6

Part-of-Speech Tagging Using Rules

6.1 Resolving Part-of-Speech Ambiguity

6.1.1 A Manual Method

We saw that looking up a word in a lexicon or carrying out a morphological analysis on it can leave it with an ambiguous part of speech. The word *chair*, which can be assigned two tags, noun or verb, is an example of ambiguity. It is a noun in the phrase *a chair*, and a verb in *to chair a session*. Ambiguity resolution, that is, retaining only one part of speech (POS) and discarding the others, is generally referred to as POS tagging or POS annotation.

As children we learned to carry out a manual disambiguation by considering the grammatical context of the word. In the first phrase, *chair* is preceded by an article and therefore is part of a noun phrase. Since there is no other word here, *chair* is a noun. In the second phrase, *chair* is preceded by *to*, which would not precede a noun, and therefore is a verb.

Voutilainen and Järvinen (1995) describe a more complex example with the sentence

That round table might collapse.

While the correct part-of-speech tagging is:

That/determiner *round*/adjective *table*/noun *might*/modal verb *collapse*/verb.

a simple dictionary lookup or a morphological analysis produces many ambiguities, as shown in Table 6.1.

6.1.2 Which Method to Use to Automatically Assign Parts of Speech

Grammatical constraints are not always sufficient to resolve ambiguous tags. Church and Mercer (1993) exemplify this with the phrase *I see a bird*, which can be annotated as

Table 6.1. Ambiguities in part-of-speech annotation with the sentence: *That round table might collapse*.

Words	Possible tags	Example of use
that	Subordinating conjunction	*That he can swim is good*
	Determiner	*That white table*
	Adverb	*It is not that easy*
	Pronoun	*That is the table*
	Relative pronoun	*The table that collapsed*
round	Verb	*Round up the usual suspects*
	Preposition	*Turn round the corner*
	Noun	*A big round*
	Adjective	*A round box*
	Adverb	*He went round*
table	Noun	*That white table*
	Verb	*I table that*
might	Noun	*The might of the wind*
	Modal verb	*She might come*
collapse	Noun	*The collapse of the empire*
	Verb	*The empire can collapse*

I/noun *see*/noun *a*/noun *bird*/noun

This tagging corresponds to: *I*/letter of alphabet, *see*/noun as in *Holy See*, *a*/letter of alphabet, *bird*/noun. Although, this tag sequence makes no sense here, it cannot be ruled out as syntactically ill formed, because the parser must accept sequences of four nouns in other situations, as in *city school committee meeting*. The proper tagging is, of course, *I*/pronoun *see*/verb *a*/article *bird*/noun.

Semantic rules could implement common-sense reasoning and prevent inconsistencies. However, this method is no longer favored. It would imply writing many rules that could operate in very specific applications, and not on unrestricted texts.

Instead of using general grammar rules, we can consider word preferences. Most words taken from a dictionary have only one part of speech or have a strong preference for only one of them, although frequent words tend to be more ambiguous. From text statistics based on different corpora, in English and in French, Merialdo (1994) and Vergne (1999) report that 50% to 60% of words have a unique possible tag and 15% to 25% have only two tags. In both languages, tagging a word with its most common part of speech yields a success rate of more than 75%. Charniak (1993) reports a score of more than 90% for English. This figure is called the **baseline**. It corresponds to the accuracy obtained with a minimal algorithm, here the word annotation with its most frequent tag.

Two efficient methods applied locally have emerged to improve this figure and to solve reasonably well POS tagging. The first one uses rule-based constraints. Rules consider the left and right context of the word to disambiguate, that is, either discard or replace a wrong part of speech. Rules are symbolic and can be designed by hand or derived automatically from hand-annotated corpora.

The second method is based on statistics. Sequence statistics are automatically learned from hand-annotated corpora, and probabilistic models are applied that assign the most likely tags to words of a sentence. Both methods enable to tag successfully more than 95% of the words of a text. We will describe the first one in this chapter and the second one in the next chapter.

