desired result, in which the simple past tense tag (rather than the past participle tag) is applied to had, and the complementizer (CS) tag is applied to that. The constraints are used in a negative way, to eliminate tags that are inconsistent with the context. For example one constraint eliminates all readings of that except the ADV (adverbial intensifier) sense (this is the sense in the sentence it isn’t that odd). Here’s a simplified version of the constraint:

**ADVERBIAL-THAT RULE**

**Given input:** “that”

if (+1 ADV/QUANT); /* if next word is adj, adverb, or quantifier */
(+2 SENT-LIM); /* and following which is a sentence boundary. */
(NOT -1 SVOC/A); /* and the previous word is not a verb like */
/* ‘consider’ which allows adjs as object complements */
then eliminate non-ADV tags
else eliminate ADV tag

The first two clauses of this rule check to see that the that directly precedes a sentence-final adjective, adverb, or quantifier. In all other cases the adverb reading is eliminated. The last clause eliminates cases preceded by verbs like consider or believe which can take a noun and an adjective; this is to avoid tagging the following instance of that as an adverb:

I consider that odd.

Another rule is used to express the constraint that the complementizer sense of that is most likely to be used if the previous word is a verb which expects a complement (like believe, think, or show), and if that is followed by the beginning of a noun phrase, and a finite verb.

This description oversimplifies the EngCG architecture; the system also includes probabilistic constraints, and also makes use of other syntactic information we haven’t discussed. The interested reader should consult Karlsson et al. (1995b) and Voutilainen (1999).

5.5 HMM PART-OF-SPEECH TAGGING

The use of probabilities in tags is quite old; probabilities in tagging were first used by Stolz et al. (1965), a complete probabilistic tagger with Viterbi decoding was sketched by Bahl and Mercer (1976), and various stochastic taggers were built in the 1980s (Marshall, 1983; Garside, 1987; Church, 1988; DeRose, 1988). This section describes a particular stochastic tagging algorithm generally known as the Hidden Markov Model or HMM tagger. Hidden Markov Models themselves will be more fully introduced and defined in Chapter 6. In this section, we prefigure Chapter 6 a bit by giving an initial introduction to the Hidden Markov Model as applied to part-of-speech tagging.

Use of a Hidden Markov Model to do part-of-speech-tagging, as we will define it, is a special case of Bayesian inference, a paradigm that has been known since the work of Bayes (1763). Bayesian inference or Bayesian classification was applied successfully to language problems as early as the late 1950s, including the OCR work of Bledsoe in 1959, and the seminal work of Mosteller and Wallace (1964) on applying Bayesian inference to determine the authorship of the Federalist papers.

In a classification task, we are given some observation(s) and our job is to determine which of a set of classes it belongs to. Part-of-speech tagging is generally treated as a sequence classification task. So here the observation is a sequence of words (let’s say a sentence), and it is our job to assign them a sequence of part-of-speech tags.

For example, say we are given a sentence like
Section 5.5. HMM Part-of-Speech Tagging

(5.23) Secretariat is expected to race tomorrow.

What is the best sequence of tags which corresponds to this sequence of words? The Bayesian interpretation of this task starts by considering all possible sequences of classes—in this case, all possible sequences of tags. Out of this universe of tag sequences, we want to choose the tag sequence which is most probable given the observation sequence of \( n \) words \( w_n \). In other words, we want, out of all sequences of \( n \) tags \( t_n \), the single tag sequence such that \( P(t_n | w_n) \) is highest.

We use the hat notation \( \hat{t} \) to mean “our estimate of the correct tag sequence”:

\[
\hat{t}_n = \arg\max_{t_n} P(t_n | w_n)
\]  

(5.24)

The function \( \arg\max \), \( f(x) \) means “the \( x \) such that \( f(x) \) is maximized”. Equation (5.24) thus means, out of all tag sequences of length \( n \), we want the particular tag sequence \( \hat{t}_n \) which maximizes the right hand side. While (5.24) is guaranteed to give us the optimal tag sequence, it is not clear how to make the equation operational; that is, for a given tag sequence \( t_i \) and word sequence \( w_i \), we don’t know how to directly compute \( P(t_i | w_i) \).

The intuition of Bayesian classification is to use Bayes’ rule to transform (5.24) into a set of other probabilities which turn out to be easier to compute. Bayes’ rule is presented in (5.25); it gives us a way to break down any conditional probability \( P(x|y) \) into three other probabilities:

\[
P(x|y) = \frac{P(x)P(y|x)}{P(y)}
\]  

(5.25)

We can then substitute (5.25) into (5.24) to get (5.26):

\[
\hat{t}_n = \arg\max_{t_n} \frac{P(w_n | t_n)P(t_n)}{P(w_n)}
\]  

(5.26)

We can conveniently simplify 5.26 by dropping the denominator \( P(w_n) \). Why is that? Since we are choosing a tag sequence out of all tag sequences, we will be computing \( P(w_n | t_n) \) for each tag sequence. But \( P(w_n) \) doesn’t change for each tag sequence; we are always asking about the most likely tag sequence for the same observation \( w_n \), which must have the same probability \( P(w_n) \). Thus we can choose the tag sequence which maximizes this simpler formula:

\[
\hat{t}_n = \arg\max_{t_n} P(w_n | t_n)P(t_n)
\]  

(5.27)

To summarize, the most probable tag sequence \( \hat{t}_n \) given some word string \( w_n \) can be computed by taking the product of two probabilities for each tag sequence, and choosing the tag sequence for which this product is greatest. The two terms are the prior probability of the tag sequence \( P(t_n) \), and the likelihood of the word sequence \( L(t_n | w_n) \).
the 87-tag tagset, and again in the 1990’s with the 45-tag Treebank tagset. This makes it useful for comparing tagsets, and is also widely available.

In the 45-tag Treebank Brown corpus, the tag DT occurs 116,454 times. Of these, DT is followed by NN 56,509 times (if we ignore the few cases of ambiguous tags). Thus the MLE estimate of the transition probability is calculated as follows:

\[ P(\text{NN|DT}) = \frac{C(\text{DT,NN})}{C(\text{DT})} \]

The probability of getting a common noun after a determiner, .49, is indeed quite high, as we suspected.

The word likelihood probabilities, \( P(w_i|t) \), represent the probability, given that we see a given tag, that it will be associated with a given word. For example if we were to see the tag VBZ (third person singular present verb) and guess the verb that is likely to have that tag, we might likely guess the verb is, since the verb to be is so common in English.

We can compute the MLE estimate of a word likelihood probability like \( P(\text{is|VBZ}) \) again by counting, out of the times we see VBZ in a corpus, how many of those times the VBZ is labeling the word is. That is, we compute the following ratio of counts:

\[ P(\text{is|VBZ}) = \frac{C(\text{is,VBZ})}{C(\text{VBZ})} \]

In Treebank Brown corpus, the tag VBZ occurs 21,627 times, and VBZ is the tag for is 10,073 times. Thus:

\[ P(\text{is|VBZ}) = \frac{C(\text{is,VBZ})}{C(\text{VBZ})} = \frac{10,073}{21,627} = .47 \]

For those readers who are new to Bayesian modeling note that this likelihood term is not asking “which is the most likely tag for the word is?”. That is, the term is not \( P(\text{VBZ|is}) \). Instead we are computing \( P(\text{is|VBZ}) \). The probability, slightly counterintuitively, answers the question “If we were expecting a third person singular verb, how likely is it that this verb would be \( \text{is}\)?”.

We have now defined HMM tagging as a task of choosing a tag-sequence with the maximum probability, derived the equations by which we will compute this probability, and shown how to compute the component probabilities. In fact we have simplified the presentation of the probabilities in many ways; in later sections we will return to these equations and introduce the deleted interpolation algorithm for smoothing these counts, the trigram model of tag history, and a model for unknown words.

