Part-of-Speech Tagging M. Covington 2008, 2011

In the 1990s it was discovered that **you can label the parts of speech** (nouns, verbs, adjectives, etc.) in a text with something like 95% accuracy *without* doing a full syntactic parse.

This is useful for many reasons. In the CASPR project, we are using part-of-speech tagging (POS tagging) to determine idea density. More commonly, tagging precedes parsing or information extraction.

Tagging is more than just looking up each word in a dictionary because many words can be more than one POS. (*Can* is a noun, a modal verb, and a regular verb: *Can they can corn in aluminum cans?*)

Assuming you have a dictionary, if you just give each word its most common POS, you get about 90% accuracy.

To get considerably higher accuracy, you have to look at context. (In the phrase *the can*, you can tell that *can* is a noun without parsing any further structure.)

You cannot get accuracy higher than about 97%, probably because that is the level of the internal consistency of the Penn Treebank (it may be about 3% errors) and partly because Penn Treebank tagging requires some *very* subtle distinctions, such as VB versus VBP (which are always identical in form – see below).

There are several ways of looking at context. Two main ones:

Hidden Markov models – Probability based on what came earlier in the sequence. ("After a determiner and then an adjective, the probability of a noun is...")

> Called *hidden* because the states are sets of probabilities, not the words themselves.
>
> The *Viterbi algorithm* is an efficient way of finding the highest overall probability when a lot of different possibilities occur in immediate succession.

Brill (transformation-based) tagging – Rules to correct the initial guesses. That is, first you label each word with its most common POS, and then you make changes such as, "If a modal verb could also be a noun, change it to a noun if the preceding word is a determiner." See Eric Brill, paper in *Computational Linguistics,* 1995, which is unusually well-written.

> The rules for making changes can be learned by a simple machine-learning process. That's the appeal of Brill tagging.

Tagged corpora: The Penn Treebank and many other corpora have been tagged (by hand) and are very useful for training and testing taggers.

Morphology? In English it is customary to tag words before performing any morphological analysis. (E.g., *baby* and *babies* are in the lexicon separately.) In other languages you wouldn't be able to do that – there's a lot more inflectional morphology.

The Penn Treebank tag set

(from http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/CQP-HTMLDemo/PennTreebankTS.html):

1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

There are a few more symbols for punctuation marks.

Note that TO is in a category by itself, so that the tagger doesn't have to decide whether it's a preposition or a marker introducing a verb.

Note also (as you will see in the next example) that words like *after* and *before* are considered prepositions even when they introduce sentences rather than noun phrases.

Example of some tagged text from the Penn Treebank:

```
[ The/DT school/NN ]
finds/VBZ that/IN
[ the/DT children/NNS ]
are/VBP satisfied/VBN with/IN
[ smaller/JJR amounts/NNS ]
of/IN
[ food/NN ]
since/IN
[ all/DT ]
of/IN
[ it/PRP ]
is/VBZ high/JJ in/IN
[ quality/NN ]
./.
```

Here square brackets mark "noun groups." Taggers do not normally identify noun groups, but a very shallow parser can do so.

About tagging of verb forms

Tagging of verb forms makes some distinctions that are syntactic, not morphological. That means it is very error-prone, but for many purposes, these errors do not matter. Here's the verb system:

VB - the plain form of the verb (no suffix): *he will sing*.
VBP - present-tense verb with no suffix: *they sing*.
(VB and VBP are always identical in form, except *are*/VBP = *be*/VB.)

VBD - the past tense form: *he sang*, *he talked*
VBN - the past participle: *he has sung*, *he has talked*
(VBD and VBN are identical in form if the verb is regular.)

VBZ - verb with an –*s* suffix: *he sings*

VBG - verb with an –*ing* suffix: *he is singing*

A cookbook approach to Brill tagging

To build a Brill tagger, first scan your pre-tagged text (training corpus) and make a **lexicon** (dictionary) that tells you:

- The most common POS tag for each word.
- All the possible POS tags for each word.

(For easy updating, we actually keep counts of how many times each tag has occurred in the training corpora.)

The transformation rules will never change a tag unless the result is possible. For example, *can* might get changed from modal verb to noun, but that will not happen to *shall* in the same context because *shall* is never a noun.

Next, learn the transformation rules automatically as follows.

In its initial state, the tagger simply assigns the most common tag to each word. Looking at the tagged corpus, find a place where the tagger gets a word wrong (imagine the tag is X and should be Y). Look at the 2 preceding and following tags (call them A, B, C, and D) and construct a set of candidate rules of the form:

Change X to Y if the preceding tag is B
Change X to Y if the preceding two tags are A and B
Change X to Y if the following tag is C
Change X to Y if the following two tags are C and D

and so on. Brill provides a total of 11 rule schemas – so in any context you can hypothesize 11 different rules.

Now try using each of your 11 rules on the whole corpus and see how much it would improve (or worsen!) the tagging. Choose the rule that works best, and add it to the rule set. Also, apply it throughout and update the partially tagged text you have in memory.

Then proceed, and find *another* tag that's wrong, and do the same thing...

Keep doing this until the improvements are small (or nonexistent).

At this point you will probably be needing some rules that need to refer to specific words, not just tags. Brill therefore introduces 19 rule schemas that can do this; you can train on them after training on the original schemas has stopped paying off. He reports that doing this added very little to the accuracy, probably because the corpus was not big enough; when you're going to learn rules for individual words, you need a lot of examples of each one.

I would also advocate looking at the residual errors and making up rules by hand – we may notice something that the computer couldn't notice.

Brill's tagger is published and available as a free download. So yet another possibility is to use his rule set (already learned) together with a freshly created lexicon (from a corpus), plus some hand-tweaking.