6.2 Tagging with Rules

Part-of-speech tagging with rules is relatively old (Klein and Simmons 1963). In the beginning, rules were hand-coded and yielded good results at the expense of thoroughly and painfully crafting the rules (Voutilainen el al. 1992). The field has been completely renewed by Brill (1995), who proposed a very simple scheme to tag a text with rules and an algorithm to learn automatically the rules from annotated corpora. A good deal of the current work on part-of-speech tagging with rules is now inspired by his foundational work.

6.2.1 Brill's Tagger

Brill's tagger uses a dictionary and assumes that it contains all the words to tag. Each word in the dictionary is labeled with its most likely (frequent) part of speech and includes the list of its other legal – possible – parts of speech. Part-of-speech distributions and statistics for each word can be derived from annotated corpora and using methods described in Chaps. 2 and 4.

The tagger first assigns each word with its most likely part of speech. It does not depend on a morphological parser, although it could use one as a preprocessor. It also features a module to tag unknown words that we will examine in Sect. 6.3. Examples of likely tags assigned to words are given in Table 6.2.

Table 6.2. Initial step of Brill's algorithm.

	Likely tags yielding a correct tagging	**Likely tags yielding a wrong tagging**
English	*I*/pro *can*/modal *see*/verb *a*/art *bird*/noun	*The*/art *can*/**modal** *rusted*/verb
French	*Je*/pro *donne*/verb *le*/art *cadeau*/noun	*Je*/pro *le*/**art** *fais*/verb *demain*/adv
German	*Der*/art *Mann*/noun *kommt*/verb	*Wer*/pro *ist*/verb *der*/art *Mann*/noun , *der*/**art** *kommt*/verb *?*

The tagger then applies a list of transformations to alter the initial tagging. Transformations are contextual rules that rewrite a word tag into a new one. The transformation is performed only if the new tag of the word is legal – is in the dictionary. If so, the word is assigned the new tag. Transformations are executed sequentially and each transformation is applied to the text from left to right. Examples of transformations are:

1. In English: Change the tag from modal to noun if the previous word is an article.
2. In French: Change the tag from article to pronoun if the previous word is a pronoun.
3. In German: Change the tag from article to pronoun if the previous word is a noun (or a comma.)

These rules applied to the sentences in Table 6.2 yield:

1. In English: *The*/art *can*/noun *rusted*/verb
2. In French: *Je*/pro *le*/pro *fais*/verb *demain*/adv
3. In German: *Wer*/pro *ist*/verb *der*/art *Mann*/noun *, der*/pro *kommt*/verb *?*

Rules conform to a limited number of transformation types, called templates. For example, the rule

Change the tag from modal to noun if the previous word is an article.

corresponds to template:

Change the tag from X to Y if the previous tag is Z.

The tagger uses in total 11 templates shown in Table 6.3. Brill reports that less than 500 rules – instantiated templates – are needed in English to obtain an accuracy of 97%.

Table 6.3. Contextual rule templates, where A, B, C, and D denotes parts of speech, members of the POS tagset.

Rules	Explanation
`alter(A, B, prevtag(C))`	Change A to B if preceding tag is C
`alter(A, B, nexttag(C))`	Change A to B if the following tag is C
`alter(A, B, prev2tag(C))`	Change A to B if tag two before is C
`alter(A, B, next2tag(C))`	Change A to B if tag two after is C
`alter(A, B, prev1or2tag(C))`	Change A to B if one of the two preceding tags is C
`alter(A, B, next1or2tag(C))`	Change A to B if one of the two following tags is C
`alter(A, B, prev1or2or3tag(C))`	Change A to B if one of the three preceding tags is C
`alter(A, B, next1or2or3tag(C))`	Change A to B if one of the three following tags is C
`alter(A, B, surroundingtag(C, D))`	Change A to B if surrounding tags are C and D
`alter(A, B, nextbigram(C, D))`	Change A to B if next bigram tag is C D
`alter(A, B, prevbigram(C, D))`	Change A to B if previous bigram tag is C D

6.2.2 Implementation in Prolog

We will exemplify the tagging algorithm with an implementation of two rule templates:

```
alter(A, B, prevtag(C))
alter(A, B, prev1or2tag(C))
```

These rules being instantiated under the form of:

```
alter(verb, noun, prevtag(art)).
alter(verb, noun, prev1or2tag(art)).
```

The first rule changes the tag from `verb` to `noun` if the previous word is an `article`, and the second changes the tag from `verb` to `noun` if one of the two previous words is an `article`. The second rule is more general that the first one. We give the code of the first one because it is easier to start with it.