But before turning to these augmentations, we need to introduce the decoding algorithm by which these probabilities are combined on line to choose the most likely tag sequence.
5.5.1 Computing the most-likely tag sequence: A motivating example

The previous section showed that the HMM tagging algorithm chooses as the most likely tag sequence the one that maximizes the product of two terms; the probability of the sequence of tags, and the probability of each tag generating a word. In this section we ground these examples in a specific sentence, showing for a particular sentence how the correct tag sequence achieves a higher probability than one of the many possible wrong sequences.

We will focus on resolving the part-of-speech ambiguity of the word *race*, which can be a noun or verb in English, as we show in two examples modified from the Brown and Switchboard corpus. For this example, we will use the 87-tag Brown corpus tagset, because it has a specific tag for *to*, TO, used only when *to* is an infinitive; prepositional uses of *to* are tagged as IN. This will come in handy in our example.¹

In (5.36) *race* is a verb (VB) while in (5.37) *race* is a common noun (NN):

(5.36) Secretariat/NNP is/BEZ expected/VBN to/TO race/VB tomorrow/NR

(5.37) People/NNS continue/VB to/TO inquire/VB the/AT reason/NN for/IN the/AT race/NN for/IN outer/JJ space/NN

Let’s look at how *race* can be correctly tagged as a VB instead of an NN in (5.36). HMM part-of-speech taggers resolve this ambiguity globally rather than locally, picking the best tag sequence for the whole sentence. There are many hypothetically possible tag sequences for (5.36), since there are other ambiguities in the sentence (for example *expected* can be an adjective (JJ), a past tense/preterite (VBD) or a past participle (VBN)). But let’s just consider two of the potential sequences, shown in Fig. 5.12. Note that these sequences differ only in one place; whether the tag chosen for *race* is VB or NN.

Almost all the probabilities in these two sequences are identical; in Fig. 5.12 we have highlighted in boldface the three probabilities that differ. Let’s consider two of these, corresponding to \( P(t_i|t_{i-1}) \) and \( P(w_i|t_i) \). The probability \( P(t_i|t_{i-1}) \) in Figure 5.12a is \( P(VB|TO) \), while in Figure 5.12b the transition probability is \( P(NN|TO) \).

The tag transition probabilities \( P(NN|TO) \) and \( P(VB|TO) \) give us the answer to the question “How likely are we to expect a verb (noun) given the previous tag?” As we saw in the previous section, the maximum likelihood estimate for these probabilities can be derived from corpus counts.

Since the (87-tag Brown tagset) tag TO is used only for the infinitive marker *to*, we expect that only a very small number of nouns can follow this marker (as an exercise, try to think of a sentence where a noun can follow the infinitive marker use of *to*). Sure enough, a look at the (87-tag) Brown corpus gives us the following probabilities, showing that verbs are about 500 times as likely as nouns to occur after TO:

\[
P(NN|TO) = .00047
\]
\[
P(VB|TO) = .83
\]

Let’s now turn to \( P(w_i|t_i) \), the lexical likelihood of the word *race* given a part-of-speech tag. For the two possible tags VB and NN, these correspond to the probabilities \( P(race|VB) \) and \( P(race|NN) \). Here are the lexical likelihoods from Brown:

\[
P(race|NN) = .00057
\]
\[
P(race|VB) = .00012
\]

Finally, we need to represent the tag sequence probability for the following tag (in this case the tag NR for *tomorrow*):

\[
P(NR|VB) = .0027
\]

¹ The 45-tag Treebank-3 tagset does make this distinction in the Switchboard corpus but not, alas, in the Brown corpus. Recall that in the 45-tag tagset time adverbs like *tomorrow* are tagged as NN; in the 87-tag tagset they appear as NR.
5.5.2 Formalizing Hidden Markov Model taggers

Now that we have seen the equations and some examples of choosing the most probable tag sequence, we show the formalization of this problem as a Hidden Markov Model.

The HMM is an extension of the finite automata of Ch. 3. Recall that a finite automaton is defined by a set of states, and a set of transitions between states that are taken based on the input observations. A weighted finite-state automaton is a simple augmentation of the finite automaton in which each arc is associated with a probability, indicating how likely that path is to be taken. The probability on all the arcs leaving a node must sum to 1. A Markov chain is a special case of a weighted automaton in which the input sequence uniquely determines which states the automaton will go through. Because they can’t represent inherently ambiguous problems, a Markov chain is only useful for assigning probabilities to unambiguous sequences.

While the vanilla Markov Model is appropriate for situations where we can see the actual conditioning events, it is not appropriate in part-of-speech tagging. This is because in part-of-speech tagging, while we observe the words in the input, we do not observe the part-of-speech tags. Thus we can’t condition any probabilities on, say, a previous part-of-speech tag, because we cannot be completely certain exactly which tag applied to the previous word. A Hidden Markov Model (HMM) allows us to talk about both observed events (like words that we see in the input) and hidden events (like part-of-speech tags) that we think of as causal factors in our probabilistic model.

An HMM is specified by a set of states $Q$, a set of observation likelihoods $B$, a defined start state and end state(s), and a set of observation symbols $O$, which is not drawn from the same alphabet as the state set $Q$.

In summary, here are the parameters we need to define an HMM:

- states: a set of states $Q = \{q_1, q_2, \ldots, q_n\}$
- observation symbols: $O = \{o_1, o_2, \ldots, o_m\}$
- start state: $\pi$ (probability of starting in state $q_1$)
- transition probabilities: $A = \{a_{ij}\}$
  
  $$P(q'_t \mid q_t) = a_{q_t q'_t}$$

- emission probabilities: $B = \{b_{q_o}\}$
  
  $$P(o_t \mid q_t) = b_{q_t o_t}$$

- end state(s): $\eta$ (probability of ending in state $q_n$)

If we multiply the lexical likelihoods with the tag sequence probabilities, we see that the probability of the sequence with the VB tag is higher and the HMM tagger correctly tags race as a VB in Fig. 5.12 despite the fact that it is the less likely sense of race:

$$P(\text{race}|\text{VB})P(\text{NN}|\text{VB}) = 0.00000032$$

Taggers are often evaluated by comparing them with a human-labeled Gold Standard test set, based on accuracy: the percentage of all tags in the test set where the tagger and the Gold standard agree. Most current tagging algorithms have an accuracy of around 96-97% for simple tagsets like the Penn Treebank set. These accuracies are for words and punctuation; the accuracy for words only would be lower.

How good is 97%? Since tagsets and tasks differ, the performance of tags can be compared against a lower-bound baseline and an upper-bound ceiling. One way to set a ceiling is to see how well humans do on the task. Marcus et al. (1993), for example, found that human annotators agreed on about 96-97% of the tags in the Penn Treebank version of the Brown corpus. This suggests that the Gold Standard may have a 3-4% margin of error, and that it is meaningless to get 100% accuracy, (modeling the last 3% would just be modeling noise). Indeed Ratnaparkhi (1996) showed that the tagging ambiguities that caused problems for his tagger were exactly the ones that humans had labeled inconsistently in the training set. Two experiments by Toutainen (1995, p. 174), however, found that when humans were allowed to discuss tags, they reached consensus on 100% of the tags.

Human Ceiling: When using a human Gold Standard to evaluate a classification algorithm, check the agreement rate of humans on the standard.

The standard baseline, suggested by Gale et al. (1992) (in the slightly different context of word-sense disambiguation), is to choose the unigram most-likely tag for each ambiguous word. The most-likely tag for each word can be computed from a hand-tagged corpus (which may be the same as the training corpus for the tagger being evaluated).

Unigram Baseline: Always compare a classifier against a baseline at least as good as the unigram baseline (assigning each token to the class it occurred in most often in the training set).

Tagging algorithms since Harris (1962) incorporate this tag frequency intuition. Charniak et al. (1993) showed that this baseline algorithm achieves an accuracy of 90-91% on the 87-tag Brown tagset. Toutanova et al. (2003) showed that a more complex version, augmented with an unknown word model, achieved 93.69% on the 45-tag Treebank tagset.
Section 5.5. HMM Part-of-Speech Tagging

- transition probabilities: a set of probabilities \( A = a_{01}, a_{02}, \ldots, a_{0n} \). Each \( a_{ij} \) represents the probability of transitioning from state \( i \) to state \( j \). The set of these is the transition probability matrix.

