Part-of-speech tagging (1)

Steve Renals
s.renals@ed.ac.uk

ICL — 13 October 2005
Parts of Speech
  Introduction
  Open and closed classes
  Tagsets

PoS Tagging in NLTK
  Tagging
  Simple taggers
  Unigram taggers

Evaluating taggers
  Accuracy and gold standard
  Error analysis

Summary
Parts of speech

- How can we predict the behaviour of a previously unseen word?
- Words can be divided into classes that behave similarly.
- Traditionally eight parts of speech: noun, verb, pronoun, preposition, adverb, conjunction, adjective and article.
- More recently larger sets have been used: eg Penn Treebank (45 tags), Susanne (353 tags).
Parts of Speech

What use are parts of speech?
They tell us a lot about a word (and the words near it).
Parts of Speech

What use are parts of speech?
They tell us a lot about a word (and the words near it).

▶ Tell us what words are likely to occur in the neighbourhood (eg adjectives often followed by nouns, personal pronouns often followed by verbs, possessive pronouns by nouns)

▶ Pronunciations can be dependent on part of speech, eg object, content, discount (useful for speech synthesis and speech recognition)

▶ Can help information retrieval and extraction (stemming, partial parsing)

▶ Useful component in many NLP systems
Closed and open classes

- Parts of speech may be categorised as *open* or *closed* classes
- Closed classes have a fixed membership of words (more or less), eg determiners, pronouns, prepositions
- Closed class words are usually *function words* — frequently occurring, grammatically important, often short (eg of, it, the, in)
- The major open classes are *nouns*, *verbs*, *adjectives* and *adverbs*
Closed classes in English

prepositions  on, under, over, to, with, by

determiners  the, a, an, some

pronouns   she, you, I, who

conjunctions  and, but, or, as, when, if

auxiliary verbs  can, may, are

particles   up, down, at, by

numerals  one, two, first, second
Open classes

nouns  Proper nouns (Scotland, BBC),
        common nouns:
        ▶ count nouns (goat, glass)
        ▶ mass nouns (snow, pacifism)
verbs  actions and processes (run, hope), also auxiliary verbs
adjectives properties and qualities (age, colour, value)
adverbs  modify verbs, or verb phrases, or other adverbs:
         Unfortunately John walked home extremely slowly yesterday
The Penn Treebank tagset (1)

<table>
<thead>
<tr>
<th>CC</th>
<th>Coord Conjunct</th>
<th>and, but, or</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>Cardinal number</td>
<td>one, two</td>
</tr>
<tr>
<td>DT</td>
<td>Determiner</td>
<td>the, some</td>
</tr>
<tr>
<td>EX</td>
<td>Existential there</td>
<td>there</td>
</tr>
<tr>
<td>FW</td>
<td>Foreign Word</td>
<td>mon dieu</td>
</tr>
<tr>
<td>IN</td>
<td>Preposition</td>
<td>of, in, by</td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
<td>big</td>
</tr>
<tr>
<td>JJR</td>
<td>Adj., comparative</td>
<td>bigger</td>
</tr>
<tr>
<td>JJS</td>
<td>Adj., superlative</td>
<td>biggest</td>
</tr>
<tr>
<td>LS</td>
<td>List item marker</td>
<td>1, One</td>
</tr>
<tr>
<td>MD</td>
<td>Modal</td>
<td>can, should</td>
</tr>
<tr>
<td>NN</td>
<td>Noun, sing. or mass</td>
<td>dog</td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
<td>dogs</td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, sing.</td>
<td>Edinburgh</td>
</tr>
<tr>
<td>NNPS</td>
<td>Proper noun, plural</td>
<td>Orkneys</td>
</tr>
<tr>
<td>PDT</td>
<td>Predeterminer</td>
<td>all, both</td>
</tr>
<tr>
<td>POS</td>
<td>Possessive ending</td>
<td>’s</td>
</tr>
<tr>
<td>PP</td>
<td>Personal pronoun</td>
<td>I, you, she</td>
</tr>
<tr>
<td>PP$</td>
<td>Possessive pronoun</td>
<td>my, one’s</td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
<td>quickly</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>
</tbody>
</table>
### The Penn Treebank tagset (2)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP</td>
<td>Particle</td>
<td>up, off</td>
</tr>
<tr>
<td>SYM</td>
<td>Symbol</td>
<td>+, %, &amp;</td>
</tr>
<tr>
<td>TO</td>
<td>“to”</td>
<td>to</td>
</tr>
<tr>
<td>UH</td>
<td>Interjection</td>
<td>oh, oops</td>
</tr>
<tr>
<td>VB</td>
<td>verb, base form</td>
<td>eat</td>
</tr>
<tr>
<td>VBD</td>
<td>verb, past tense</td>
<td>ate</td>
</tr>
<tr>
<td>VBG</td>
<td>verb, gerund</td>
<td>eating</td>
</tr>
<tr>
<td>VBN</td>
<td>verb, past part</td>
<td>eaten</td>
</tr>
<tr>
<td>VBP</td>
<td>Verb, non-3sg, pres</td>
<td>eat</td>
</tr>
<tr>
<td>VBZ</td>
<td>Verb, 3sg, pres</td>
<td>eats</td>
</tr>
<tr>
<td>WDT</td>
<td>Wh-determiner</td>
<td>which, that</td>
</tr>
<tr>
<td>WP</td>
<td>Wh-pronoun</td>
<td>what, who</td>
</tr>
<tr>
<td>WP$</td>
<td>Possessive-Wh</td>
<td>whose</td>
</tr>
<tr>
<td>WRB</td>
<td>Wh-adverb</td>
<td>how, where</td>
</tr>
<tr>
<td>$</td>
<td>Dollar sign</td>
<td>$</td>
</tr>
<tr>
<td>#</td>
<td>Pound sign</td>
<td>#</td>
</tr>
<tr>
<td>“</td>
<td>Left quote</td>
<td>‘ , “</td>
</tr>
<tr>
<td>”</td>
<td>Right quote</td>
<td>’ , ”</td>
</tr>
<tr>
<td>(</td>
<td>Left paren</td>
<td>(</td>
</tr>
<tr>
<td>)</td>
<td>Right paren</td>
<td>)</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
<td>,</td>
</tr>
<tr>
<td>.</td>
<td>Sent-final punct</td>
<td>. ! ?</td>
</tr>
<tr>
<td>:</td>
<td>Mid-sent punct.</td>
<td>: ; — ...</td>
</tr>
</tbody>
</table>
Tagging