The `tag` predicate enables us to alter an initially tagged text:

```
?- tag([the/art, holy/adj, see/verb], L).
L = [the/art, holy/adj, see/noun]

% tag(+InitialTaggedText, -TaggedText)
% Implementation of Brill's algorithm

tag(InitialTaggedText, TaggedText) :-
  bagof(alter(FromPOS, ToPOS, Condition),
  alter(FromPOS, ToPOS, Condition), Rules),
  forall(Rules, InitialTaggedText, TaggedText).

% Collect all the rules and apply them sequentially

forall([Rule | Rules], Text, TaggedText) :-
  apply(Rule, Text, AlteredText),
  forall(Rules, AlteredText, TaggedText).
forall([], TaggedText, TaggedText).

%Apply prevtag template
apply(alter(FromPOS, ToPOS, prevtag(POS)),
  [PrevWord/POS, Word/FromPOS | RemainingText],
  [PrevWord/POS, Word/ToPOS | RemainingText1] ) :-
  !,
  apply(alter(FromPOS, ToPOS, prevtag(POS)),
    [Word/ToPOS | RemainingText],
    [Word/ToPOS | RemainingText1] ).
apply(alter(FromPOS, ToPOS, prevtag(POS)),
  [X, Y| RemainingText], [X, Y| RemainingText1] ) :-
```

```
    apply(alter(FromPOS, ToPOS, prevtag(POS)),
      [Y| RemainingText], [Y | RemainingText1] ).
  apply(alter(_, _, prevtag(_)), [X], [X]).

  % Apply prev1or2tag template
  % The first two rules take into account that the rule
  % can apply to the second word of the text
  apply(alter(FromPOS, ToPOS, prev1or2tag(POS)),
    [FirstWord/POS, Word/FromPOS | RemainingText],
    [FirstWord/POS, Word/ToPOS | RemainingText1] ) :-
    apply_aux(alter(FromPOS, ToPOS, prev1or2tag(POS)),
      [FirstWord/POS, Word/ToPOS | RemainingText],
      [FirstWord/POS, Word/ToPOS | RemainingText1] ).
  apply(alter(FromPOS, ToPOS, prev1or2tag(POS)),
    [X, Y| RemainingText], [X, Y| RemainingText1] ) :-
    apply_aux(alter(FromPOS, ToPOS, prev1or2tag(POS)),
      [X, Y| RemainingText], [X, Y| RemainingText1] ).

  apply_aux(alter(FromPOS, ToPOS, prev1or2tag(POS)),
    [Prev2Word/POS, Prev1Word/POS1, Word/FromPOS |
     RemainingText],
    [Prev2Word/POS, Prev1Word/POS1, Word/ToPOS |
     RemainingText1] ) :-
    !,
     apply_aux(alter(FromPOS, ToPOS, prev1or2tag(POS)),
      [Prev1Word/POS1, Word/ToPOS | RemainingText],
      [Prev1Word/POS1, Word/ToPOS | RemainingText1] ).
  apply_aux(alter(FromPOS, ToPOS, prev1or2tag(POS)),
    [Prev2Word/POS2, Prev1Word/POS, Word/FromPOS |
    RemainingText], [Prev2Word/POS2, Prev1Word/POS,
    Word/ToPOS | RemainingText1] ) :-
    !,
    apply_aux(alter(FromPOS, ToPOS, prev1or2tag(POS)),
      [Prev1Word/POS, Word/ToPOS | RemainingText],
      [Prev1Word/POS, Word/ToPOS | RemainingText1] ).
  apply_aux(alter(FromPOS, ToPOS, prev1or2tag(POS)),
    [X, Y, Z | RemainingText],
    [X, Y, Z| RemainingText1] ) :-
    apply_aux(alter(FromPOS, ToPOS, prev1or2tag(POS)),
      [Y, Z| RemainingText], [Y, Z | RemainingText1] ).
  apply_aux(alter(FromPOS, ToPOS, prev1or2tag(POS)),
     [PrevWord/POS, Word/FromPOS],
       [PrevWord/POS, Word/ToPOS]).
  apply_aux(alter(_, _, prev1or2tag(_)), [X,Y], [X,Y]).
```

```
%The ordered contextual rules
alter(verb, noun, prevtag(art)).
alter(verb, noun, prev1or2tag(art)).
```