- observation likelihoods: a set of observation likelihoods \( B = b_i(\alpha) \), each expressing the probability of an observation \( \alpha \) being generated from a state \( i \).

In our examples so far we have used two “special” states (non-emitting states) as the start and end state; it is also possible to avoid the use of these states by specifying two more things:

- initial distribution: an initial probability distribution over states, \( \pi \), such that \( \pi_i \) is the probability that the HMM will start in state \( i \). Of course some states \( j \) may have \( \pi_j = 0 \), meaning that they cannot be initial states.

- accepting states: a set of legal accepting states

An HMM thus has two kinds of probabilities: the \( A \) transition probabilities, and the \( B \) observation likelihoods, corresponding respectively to the prior and likelihood probabilities that we saw in equation (5.31). Fig. 5.13 illustrates the prior probabilities in an HMM part-of-speech tagger, showing 3 sample states and some of the \( A \) transition probabilities between them. Fig. 5.14 shows another view of an HMM part-of-speech tagger, focusing on the word likelihoods \( B \). Each hidden state is associated with a vector of likelihoods for each observation word.

![Figure 5.13](image)

**Figure 5.13** The weighted finite-state network corresponding to the hidden states of the HMM. The \( A \) transition probabilities are used to compute the prior probability.

![Figure 5.14](image)

**Figure 5.14** The \( B \) observation likelihoods for the HMM in the previous figure. Each state (except the non-emitting Start and End states) is associated with a vector of probabilities, one likelihood for each possible observation word.

### 5.5.3 The Viterbi Algorithm for HMM Tagging

For any model, such as an HMM, that contains hidden variables, the task of determining which sequence of variables is the underlying source of some sequence of observations is called the decoding task. The Viterbi algorithm is perhaps the most common decoding algorithm used for HMMs, whether for part-of-speech tagging or for speech recognition. The term Viterbi is common in speech and language processing, but this is really a standard application of the classic dynamic programming algorithm, and looks a lot like the minimum edit distance algorithm of Ch. 5. The Viterbi algorithm was first applied to speech and language processing in the context of speech recognition by Vintsyuk (1968), but has what Kruskal (1983) calls a ‘remarkable history of multiple independent discovery and publication’; see the History section at the end of Chapter 6 for more details.

The slightly simplified version of the Viterbi algorithm that we will present takes as input a single HMM and a set of observed words \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_T) \) and returns the most probable state-tag sequence \( q = (q_1, q_2, \ldots, q_T) \), together with its probability.

Let the HMM be defined by the two tables in Fig. 5.15 and Fig. 5.16. Fig. 5.15 expresses the \( a_{ij} \) probabilities, the transition probabilities between hidden states (i.e. part-of-speech tags). Fig. 5.16 expresses the \( b_i(\alpha) \) probabilities, the observation likelihoods of words given tags.

Fig. 5.17 shows pseudocode for the Viterbi algorithm. The Viterbi algorithm...
for each state in the state graph. Each column thus has a cell for each state. In Fig. 5.18, each cell in the column for the word VB is computed by multiplying the previous path probability at the start state (1.0), the transition probability from the start state to the tag for that cell, and the observation likelihood of the word VB.

In Fig. 5.18, each cell in the column for the word I is computed by multiplying the previous path probability at the start state (1.0), the transition probability from the start state to the tag for that cell, and the observation likelihood of the word I.

<table>
<thead>
<tr>
<th>state</th>
<th>TO</th>
<th>NN</th>
<th>PPSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB</td>
<td>.0038</td>
<td>.045</td>
<td>.041</td>
</tr>
<tr>
<td>TO</td>
<td>.83</td>
<td>0</td>
<td>.00047</td>
</tr>
<tr>
<td>PPSS</td>
<td>.00040</td>
<td>.016</td>
<td>.087</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.23</td>
</tr>
</tbody>
</table>

Figure 5.15 Tag transition probabilities (the a array, p(b,s−1)) computed from the 87-tag Brown corpus without smoothing. The rows are labeled with the conditioning event, thus P(PPSS|VB) is .0070. The symbol <s> is the start-of-sentence symbol.

<table>
<thead>
<tr>
<th>I</th>
<th>want to race</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB</td>
<td>0</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
</tr>
<tr>
<td>NN</td>
<td>0</td>
</tr>
<tr>
<td>PPSS</td>
<td>.37</td>
</tr>
</tbody>
</table>

Figure 5.16 Observation likelihoods (the b array) computed from the 87-tag Brown corpus without smoothing.

<table>
<thead>
<tr>
<th></th>
<th>TO</th>
<th>NN</th>
<th>PPSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB</td>
<td>.000070</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO</td>
<td>.000045</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>.000041</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPSS</td>
<td>.000054</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

sets up a probability matrix, with one column for each observation t and one row for each state in the state graph. Each column thus has a cell for each state qi in the
given the tag for that cell. As it turns out, three of the cells are zero (since the word
I can be neither NN, TO nor VB). Next, each cell in the want column gets updated
with the maximum probability path from the previous column. We have shown
only the value for the VB cell. That cell gets the max of four values; as it happens
in this case, three of them are zero (since there were zero values in the previous
column). The remaining value is multiplied by the relevant transition probability,
and the (trivial) max is taken. In this case the final value, .000051, comes from the
PPSS state at the previous column.

The reader should fill in the rest of the table in Fig. 5.18, and backtrack to
reconstruct the correct state sequence PPSS VB TO VB.

5.5.4 Extending the HMM algorithm to trigrams

We mentioned earlier that HMM taggers in actual use have a number of sophis-
tications not present in the simplified tagger as we have described it so far. One
important missing feature has to do with the tag context. In the tagger described
above, we assume that the probability of a tag appearing is dependent only on the
previous tag:

\[ P(t_i^1) \approx \prod_{t=1}^{n} P(t_i|t_{i-1}) \] (5.38)

Most modern HMM taggers actually use a little more of the history, letting
the probability of a tag depend on the two previous tags:

\[ P(t_i^2) \approx \prod_{t=1}^{n} P(t_i|t_{i-1}, t_{i-2}) \] (5.39)

In addition to increasing the window before a tagging decision, state-of-the-
art HMM taggers like Brants (2000) let the tagger know the location of the end
of the sentence by adding dependence on an end-of-sentence marker for \( \tau_{n+1} \). This
gives the following equation for part of speech tagging:

\[ r_n^* = \arg \max_{\tau_n} P(\tau_n^*|w_n^*) \approx \arg \max_{\tau_n} \prod_{t=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}, t_{i-2}) P(\tau_{n+1}|\tau_n) \] (5.40)

In tagging any sentence with (5.40), three of the tags used in the context will
fall off the edge of the sentence, and hence will not match regular words. These
tags, \( t_{-1}, \), \( k_0 \), and \( \tau_{n+1} \), can all be set to be a single special ‘sentence boundary’ tag
which is added to the tagset. This requires that sentences passed to the tagger have
sentence boundaries demarcated, as discussed in Ch. 5.

There is one large problem with (5.40); data sparsity. Any particular se-
queness of tags \( t_{-2}, t_{-1}, t_1 \) that occurs in the test set may simply never have occurred
in the training set. That means we cannot compute the tag trigram probability just
by the maximum likelihood estimate from counts, following Equation 5.41:

\[ P(t_i|t_{i-1}, t_{i-2}) = \frac{C(t_i, t_{i-1}, t_{i-2})}{C(t_{i-2}, t_{i-1})} \] (5.41)

Why not? Because many of these counts will be zero in any training set, and we
will incorrectly predict that a given tag sequence will never occur! What we need
is a way to estimate \( P(t_i|t_{i-1}, t_{i-2}) \) even if the sequence \( t_{-2}, t_{-1}, t_1 \) never occurs
in the training data.

The standard approach to solve this problem is to estimate the probability
by combining more robust, but weaker estimators. For example, if we’ve never
seen the tag sequence FRP VB TO, so we can’t compute \( P(TO/FRP/VB) \) from this
frequency, we still could rely on the bigram probability \( P(TO/VB) \), or even the
unigram probability \( P(TO) \). The maximum likelihood estimation of each of these
probabilities can be computed from a corpus via the following counts:

- **Trigrams**: \( \hat{P}(t_i|t_{i-1}, t_{i-2}) = \frac{C(t_i, t_{i-1}, t_{i-2})}{C(t_{i-2}, t_{i-1})} \) (5.42)
- **Bigrams**: \( \hat{P}(t_i|t_{i-1}) = \frac{C(t_i, t_{i-1})}{C(t_{i-1})} \) (5.43)
- **Unigrams**: \( \hat{P}(t_i) = \frac{C(t_i)}{N} \) (5.44)

How should these three estimators be combined in order to estimate the tri-
gram probability \( P(t_i|t_{i-1}, t_{i-2}) \)? The simplest method of combination is linear
interpolation. In linear interpolation, we estimate the probability
\( P(t_i|t_{i-1}, t_{i-2}) \) by a weighted sum of the unigram, bigram, and trigram probabilities:

\[ P(t_i|t_{i-1}, t_{i-2}) = \lambda_1 \hat{P}(t_i|t_{i-1}, t_{i-2}) + \lambda_2 \hat{P}(t_i|t_{i-1}) + \lambda_3 \hat{P}(t_i) \] (5.45)

We require \( \lambda_1 + \lambda_2 + \lambda_3 = 1 \), insuring that the resulting \( P \) is a probability
distribution. How should these \( \lambda \)s be set? One good way is **deleted interpolation**, developed by Jelinek and Mercer (1980). In deleted interpolation, we successively
delete each trigram from the training corpus, and choose the \( \lambda \)s so as to maximize
the likelihood of the rest of the corpus. The idea of the deletion is to set the \( \lambda \)
in such a way as to generalize to unseen data and not overfit the training corpus.
Fig. 5.19 gives the Brants (2000) version of the deleted interpolation algorithm for
tag trigrams.

Brants (2000) achieves an accuracy of 96.7% on the Penn Treebank with a
trigram HMM tagger. Weischedel et al. (1993) and DeRose (1988) have also
reported accuracies of above 96% for HMM tagging. (Thede and Harper, 1999)
function DELETED-INTERPOLATION(corpus) returns \( \lambda_1, \lambda_2, \lambda_3 \)

\[ \lambda_1 = 0 \]
\[ \lambda_2 = 0 \]
\[ \lambda_3 = 0 \]

\[ \text{foreach trigram } t_1, t_2, t_3 \text{ with } f(t_1, t_2, t_3) > 0 \]

\[ \text{depending on the maximum of the following three values} \]

\[ \text{case } C(t_1, t_2, t_3) \]
\[ \text{increment } \lambda_3 \text{ by } C(t_1, t_2, t_3) \]

\[ \text{case } C(t_2, t_3) \]
\[ \text{increment } \lambda_2 \text{ by } C(t_1, t_2, t_3) \]

\[ \text{case } N - 1 \]
\[ \text{increment } \lambda_1 \text{ by } C(t_1, t_2, t_3) \]

\[ \text{end} \]

\[ \text{end} \]

normalize \( \lambda_1, \lambda_2, \lambda_3 \)

return \( \lambda_1, \lambda_2, \lambda_3 \)

Figure 5.19 The deleted interpolation algorithm for setting the weights for combining unigram, bigram, and trigram tag probabilities. If the denominator is 0 for any case, we define the result of that case to be 0. \( N \) is the total number of tokens in the corpus. After Brants (2000).

offer a number of augmentations of the trigram HMM model, including the idea of conditioning word likelihoods on neighboring words and tags.

The HMM taggers we have seen so far are trained on hand-tagged data. Ku- piec (1992), Cutting et al. (1992), and others show that it is also possible to train an HMM tagger on unlabeled data, using the EM algorithm that we will introduce in Chapter 6. These taggers still start with a dictionary which lists which tags can be assigned to which words; the EM algorithm then learns the word likelihood function for each tag, and the tag transition probabilities. An experiment by Meraldo (1994), however, indicates that with even a small amount of training data, a tagger trained on hand-tagged data worked better than one trained via EM. Thus the EM-trained “pure HMM” tagger is probably best suited to cases where no training data is available, for example when tagging languages for which there is no previously hand-tagged data.

5.6 TRANSFORMATION-BASED TAGGING

Transformation-Based Tagging, sometimes called Brill tagging, is an instance of the Transformation-Based Learning (TBL) approach to machine learning (Brill, 1995), and draws inspiration from both the rule-based and stochastic taggers. Like the rule-based taggers, TBL is based on rules that specify what tags should be assigned to what words. But like the stochastic taggers, TBL is a machine learning technique, in which rules are automatically induced from the data. Like some but not all of the HMM taggers, TBL is a supervised learning technique, it assumes a pre-tagged training corpus.

Samuel et al. (1998) offer a useful analogy for understanding the TBL paradigm, which they credit to Terry Harvey. Imagine an artist painting a picture of a white house with green trim against a blue sky. Suppose most of the picture was sky, and hence most of the picture was blue. The artist might begin by using a very broad brush and painting the entire canvas blue. Next she might switch to a somewhat smaller white brush, and paint the entire house white. She would just color in the whole house, not worrying about the brown roof, or the blue windows or the green gables. Next she takes a smaller brown brush and colors over the roof. Now she takes up the blue paint on a small brush and paints in the blue windows on the house. Finally she takes a very fine green brush and does the trim on the gables.

The painter starts with a broad brush that covers a lot of the canvas but colors a lot of areas that will have to be repainted. The next layer colors less of the canvas, but also makes less “mistakes”. Each new layer uses a finer brush that corrects less of the picture, but makes fewer mistakes. TBL uses somewhat the same method as this painter. The TBL algorithm has a set of tagging rules. A corpus is first tagged using the broadest rule, that is, the one that applies to the most cases. Then a slightly more specific rule is chosen, which changes some of the original tags. Next an even narrower rule, which changes a smaller number of tags (some of which might be previously changed tags).

5.6.1 How TBL Rules Are Applied

Let’s look at one of the rules used by Brill’s (1995) tagger. Before the rules apply, the tagger labels every word with its most-likely tag. We get these most-likely tags from a tagged corpus. For example, in the Brown corpus, race is most likely to be a noun:

\[ P(\text{NN}|\text{race}) = 0.98 \]
\[ P(\text{VB}|\text{race}) = 0.02 \]

This means that the two examples of race that we saw above will both be coded as NN. In the first case, this is a mistake, as NN is the incorrect tag:

(5.46) is/VBZ expected/VBN to/TO race/NN tomorrow/NN

In the second case this race is correctly tagged as an NN:

(5.47) the/DT race/NN for/IN outer/JJ space/NN
Section 5.6. Transformation-Based Tagging

After selecting the most-likely tag, Brill’s tagger applies its transformation rules. As it happens, Brill’s tagger learned a rule that applies exactly to this mis-tagging of race:

Change NN to VB when the previous tag is TO

This rule would change race/NN to race/VB in exactly the following situation, since it is preceded by to/TO:

expected/VBN to/TO race/NN → expected/VBN to/TO race/VB

(5.48)

5.6.2 How TBL Rules Are Learned

Brill’s TBL algorithm has three major stages. It first labels every word with its most-likely tag. It then examines every possible transformation, and selects the one that results in the most improved tagging. Finally, it then re-tags the data according to this rule. The last two stages are repeated until some stopping criterion is reached, such as insufficient improvement over the previous pass. Note that stage two requires that TBL knows the correct tag of each word; that is, TBL is a supervised learning algorithm.

The output of the TBL process is an ordered list of transformations; these then constitute a “tagging procedure” that can be applied to a new corpus. In principle the set of possible transformations is infinite, since we could imagine transformations such as “transform NN to VB if the previous word was “IBM” and the word “the” occurs between 17 and 158 words before that”. But TBL needs to consider every possible transformation, in order to pick the best one on each pass through the algorithm. Thus the algorithm needs a way to limit the set of transformations. This is done by designing a small set of templates (abstracted transformations). Every allowable transformation is an instantiation of one of the templates. Brill’s set of templates is listed in Fig. 5.20. Fig. 5.21 gives the details of this algorithm for learning transformations.