6.2.3 Deriving Rules Automatically

One of the most interesting features of Brill's rules is that they can be learned automatically from a hand-annotated corpus. This type of algorithm is called transformation-based learning (TBL). Let us denote *Corpus* this corpus and *AnnotationReference* its hand-annotation. In the context, the hand-annotation is often called the **Gold Standard**.

The TBL algorithm first assigns the most likely (frequent) tag to each word. It produces errors, and all rules templates are instantiated for each tagging error measured against *AnnotationReference.* The rule that yields the greatest error reduction is selected and applied to alter the *Corpus* tagging. This process is iterated as long as the annotation results are not close enough to *AnnotationReference*.

Table 6.4 shows the steps of the algorithm. *Corpus* annotated at iteration i of the process is denoted *AnnotatedCorpus(i).* Each iteration enables us to derive a new rule, which is denoted *Rule(i).*

Table 6.4. Brill's learning algorithm.

St.	Operation	Input	Output
1.	Annotate each word of the corpus with its most likely part of speech	*Corpus*	*AnnotatedCorpus(1)*
2.	Compare pairwise the part of speech of each word of the *AnnotationReference* and *AnnotatedCorpus(i)*	*AnnotationReference* *AnnotatedCorpus(i)*	List of errors
3.	For each error, instantiate the rule templates to correct the error	List of errors	List of tentative rules
4.	For each instantiated rule, compute on *AnnotatedCorpus(i)* the number of good transformations minus the number of bad transformations the rule yields	*AnnotatedCorpus(i)* Tentative rules	Scored tentative rules
5.	Select the rule that has the greatest error reduction and append it to the ordered list of transformations	Tentative rules	*Rule(i)*
6.	Apply *Rule(i)* to *AnnotatedCorpus(i)*	*AnnotatedCorpus(i)* *Rule(i)*	*AnnotatedCorpus(i+1)*
7.	If number of errors is under predefined threshold, end the algorithm else go to step 2.	–	List of rules

As hand-annotated corpus, Brill (1995) used the Penn Treebank (Marcus et al. 1993). Table 6.5 lists the five most productive rules that the algorithm learned from the *Wall Street Journal* annotated section of the corpus (Brill 1995).

Table 6.5. The five first transformations learned from the *Wall Street Journal* corpus (Brill 1995), where NN is a singular noun; VB is a verb, base form; TO is the word *to*; VBP is a verb, non-third person singular present; MD is a modal; DT is a determiner; VBD is a verb, past tense; and VBZ is a verb, third-person singular present. These tags are defined by the Penn Treebank, and Sect. 6.4.2 details the complete tagset.

	Change		
#	From	To	Condition
1	NN	VB	Previous tag is TO
2	VBP	VB	One of the previous three tags is MD
3	NN	VB	One of the previous two tags is MD
4	VB	NN	One of the previous two tags is DT
5	VBD	VBN	One of the previous three tags is VBZ

6.2.4 Confusion Matrices

At each iteration of TBL algorithm, we can derive a confusion matrix that shows for each tag how many times a word has been wrongly labeled. Table 6.6 shows an example of it (Franz 1996), which enables us to understand and track errors. Again, parts of speech use the Penn Treebank tagset described in Sect. 6.4.2. The diagonal shows the breakdown of the tags correctly assigned, for example, 99.4% for determiners (DT). The rest of the table shows the tags wrongly assigned, i.e. for determiners: 0.3% to prepositions (IN) and 0.3% to adverbs (RB). This table is only an excerpt, therefore the sum of rows and columns is not equal to 100.