The preceding (following) word is tagged z.
The word two before (after) is tagged z.
One of the two preceding (following) words is tagged z.
The preceding word is tagged z and the following word is tagged w.
The preceding (following) word is tagged z and the word two before (after) is tagged w.

Figure 5.20 Brill’s (1995) templates. Each begins with “Change tag a to tag b when . . . “. The variables a, b, z, and w range over parts-of-speech.
Section 5.7. Advanced Issues in Part-of-Speech Tagging

At the heart of Fig. 5.21 are the two functions GET\_BEST\_TRANSFORMATION and GET\_BEST\_INSTANCE. GET\_BEST\_TRANSFORMATION is called with a list of potential templates; for each template, it calls GET\_BEST\_INSTANCE. GET\_BEST\_INSTANCE iteratively tests every possible instantiation of each template by filling in specific values for the tag variables $a$, $b$, $z$, and $w$.

In practice, there are a number of ways to make the algorithm more efficient. For example, templates and instantiated transformations can be suggested in a data-driven manner; a transformation-instance might only be suggested if it would improve the tagging of some specific word. The search can also be made more efficient by pre-indexing the words in the training corpus by potential transformation. Roche and Schabes (1997) show how the tagger can also be speeded up by converting each rule into a finite-state transducer and composing all the transducers.

Fig. 5.22 shows a few of the rules learned by Brill’s original tagger.

<table>
<thead>
<tr>
<th>Change tags</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>VB</td>
<td>Previous tag is TO</td>
<td>to/TO race/NN $\rightarrow$ VB</td>
</tr>
<tr>
<td>2</td>
<td>VBP</td>
<td>VB</td>
<td>One of the previous 3 tags is MD</td>
<td>might/MD vanish/VBP $\rightarrow$ VB</td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>VB</td>
<td>One of the previous 2 tags is MD</td>
<td>might/MD not reply/NN $\rightarrow$ VB</td>
</tr>
<tr>
<td>4</td>
<td>VB</td>
<td>NN</td>
<td>One of the previous 2 tags is DT</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>VBD</td>
<td>VBN</td>
<td>One of the previous 3 tags is VBDZ</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.22 The first 20 nonlexicalized transformations from Brill (1995).

5.7 ADVANCED ISSUES IN PART-OF-SPEECH TAGGING

5.7.1 Practical Issues: Tag Indeterminacy and Tokenization

Tag indeterminacy arises when a word is ambiguous between multiple tags and it is impossible or very difficult to disambiguate. In this case, some taggers allow the use of multiple tags. This is the case in both the Penn Treebank and in the British National Corpus. Common tag indeterminacies include adjective versus preterite versus past participle (JJ/VBD/VBN), and adjective versus noun as prenominal modifier (JJ/NN). Given a corpus with these indeterminate tags, there are 3 ways to deal with tag indeterminacy when training and scoring part-of-speech taggers:

1. Somehow replace the indeterminate tags with only one tag.
2. In testing, count a tagger as having correctly tagged an indeterminate token if it gives either of the correct tags. In training, somehow choose only one of the tags for the word.

Table 5.7.1 HMM tagging experiments of Franz (1996). The row labels indicate correct tags, column labels indicate the tagger’s hypothesized tags, and each cell indicates percentage of the overall tagging error. Thus 4.4% of the total errors were caused by mistagging a VBD as a VBN. Common errors are boldfaced.

<table>
<thead>
<tr>
<th></th>
<th>IN</th>
<th>JJ</th>
<th>NN</th>
<th>NNP</th>
<th>RB</th>
<th>VBD</th>
<th>VBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>.2</td>
<td>-</td>
<td>3.3</td>
<td>2.1</td>
<td>1.7</td>
<td>.2</td>
<td>2.7</td>
</tr>
<tr>
<td>NN</td>
<td>8.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NNP</td>
<td>.2</td>
<td>3.3</td>
<td>4.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RB</td>
<td>2.2</td>
<td>2.0</td>
<td>.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VBD</td>
<td>3</td>
<td>.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VBN</td>
<td>2.8</td>
<td>-</td>
<td>-</td>
<td>4.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The confusion matrix above, and related error analyses in Franz (1996), Kupec (1992), and Ratnaparkhi (1996), suggest that some major problems facing current taggers are:

1. NN versus NNP versus JJ: These are hard to distinguish prenominally. Distinguishing proper nouns is especially important for information extraction and machine translation.
2. RP versus RB versus IN: All of these can appear in sequences of satellites immediately following the verb.
3. VBD versus VBN versus JJ: Distinguishing these is important for partial parsing (participles are used to find passives), and for correctly labeling the edges of noun-phrases.

Error analysis like this is a crucial part of any computational linguistic application. Error analysis can help find bugs, find problems in the training data, and, most important, help in developing new kinds of knowledge or algorithms to use in solving problems.
3. Treat the indeterminate tag as a single complex tag.

The second approach is perhaps the most sensible, although most previous published results seem to have used the third approach. This third approach applied to the Penn Treebank Brown corpus, for example, results in a much larger tagset of 85 tags instead of 45, but the additional 40 complex tags cover a total of only 121 word instances out of the million word corpus.

Most tagging algorithms assume a process of tokenization has been applied to the tags. Ch. 5 discussed the issue of tokenization of periods for distinguishing sentence-final periods from word-internal period in words like etc.. An additional role for tokenization is in word splitting. The Penn Treebank and the British National Corpus split contractions and the ’s-genitive from their stems:

\[
\text{would}/\text{MD} \text{a}/\text{DT} \text{r}/\text{br}
\]

\[
\text{children}/\text{NNS} \text{’s}/\text{POS}
\]

Indeed, the special Treebank tag POS is used only for the morpheme ’s which must be segmented off during tokenization.

Another tokenization issue concerns multi-part words. The Treebank tagset assumes that tokenization of words like New York is done at whitespace. The phrase a New York City firm is tagged in Treebank notation as five separate words: \text{a}/\text{DT} \text{New}/\text{NNP} \text{York}/\text{NNP} \text{City}/\text{NNP} \text{firm}/\text{NN}. The C5 tagset, by contrast, allow prepositions like ”in terms of” to be treated as a single word by adding numbers to each tag, as in \text{in}/\text{JJ}\text{31 terms}/\text{JJ}\text{32 of}/\text{JJ}\text{33}.

5.7.2 Unknown Words

All the tagging algorithms we have discussed require a dictionary that lists the possible parts-of-speech of every word. But the largest dictionary will still not contain every possible word, as we saw in Ch. 4. Proper names and acronyms are created very often, and even new common nouns and verbs enter the language at a surprising rate. Therefore in order to build a complete tagger we cannot always use a dictionary to give us \( p(w_i | t) \). We need some method for guessing the tag of an unknown word.

The simplest possible unknown-word algorithm is to pretend that each unknown word is ambiguous among all possible tags, with equal probability. Then the tagger must rely solely on the contextual POS-trigrams to suggest the proper tag. A slightly more complex algorithm is based on the idea that the probability distribution of tags over unknown words is very similar to the distribution of tags over words that occurred only once in a training set, an idea that was suggested by both Baayen and Sprout (1996) and Dermatas and Kokkinakis (1995). These words that only occur once are known as \textit{hapax legomena} (singular \textit{hapax legomenon}).

For example, unknown words and \textit{hapax legomena} are similar in that they are both most likely to be nouns, followed by verbs, but are very unlikely to be determiners or interjections. Thus the likelihood \( p(w_i | t) \) for an unknown word is determined by the average of the distribution over all singleton words in the training set. This idea of using “things we’ve seen once” as an estimator for “things we’ve never seen” will prove useful in the Good-Turing algorithm of Ch. 6.)

Most unknown-word algorithms, however, make use of a much more powerful source of information: the morphology of the words. For example, words that end in -s are likely to be plural nouns (NNS), words ending with -ed tend to be past participles (VBN), words ending with able tend to be adjectives (JJ), and so on. Even if we’ve never seen a word, we can use facts about its morphological form to guess its part-of-speech. Besides morphological knowledge, orthographic information can be very helpful. For example words starting with capital letters are likely to be proper nouns (NP). The presence of a hyphen is also a useful feature: hyphenated words in the Treebank version of Brown are most likely to be adjectives (JJ). This prevalence of JJs is caused by the labeling instructions for the Treebank, which specified that prenominal modifiers should be labeled as JJ if they contained a hyphen.

How are these features combined and used in part-of-speech taggers? One method is to train separate probability estimators for each feature, assume independence, and multiply the probabilities. Weischedel et al. (1993) built such a model, based on four specific kinds of morphological and orthographic features. They used 3 inflectional endings (-ed, -s, -ing), 32 derivational endings (such as -ion, -al, -ive, and -ly), 4 values of capitalization depending on whether a word is sentence-initial (+/- capitalization, +/- initial) and whether the word was hyphenated. For each feature, they trained maximum likelihood estimates of the probability of the feature given a tag from a labeled training set. They then combined the features to estimate the probability of an unknown word by assuming independence and multiplying:

\[
P(w_i | t) = p(\text{unknown-word}|t) \times p(\text{capital}|t) \times p(\text{endings/hyphen}|t) \quad (5.49)
\]

Another HMM-based approach, due to Samuelsson (1993) and Brants (2000), generalizes this use of morphology in a data-driven way. In this approach, rather than pre-selecting certain suffixes by hand, all final letter sequences of all words are considered. They consider such suffixes of up to ten letters, computing for each suffix of length \( t \) the probability of the tag \( t \) given the suffix:

\[
P(t_j | t_{j-1} \ldots t_0) \quad (5.50)
\]

These probabilities are smoothed using successively shorter and shorter suffixes. Separate suffix tries are kept for capitalized and uncapitalized words.

In general, most unknown word models try to capture the fact that unknown words are unlikely to be closed-class words like prepositions. Brants models this...
fact by only computing suffix probabilities from the training set for words whose frequency in the training set is \( \leq 10 \). In the HMM tagging model of Thede and Harper (1999), this fact is modeled instead by only training on open-class words.

Note that (5.50) gives an estimate of \( p(t_i|w_i) \); since for the HMM tagging approach we need the likelihood \( p(w_i|t_i) \), this can be derived from (5.50) using Bayesian inversion (i.e. using Bayes rule and computation of the two priors \( P(t_i) \) and \( P(t_i, w_{i-1}, \ldots, w_1) \)).

In addition to using capitalization information for unknown words, Brants (2000) also uses capitalization information for tagging known words, by adding a capitalization feature to each tag. Thus instead of computing \( P(t_i|c_{i-1}) \) as in (5.43), he actually computes the probability \( P(t_i, c_{i-1}, \ldots, c_1|w_1, \ldots, w_{i-1}) \). This is equivalent to having a capitalized and uncapsulated version of each tag, essentially doubling the size of the tagset.

A non-HMM based approach to unknown word detection was that of Brill (1995) using the TBL algorithm, where the allowable templates were defined orthographically (the first \( N \) letters of the words, the last \( N \) letters of the word, etc.).

A third approach to unknown word handling, due to Ratnaparkhi (1996), uses the maximum entropy approach. The maximum entropy approach is one of a family of loglinear classifiers in which many features are computed for the word to be tagged, and all the features are combined in a regression-like model. For each word, the Ratnaparkhi (1996) model includes as features all prefixes and suffixes of length \( \leq 4 \) (i.e. 8 total prefix and suffix features), plus three more features indicating whether the word contains a number, contains a hyphen, or contains an upper-case letter. The model ignored all features with counts less than 10.

A more recent loglinear model, Toutanova et al. (2003) augmented the Ratnaparkhi features with an ‘all-caps’ feature, as well as a feature for words that are capitalized and have a digit and dash (since words like CPC-12 tend to be common nouns). But the most significant unknown word improvement of the Toutanova et al. (2003) model is a simple company name detector, which marks capitalized words followed within 3 words by a word like Co. or Inc.

Loglinear models have also been applied to Chinese tagging by Tseng et al. (2005). Chinese words are very short (around 2.4 characters per unknown word compared with 7.7 for English), but Tseng et al. (2005) found that morphological features nonetheless gave a huge increase in tagging performance for unknown words. For example for each character in an unknown word and each POS tag, they added a binary feature indicating whether that character ever occurred with that tag in any training set word. There is also an interesting distributional difference in unknown words between Chinese and English. While English unknown words tend to be proper nouns (41% of unknown words in WSJ are NP), in Chinese the majority of unknown words are common nouns and verbs (61% in the Chinese TreeBank 5.0). These ratios are similar to German, and seem to be caused by the prevalence of compounding as a morphological device in Chinese and German.

5.7.3 Part-of-Speech Tagging for Other Languages

As the previous paragraph suggests, part-of-speech tagging algorithms have all been applied to many other languages as well. In some cases, the methods work well without large modifications; Brants (2000) showed the exact same performance for tagging on the German NEGRA corpus (96.7%) as on the English Penn Treebank. But a number of augmentations and changes become necessary when dealing with highly inflected or agglutinative languages.

One problem with these languages is simply the large number of words, when compared to English. Recall from Ch. 3 that agglutinative languages like Turkish (and to some extent mixed agglutinative-inflectional languages like Hungarian) are those in which words contain long strings of morphemes, where each morpheme has relatively few surface forms, and so it is often possible to clearly see the morphemes in the surface text. For example Megyesi (1999) gives the following typical example of a Hungarian word meaning "of their hits":

\[
(5.51) \text{találataiknak}
\]

- talál - hit/find
- at - nominalizer
- a - his
- -i - poss.plur
- -nak - their dat/gen

"of their hits"

Similarly, the following list, excerpted from Hakkani-Tür et al. (2002), shows a few of the words producible in Turkish from the root uyu- 'sleep':

- uyuyorum 'I am sleeping'
- uyuyorsun 'you are sleeping'
- uyuduk 'we slept'
- uyumadan 'without sleeping'
- uyuman 'your sleeping'
- uyurken 'while (somebody) is sleeping'
- uyutmak 'to cause someone to sleep'
- uyurumak 'to cause someone to cause another person to sleep'

These productive word-formation processes result in a large vocabulary for these languages. Oravecz and Dienes (2002), for example, show that a quarter-million word corpus of English has about 19,000 different words (i.e. word types); the same size corpus of Hungarian has almost 50,000 different words. This problem continues even with much larger corpora; note in the table below on Turkish from Hakkani-Tür et al. (2002) that the vocabulary size of Turkish is far bigger than that of English and is growing faster than English even at 10 million words.
Vocabulary Size

<table>
<thead>
<tr>
<th>Corpus Size</th>
<th>Turkish</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>1M words</td>
<td>106,547</td>
<td>33,398</td>
</tr>
<tr>
<td>10M words</td>
<td>417,775</td>
<td>97,734</td>
</tr>
</tbody>
</table>

The large vocabulary size seems to cause a significant degradation in tagging performance when the HMM algorithm is applied directly to agglutinative languages. For example Oravecz and Dienes (2002) applied the exact same HMM software (called ‘TrnT’) that Brants (2000) used to achieve 96.7% on both English and German, and achieved only 92.88% on Hungarian. The performance on known words (98.32%) was comparable to English results; the problem was the performance on unknown words: 67.07% on Hungarian, compared to around 84-85% for unknown words with a comparable amount of English training data. Hajič (2000) notes the same problem in a wide variety of other languages (including Czech, Slovene, Estonian, and Romanian); the performance of these taggers is hugely improved by adding a dictionary which essentially gives a better model of unknown words. In summary, one difficulty in tagging highly inflected and agglutinative languages is tagging of unknown words.