6.3 Unknown Words

We have made the assumption of a finite vocabulary. This is never the case in practice. Many words will likely be absent from the dictionary: proper and common nouns, verbs, adjectives, or adverbs.

There is no standard technique to deal with the unknown words. The baseline is to tag unknown words as nouns since it is the most frequent part of speech. Another technique is to use suffixes. Brill (1995) proposes a combination of both to extend the transformation-based algorithm. The initial step tags unknown words as proper nouns for capitalized words and as common nouns for the rest. Then it applies transformations from a set of predefined templates: change the tag of an unknown word from X to Y if:

Table 6.6. A confusion matrix. The first column corresponds to the correct tags, and for each tag, the rows give the assigned tags. Excerpt from Franz (1996, p. 124). IN is a preposition, RB is an adverb, JJ is an adjective, RP is a particle, VBG is a verb, gerund (complete tagset in Sect. 6.4.2).

↓**Correct**	**Tagger** →									
	DT	IN	JJ	NN	RB	RP	VB	VBD	VBG	VBN
DT	99.4	0.3	–	–	0.3	–	–	–	–	–
IN	0.4	97.5	–	–	1.5	0.5	–	–	–	–
JJ	–	0.1	93.9	1.8	0.9	–	0.1	0.1	0.4	1.5
NN	–	–	2.2	95.5	–	–	0.2	–	0.4	–
RB	0.2	2.4	2.2	0.6	93.2	1.2	–	–	–	–
RP	–	24.7	–	1.1	12.6	61.5	–	–	–	–
VB	–	–	0.3	1.4	–	–	96.0	–	–	0.2
VBD	–	–	0.3	–	–	–	–	94.6	–	4.8
VBG	–	–	2.5	4.4	–	–	–	–	93.0	–
VBN	–	–	4.6	–	–	–	–	4.3	–	90.6

1. Deleting the prefix (suffix) x, $|x| \leq 4$, results in a word (x is any string of length 1 to 4).
2. The first (last) (1, 2, 3, 4) characters of the word are x.
3. Adding the character string x as a prefix (suffix) results in a word.
4. Word w ever appears immediately to the left (right) of the word.
5. Character z appears in the word.

These templates are specific to English, but they can easily be modified to accommodate other European languages. Table 6.7 shows the first five transformations learned from the *Wall Street Journal* corpus.

Table 6.7. The first five transformations for unknown words (Brill 1995), where NN is a noun, singular; NNS a noun, plural; CD cardinal number; JJ an adjective; VBN a verb, past participle; VBG a verb, gerund (complete tagset in Sect. 6.4.2).

	Change		
#	**From**	**To**	**Condition**
1	NN	NNS	Has suffix **s**
2	NN	CD	Has character **.**
3	NN	JJ	Has character **-**
4	NN	VBN	Has suffix **ed**
5	NN	VBG	Has suffix **ing**

6.4 Standardized Part-of-Speech Tagsets

While basic parts of speech are relatively well defined: determiners, nouns, pronouns, adjectives, verbs, auxiliaries, adverbs, conjunctions, and prepositions, there is a debate on how to standardize them for a computational analysis. One issue is the level of detail. Some tagsets feature a dozen tags, some over a hundred. Another issue that is linked to the latter and is that of subcategories. How many classes for verbs? Only one or should we create auxiliaries, modal, gerund, intransitive, transitive verbs, etc.?

The debate becomes even more complicated when we consider multiple languages. In French and German, the main parts of speech can be divided into subclasses depending on their gender, case, and number. In English, these divisions are useless. Although it is sometimes possible to map tagsets from one language to another, there is no universal scheme, even within the same language.

A few years ago, many computational linguists had a personal tagset. There are now standards, but the discussion is not over. We will examine here a multilingual part-of-speech scheme (MULTEXT), a widely accepted tagset for English (the Penn Treebank), and a tagset for Swedish.

6.4.1 Multilingual Part-of-Speech Tags

Building a multilingual tagset imposes the condition of having a set of common classes, which enables a comparison between languages. These classes correspond to traditional parts of speech and gather a relatively large consensus among European languages. However, they are not sufficiently accurate for any language in particular. Dermatas and Kokkinakis (1995) retained the traditional parts of speech to tag texts in seven European languages using statistical methods. They also added features (subcategories) specific to each language (Table 6.8).

Table 6.8. Parts of speech and grammatical features.

Main parts of speech	Features (subcategories)
Adjective, noun, pronoun	Regular base comparative superlative interrogative person number case
Adverb	Regular base comparative superlative interrogative
Article, determiner, preposition	Person case number
Verb	Tense voice mood person number case

MULTEXT (Ide and Véronis 1995) is a multinational initiative that aims at providing an annotation scheme for all the Western and Eastern European languages. MULTEXT also retains the traditional parts of speech (Table 6.9) that are common to all languages and complements them by a set of features, which they call attributes. Attributes enable us to subcategorize words and reconcile specific features of different European languages. Attributes for nouns and verbs are shown in Tables 6.10 and 6.11.

MULTEXT attributes concern only the morpho-syntactic layer and represent a superset of what is needed by all the languages. Some attributes may not be relevant for a specific language. For instance, English nouns have no gender, and French ones have no case. In addition, applications may not make use of some of the attributes even if they are part of the language. Tense, for instance, may be useless for some applications.

Table 6.9. MULTEXT's main parts of speech.

Part of speech	Code
Noun	N
Verb	V
Adjective	A
Pronoun	P
Determiner	D
Adverb	R
Adposition (Preposition)	S
Conjunction	C
Numeral	M
Interjection	I
Residual	X

Table 6.10. Features (attributes) and values for nouns.

Position	Attribute	Value	Code
1	Type	Common	c
		Proper	p
2	Gender	Masculine	m
		Feminine	f
		Neuter	n
3	Number	Singular	s
		Plural	p
4	Case	Nominative	n
		Genitive	g
		Dative	d
		Accusative	a

A part-of-speech tag is a string where the first character is the main class of the word to annotate and then a sequence of attribute values. Attribute positions correspond to their rank in the table, such as those defined in Tables 6.10 and 6.11 for nouns and verbs. When an attribute is not applicable, it is replaced by a dash (-). An English noun could receive the tag:

```
N[type=common number=singular] Nc-s-
```

Table 6.11. Attributes (features) and values for verbs.

Position	Attribute	Value	Code
1	Type	Main	m
		Auxiliary	a
		Modal	o
2	Mood/form	Indicative	i
		Subjunctive	s
		Imperative	m
		Conditional	c
		Infinitive	i
		Participle	p
		Gerund	g
		Supine	s
3	Tense	Base	b
		Present	p
		Imperfect	i
		Future	f
		Past	s
4	Person	First	1
		Second	2
		Third	3
5	Number	Singular	s
		Plural	p
6	Gender	Masculine	m
		Feminine	f
		Neuter	n

a French one:

```
N[type=common gender=masculine number=singular] Ncms-
```

and a German one:

```
N[type=common gender=neuter number=singular
    case=nominative] Ncnsn
```

A user can extend the coding scheme and add attributes if the application requires it. A noun could be tagged with some semantic features such as country names, currencies, etc.

6.4.2 Parts of Speech for English

The Penn Treebank is a large corpus of texts annotated with part-of-speech and syntactic tags (Marcus et al. 1993). The Penn Treebank part-of-speech tagset features 48 tags (Table 6.12).

Unlike MULTEXT, the Penn Treebank tagset concerns only English and shows little possibility of being adapted to another language. However, it is now widely

established in the North American language processing community and in industry. Lancaster University (UK) has defined another important tagset for English.