A second, related issue with such languages is the vast amount of information that is coded in the morphology of the word. In English, lots of information about syntactic function of a word is represented by word order, or neighboring function words. In highly inflectional languages, information such as the case (nominative, accusative, genitive) or gender (masculine, feminine) is marked on the words themselves, and word order plays less of a role in marking syntactic function. Since tagging is often used a preprocessing step for other NLP algorithms such as parsing or information extraction, this morphological information is crucial to extract. This means that a part-of-speech tagging output for Turkish or Czech needs to include information about the case and gender of each word in order to be as useful as parts-of-speech without case or gender are in English.

For this reason, tagsets for agglutinative and highly inflectional languages are usually much larger than the 50-100 tags we have seen for English. Tags in such enriched tagsets are sequences of morphological tags rather than a single primitive tag. Assigning tags from such a tagset to words means that we are jointly solving the problems of part-of-speech tagging and morphological disambiguation. Hakkani-Tür et al. (2002) give the following example of tags from Turkish, in which the word *izin* has three possible morphological/part-of-speech tags (and meanings):

1. **Yerdeki izin temizlenmesi gerek.** *izin + Noun+A3sg+Pnon+Gen*
   *The trace on the floor should be cleaned.*

2. **Üzerinde parmak izin kalmış.** *izin + Noun+A3sg+Pnon+Nom*
   *Your fingerprint is left on (it).*

With such large tagsets, it is generally necessary to perform morphological analysis on each word to generate the list of possible morphological tag sequences (i.e. the list of possible part-of-speech tags) for the word. The role of the tagger is then to disambiguate among these tags. The morphological analysis can be done in various ways. The Hakkani-Tür et al. (2002) model of Turkish morphological analysis is based on the two-level morphology we introduced in Ch. 3. For Czech and the MULTEXT-East languages, Hajič (2000) and Hajič and Hadlák (1998) use a fixed external dictionary for each language which compiles out all the possible forms of each word, and lists possible tags for each wordform. The morphological parse also crucially helps address the problem of unknown words, since morphological parsers can accept unknown stems and still segment the affixes properly.

Given such a morphological parse, various methods for the tagging itself can be used. The Hakkani-Tür et al. (2002) model for Turkish uses a Markov model of tag sequences. The model assigns a probability to sequences of tags like *izin+Noun+A3sg+Pnon+Gen* by computing tag transition probabilities from a training set. Other models use similar techniques to those for English. Hajič (2000) and Hajič and Hadlák (1998), for example, use a log-linear exponential tagger for the MULTEXT-East languages, Oravecz and Dienes (2002) and Džeroski et al. (2000) use the TrnT HMM tagger (Brants, 2000), and so on.

### Combining Taggers

The various part-of-speech tagging algorithms we have described can also be combined. The most common approach to tagger combination is to run multiple taggers in parallel on the same sentence, and then combine their output, either by...
voting or by training another classifier to choose which tagger to trust in a given context. Brill and Wu (1998), for example, combined unigram, HMM, TBL, and maximum-entropy taggers by voting via a higher-order classifier, and showed a small gain over the best of the four classifiers. In general, this kind of combination is only useful if the taggers have complementary errors, and so research on combination often begins by checking to see if the errors are indeed different from different taggers. Another option is to combine taggers in series. Hajić et al. (2001) apply this option for Czech, using the rule-based approach to remove some of the impossible tag possibilities for each word, and then an HMM tagger to choose the best sequence from the remaining tags.

5.8 SUMMARY

This chapter introduced the idea of parts-of-speech and part-of-speech tagging. The main ideas:

- Languages generally have a relatively small set of closed class words, which are often highly frequent, generally act as function words, and can be very ambiguous in their part-of-speech tags. Open class words generally include various kinds of nouns, verbs, adjectives. There are a number of part-of-speech coding schemes, based on tagsets of between 40 and 200 tags.
- Part-of-speech tagging is the process of assigning a part-of-speech label to each of a sequence of words. Rule-based taggers use hand-written rules to distinguish tag ambiguity. HMM taggers choose the tag sequence which maximizes the product of word likelihood and tag sequence probability. Other machine learning models used for tagging include maximum entropy and other log-linear models, decision trees, memory-based learning, and transformation-based learning.
- The probabilities in HMM taggers are trained on hand-labeled training corpora, combining different N-gram levels using deleted interpolation, and using sophisticated unknown word models.
- Given an HMM and an input string, the Viterbi algorithm is used to decode the optimal tag sequence.
- Taggers are evaluated by comparing their output from a test set to human labels for that test set. Error analysis can help pinpoint areas where a tagger doesn’t perform well.

**BIBLIOGRAPHICAL AND HISTORICAL NOTES**

The earliest implemented part-of-speech assignment algorithm may have been part of the parser in Zellig Harris’s Transformations and Discourse Analysis Project (TDAP), which was implemented between June 1958 and July 1959 at the University of Pennsylvania (Harris, 1962). Previous natural language processing systems had used dictionaries with part-of-speech information for words, but have not been described as performing part-of-speech disambiguation. As part of its parsing, TDAP did part-of-speech disambiguation via 14 hand-written rules, whose use of part-of-speech tag sequences prefigures all the modern algorithms, and which were run in an order based on the relative frequency of tags for a word. The parser/tagger was reimplemented recently and is described by Joshi and Hopely (1999) and Karpunen (1999), who note that the parser was essentially implemented (ironically in a very modern way) as a cascade of finite-state transducers.

Soon after the TDAP parser was the Computational Grammar Coder (CGC) of Klein and Simmons (1963). The CGC had three components: a lexicon, a morphological analyzer, and a context disambiguator. The small 1500-word lexicon included exceptional words that could not be accounted for in the simple morphological analyzer, including function words as well as irregular nouns, verbs, and adjectives. The morphological analyzer used inflectional and derivational suffixes to assign part-of-speech classes. A word was run through the lexicon and morphological analyzer to produce a candidate set of parts-of-speech. A set of 500 context rules were then used to disambiguate this candidate set, by relying on surrounding islands of unambiguous words. For example, one rule said that between an ARTICLE and a VERB, the only allowable sequences were ADJ-NOUN, NOUN-ADVERB, or NOUN-NOUN. The CGC algorithm reported 90% accuracy on applying a 30-tag tagset to articles from the Scientific American and a children’s encyclopedia.

The TAGGIT tagger (Greene and Rubin, 1971) was based on the Klein and Simmons (1963) system, using the same architecture but increasing the size of the dictionary and the size of the tagset (to 87 tags). For example the following sample rule, which states that a word x is unlikely to be a plural noun (NNS) before a third person singular verb (VBZ):

\[ x \text{ VBZ} \rightarrow \text{not NNS} \]

TAGGIT was applied to the Brown corpus and, according to Francis and Kučera (1982, p. 9), “resulted in the accurate tagging of 77% of the corpus” (the remainder of the Brown corpus was tagged by hand).

In the 1970s the Lancaster-Oslo/Bergen (LOB) corpus was compiled as a
Section 5.8. Summary

British English equivalent of the Brown corpus. It was tagged with the CLAWS tagger (Marshall, 1983, 1987; Garside, 1987), a probabilistic algorithm which can be viewed as an approximation to the HMM tagging approach. The algorithm used tag bigram probabilities, but instead of storing the word-likelihood of each tag, tags were marked either as rare \( P(\text{tag} | \text{word}) < .01 \) infrequent \( P(\text{tag} | \text{word}) < .10 \), or normally frequent \( P(\text{tag} | \text{word}) > .10 \).

The probabilistic PARTS tagger of Church (1988) was very close to a full HMM tagger. It extended the CLAWS idea to assign full lexical probabilities to each word/tag combination, and used Viterbi decoding to find a tag sequence. Like the CLAWS tagger, however, it stored the probability of the tag given the word:

\[
P(\text{tag} | \text{word}) \times P(\text{tag} | \text{previous n tags}) \tag{5.52}
\]

rather than using the probability of the word given the tag, as an HMM tagger does:

\[
P(\text{word} | \text{tag}) \times P(\text{tag} | \text{previous n tags}) \tag{5.53}
\]

Later taggers explicitly introduced the use of the Hidden Markov Model, often with the EM training algorithm (Kupiec, 1992; Merialdo, 1994; Weischedel et al., 1993), including the use of variable-length Markov models (Schütze and Singer, 1994).