Table 6.12. The Penn Treebank tagset.

1.	CC	Coordinating conjunction	25.	TO	*to*
2.	CD	Cardinal number	26.	UH	Interjection
3.	DT	Determiner	27.	VB	Verb, base form
4.	EX	Existential *there*	28.	VBD	Verb, past tense
5.	FW	Foreign word	29.	VBG	Verb, gerund/present participle
6.	IN	Preposition/sub. conjunction	30.	VBN	Verb, past participle
7.	JJ	Adjective	31.	VBP	Verb, non-third pers. sing. pres.
8	JJR	Adjective, comparative	32.	VBZ	Verb, third-pers. sing. present
9.	JJS	Adjective, superlative	33.	WDT	*wh*-determiner
10.	LS	List item marker	34.	WP	*wh*-pronoun
11.	MD	Modal	35.	WP$	Possessive *wh*-pronoun
12.	NN	Noun, singular or mass	36.	WRB	*wh*-adverb
13.	NNS	Noun, plural	37.	#	Pound sign
14.	NNP	Proper noun, singular	38.	$	Dollar sign
15.	NNPS	Proper noun, plural	39.	.	Sentence final punctuation
16.	PDT	Predeterminer	40.	,	Comma
17.	POS	Possessive ending	41.	:	Colon, semicolon
18.	PRP	Personal pronoun	42.	(	Left bracket character
19.	PP$	Possessive pronoun	43.	)	Right bracket character
20.	RB	Adverb	44.	"	Straight double quote
21.	RBR	Adverb, comparative	45.	‘	Left open single quote
22.	RBS	Adverb, superlative	46.	“	Left open double quote
23.	RP	Particle	47.	’	Right close single quote
24.	SYM	Symbol	48.	”	Right close double quote

Figure 6.1 shows an annotated excerpt from the Penn Treebank. The Penn Treebank team proceeded in two steps to annotate their corpus. They first tagged the texts with an automatic stochastic tagger. They then reviewed and manually corrected the annotation.

Battle-tested/JJ industrial/JJ managers/NNS here/RB always/RB buck/VBP up/RP nervous/JJ newcomers/NNS with/IN the/DT tale/ NN of/IN the/DT first/JJ of/IN their/PP$ countrymen/NNS to/TO visit/VB Mexico/NNP ,/, a/DT boatload/NN of/IN samurai/FW warriors/NNS blown/VBN ashore/RB 375/CD years/NNS ago/RB ./.
“/“ From/IN the/DT beginning/NN ,/, it/PRP took/VBD a/DT man/NN with/IN extraordinary/JJ qualities/NNS to/TO succeed/VB in/IN Mexico/NNP ”/” says/VBZ Kimihide/NNP Takimura/NNP ,/, president/NN of/IN the/DT Mitsui/NNP group/NN ’s/POS Kensetsu/NNP Engineering/NNP Inc./NNP unit/NN ./.

Fig. 6.1. Sample of annotated text from the Penn Treebank. After Marcus et al. (1993).

6.4.3 An Annotation Scheme for Swedish

Current annotation schemes often use XML to encode data. This enables a stricter definition of codes through a DTD and makes it easier to use and share data. The annotation is often split into levels that reflect the processing stages. We describe here an example drawn from the Granska and CrossCheck projects to process Swedish (Carlberger et al. 2006) from the Kungliga Tekniska Högskolan in Stockholm. The annotation scheme uses the reference tagset for Swedish defined by the Stockholm-Umeå Corpus (Ejerhed et al. 1992).

The annotation has four levels, and we will describe two of them. The first one corresponds to tokenization. Figure 6.2 shows the token annotation of sentence:

> *Bilen framför justitieministern svängde fram och tillbaka över vägen så att hon blev rädd.*
> 'The car in front of the Justice Minister swung back and forth and she was frightened.'

```
<tokens>
  <token id="1">Bilen</token>
  <token id="2">framför</token>
  <token id="3">justitieministern</token>
  <token id="4">svängde</token>
  <token id="5">fram</token>
  <token id="6">och</token>
  <token id="7">tillbaka</token>
  <token id="8">över</token>
  <token id="9">vägen</token>
  <token id="10">så</token>
  <token id="11">att</token>
  <token id="12">hon</token>
  <token id="13">blev</token>
  <token id="14">rädd</token>
  <token id="15">.</token>
</tokens>
```

Fig. 6.2. Token annotation, where the identifier `id` corresponds to the word position.

The second level contains the part-of-speech information, either with lemmas (Fig. 6.3) or without (Fig. 6.4). In both annotations, the tokens have been replaced by their positions. The `tag` attribute gives the part of speech and its features as a list separated by dots. The first item of the list the main category; for example, `nn` is a noun. The rest describes the features: `utr` is the utrum gender, `sin` is the singular number, `def` means definite, and `nom` is the nominative case.

```
<taglemmas>
  <taglemma id="1" tag="nn.utr.sin.def.nom" lemma="bil"/>
  <taglemma id="2" tag="pp" lemma="framför"/>
  <taglemma id="3" tag="nn.utr.sin.def.nom"
lemma="justitieminister"/>
  <taglemma id="4" tag="vb.prt.akt" lemma="svänga"/>
  <taglemma id="5" tag="ab" lemma="fram"/>
  <taglemma id="6" tag="kn" lemma="och"/>
  <taglemma id="7" tag="ab" lemma="tillbaka"/>
  <taglemma id="8" tag="pp" lemma="över"/>
  <taglemma id="9" tag="nn.utr.sin.def.nom" lemma="väg"/>
  <taglemma id="10" tag="ab" lemma="så"/>
  <taglemma id="11" tag="sn" lemma="att"/>
  <taglemma id="12" tag="pn.utr.sin.def.sub" lemma="hon"/>
  <taglemma id="13" tag="vb.prt.akt.kop" lemma="bli"/>
  <taglemma id="14" tag="jj.pos.utr.sin.ind.nom"
lemma="rädd"/>
  <taglemma id="15" tag="mad" lemma="."/>
</taglemmas>
```

Fig. 6.3. Tokens annotated with their part of speech and lemma. Tokens are indicated by their position. The tag specifies the part of speech and its features.

```
<tags>
  <tag id="1" name="nn.utr.sin.def.nom"/>
  <tag id="2" name="pp"/>
  <tag id="3" name="nn.utr.sin.def.nom"/>
  <tag id="4" name="vb.prt.akt"/>
  <tag id="5" name="ab"/>
  <tag id="6" name="kn"/>
  <tag id="7" name="ab"/>
  <tag id="8" name="pp"/>
  <tag id="9" name="nn.utr.sin.def.nom"/>
  <tag id="10" name="ab"/>
  <tag id="11" name="sn"/>
  <tag id="12" name="pn.utr.sin.def.sub"/>
  <tag id="13" name="vb.prt.akt.kop"/>
  <tag id="14" name="jj.pos.utr.sin.ind.nom"/>
  <tag id="15" name="mad"/>
</tags>
```

Fig. 6.4. Tokens annotated with their part of speech only. Tokens are indicated by their position.

6.5 Further Reading

Part-of-speech tagging has a long history in language processing, although many researchers in computational linguistics neglected it in the beginning. Early works include Harris (1962) and Klein and Simmons (1963). Harris' TDAP system was reconstructed and described by Joshi and Hopely (1999).

Brill's tagging program marked a breakthrough in tagging with symbolic techniques. It is available from the Internet for English. Roche and Schabes (1995) proposed a dramatic optimization of it that proved ten times faster than and one third the size of stochastic methods. Constant (1991) and Vergne (1998, 1999) give examples of efficient symbolic taggers that use manually crafted rules.

Exercises

6.1. Complement Brill's tagging algorithm in Prolog with rules `alter(A, B, nexttag(C))` and `alter(A, B, surroundingtag(C, D))`.

6.2. Implement Brill's learning algorithm in Prolog or Perl with all the rule templates.