Most recent tagging algorithms, like the HMM and TBL approaches we have discussed, are machine-learning classifiers which estimate the best tag-sequence for a sentence given various features such as the current word, neighboring parts-of-speech or words, and unknown word features such as orthographic and morphological features. Many kinds of classifiers have been used to combine these features, including decision trees (Jelinek et al., 1994; Magerman, 1995), maximum entropy models (Ratnaparkhi, 1996), other log-linear models (Franz, 1996), memory-based learning (Daelemans et al., 1996), and networks of linear separators (SNOW) (Roth and Zelenko, 1998). Most machine learning models seem to achieve relatively similar performance given similar features, roughly 96-97% on the Treebank 45-tag tagset on the Wall Street Journal corpus. As of the writing of this chapter, the highest performing published model on this WSJ Treebank task is a log-linear tagger that uses information about neighboring words as well as tags, and a sophisticated unknown-word model, achieving 97.24% accuracy (Toutanova et al., 2003). Most such models are supervised; unsupervised models are considerably less developed. Brill (1997), for example, gives an unsupervised version of the TBL algorithm.

Readers interested in the history of parts-of-speech should consult a history of linguistics such as Robins (1967) or Koerner and Asher (1995), particularly the article by Householder (1995) in the latter. Sampson (1987) and Garside et al. (1997) give a detailed summary of the provenance and makeup of the Brown and other tagsets. More information on part-of-speech tagging can be found in van Halteren (1999).

EXERCISES

5.1 Find one tagging error in each of the following sentences that are tagged with the Penn Treebank tagset:

a. I/PRP need/VBP a/DT flight/NN from/IN Atlanta/NN
b. Does/VBZ this/DT flight/NN serve/VB dinner/NNS
c. I/PRP have/VB a/DT friend/NN living/VBG in/IN Denver/NNP
d. What/WDT flights/NNS do/VBP you/PRP have/VB from/IN Milwaukee/NNP to/IN Tampa/NNP
e. Can/VBP you/PRP list/VB the/DT nonstop/JJ afternoon/NN flights/NNS

5.2 Use the Penn Treebank tagset to tag each word in the following sentences from Damon Runyon’s short stories. You may ignore punctuation. Some of these are quite difficult; do your best.

a. I ti san i c en i g h t .
b. This crap game is over a garage in Fifty-second Street . . .
c. ...Nobody ever takes the newspapers she sells . . .
d. He is a tall, skinny guy with a long, sad, mean-looking kisser, and a mournful voice.
e. . .I am sitting in Mindy’s restaurant putting on the gefillte fish, which is a dish I am very fond of, . . .
f. When a guy and a doll get to taking peeks back and forth at each other, why there you are indeed.

5.3 Now compare your tags from the previous exercise with one or two friend’s answers. On which words did you disagree the most? Why?

5.4 Now tag the sentences in Exercise 5.2 using the more detailed Brown tagset in Fig. 5.7.

5.5 Implement the TBL algorithm in Fig. 5.21. Create a small number of templates and train the tagger on any POS-tagged training set you can find.
5.6 Implement the “most-likely tag” baseline. Find a POS-tagged training set, and use it to compute for each word the tag which maximizes \( p(t|w) \). You will need to implement a simple tokenizer to deal with sentence boundaries. Start by assuming all unknown words are NN and compute your error rate on known and unknown words. Now write at least 5 rules to do a better job of tagging unknown words, and show the difference in error rates.

5.7 Recall that the Church (1988) tagger is not an HMM tagger since it incorporates the probability of the tag given the word:

\[
P(tag|word) \times P(tag|previous \ n \ tags)
\]

rather than using the likelihood of the word given the tag, as an HMM tagger does:

\[
P(word|tag) \times P(tag|previous \ n \ tags)
\]

As a gedanken-experiment, construct a sentence, a set of tag transition probabilities, and a set of lexical tag probabilities that demonstrate a way in which the HMM tagger can produce a better answer than the Church tagger.

5.8 Build an HMM tagger. This requires (1) that you have implemented the Viterbi algorithm from Ch. 5 or Chapter 6, (2) that you have a dictionary with part-of-speech information and (3) that you have either (a) a part-of-speech-tagged corpus or (b) an implementation of the Forward Backward algorithm. If you have a labeled corpus, train the transition and observation probabilities of an HMM tagger directly on the hand-tagged data. If you have an unlabeled corpus, train using Forward Backward.

5.9 Now run your algorithm on a small test set that you have hand-labeled. Find five errors and analyze them.


Gil, D. (2000). Syntactic categories, cross-linguistic variation and universal gram-
Proceedings of the Third International Conference on Language
Hungarian. In
for Computational Linguistics.
, College Park, Maryland, pp. 175–182. Association
Computational Linguistics
Proceedings of the 37th conference on Association for
part-of-speech tagging. In
.
, Las Palmas, Canary Islands, Spain, pp. 710–717.
Proceedings of the Conference on Empirical Methods in Natural Language Processing,
University of Pennsylvania, pp. 133–142. ACL.
Robins, R. H. (1967). A Short History of Linguistics. Indiana University Press,
Bloomington.
Roche, E. and Schabes, Y. (1997). Deterministic part-of-speech tagging with finite-
state transducers. In Roche, E. and Schabes, Y. (Eds.), Finite-State Language
separators. In COLING/ACL-98, Montreal, pp. 1136–1142. ACL.
Garside, R., Leech, G., and Sampson, G. (Eds.), The Computational Analysis of English,
from features with transformation-based learning. In
Chu-Carroll, J. and Green, N. (Eds.), Applying Machine Learning to Discourse Processing. Papers from the
In
9th Nordic Conference on Computational Linguistics NODALIDA-93.
Stockholm.
annotated corpus of English: The Penn treebank. Computational Linguistics
19(2), 313–330.
Proceedings of ACL-95, Cambridge, MA, pp. 276–283. ACL.
annotated corpus of English: The Penn treebank. Computational Linguistics
19(2), 313–330.
coding of English words. Journal of the Association for Computing Machinery,
10(3), 334–347.
Sankoff, D. and
Kruskal, J. B. (Eds.), Time Warps, String Edits, and Macromolecules: The Theory
and Practice of Sequence Comparison, pp. 1–44. Addison-Wesley, Reading, MA.
Kučera, H. and Francis, W. N. (1967). Computational analysis of present-day
American English. Brown University Press, Providence, RI.
Computer Speech and Language, 6, 225–242.
Computer Speech and Language, 6, 225–242.
Sankoff, D. and
Kruskal, J. B. (Eds.), Time Warps, String Edits, and Macromolecules: The Theory
and Practice of Sequence Comparison, pp. 1–44. Addison-Wesley, Reading, MA.
Kučera, H. and Francis, W. N. (1967). Computational analysis of present-day
American English. Brown University Press, Providence, RI.
Sankoff, D. and
Kruskal, J. B. (Eds.), Time Warps, String Edits, and Macromolecules: The Theory
and Practice of Sequence Comparison, pp. 1–44. Addison-Wesley, Reading, MA.
Kučera, H. and Francis, W. N. (1967). Computational analysis of present-day
American English. Brown University Press, Providence, RI.
Computer Speech and Language, 6, 225–242.
Sankoff, D. and
Kruskal, J. B. (Eds.), Time Warps, String Edits, and Macromolecules: The Theory
and Practice of Sequence Comparison, pp. 1–44. Addison-Wesley, Reading, MA.
Kučera, H. and Francis, W. N. (1967). Computational analysis of present-day
American English. Brown University Press, Providence, RI.
Sankoff, D. and
Kruskal, J. B. (Eds.), Time Warps, String Edits, and Macromolecules: The Theory
and Practice of Sequence Comparison, pp. 1–44. Addison-Wesley, Reading, MA.
Kučera, H. and Francis, W. N. (1967). Computational analysis of present-day
American English. Brown University Press, Providence, RI.
Section 5.8. Summary