Definition: Tagging is the assignment of a single part-of-speech tag to each word (and punctuation marker) in a corpus. For example:

"/" The/DT guys/NNS that/WDT make/VBP traditional/JJ hardware/NN are/VBP really/RB being/VBG obsoleted/VBN by/IN microprocessor-based/JJ machines/NNS ,/, "/" said/VBD Mr./NNP Benton/NNP ./.

Non-trivial: POS tagging must resolve ambiguities since the same word can have different tags in different contexts

In the Brown corpus 11.5% of word types and 40% of word tokens are ambiguous

In many cases one tag is much more likely for a given word than any other

Limited scope: only supplying a tag for each word, no larger structures created (eg prepositional phrase attachment)
Information sources for tagging

What information can help decide the correct PoS tag for a word?

Other PoS tags Even though the PoS tags of other words may be uncertain too, we can use information that some tag sequences are more likely than others (eg the/AT red/JJ drink/NN vs the/AT red/JJ drink/VBP). Using only information about the most likely PoS tag sequence does not result in an accurate tagger (about 77% correct)

The word identity Many words can gave multiple possible tags, but some are more likely than others (eg fall/VBP vs fall/NN)

Tagging each word with its most common tag results in a tagger with about 90% accuracy
Tagging in NLTK

The simplest possible tagger tags everything as a noun:

```python
from nltk_lite import tokenize

text = 'There are 11 players in a football team'
text_tokens = list(tokenize.whitespace(text))

for t in mytagger.tag(text_tokens):
    print t
```

# ('There', 'NN')
# ('are', 'NN')
# ...

Steve Renals s.renals@ed.ac.uk

Part-of-speech tagging (1)
Tagging in NLTK

The simplest possible tagger tags everything as a noun:

```python
from nltk_lite import tokenize
text = 'There are 11 players in a football team'
text_tokens = list(tokenize.whitespace(text))
# ['There', 'are', '11', 'players', 'in', 'a', 'football', 'team']

from nltk_lite import tag
mytagger = tag.Default('nn')
for t in mytagger.tag(text_tokens):
    print t
# ('There', 'NN')
# ('are', 'NN')
# ...
```
A regular expression tagger

We can use regular expressions to tag tokens based on regularities in the text, eg numerals:

default_pattern = (r'.*', 'NN')
cd_pattern = (r' ^[0-9]+(.[0-9]+)?$', 'CD')
patterns = [cd_pattern, default_pattern]
NN_CD_tagger = tag.Regexp(patterns)
re_tagged = list(NN_CD_tagger.tag(text_tokens))
# [('There', 'NN'), ('are', 'NN'), ('11', 'NN'), ('players', 'NN'), ('in', 'NN'), ('a', 'NN'), ('football', 'NN'), ('team', 'NN')]
A unigram tagger

The NLTK UnigramTagger class implements a tagging algorithm based on a table of unigram probabilities:

\[ \text{tag}(w) = \arg \max_{t_i} P(t_i|w) \]
A unigram tagger

The NLTK UnigramTagger class implements a tagging algorithm based on a table of unigram probabilities:

\[
tag(w) = \arg \max_{t_i} P(t_i|w)\]

Training a UnigramTagger on the Penn Treebank:

```python
from nltk_lite.corpora import treebank
from itertools import islice

# sentences 0-2999
train_sents = list(islice(treebank.tagged(), 3000))
# from sentence 3000 to the end
test_sents = list(islice(treebank.tagged(), 3000, None))

unigram_tagger = tag.Unigram()
unigram_tagger.train(train_sents)
```

Steve Renals s.renals@ed.ac.uk
Part-of-speech tagging (1)
Unigram tagging

```python
>>> list(unigram_tagger.tag(tokenize.whitespace("Mr. Jones saw the book on the shelf")))
[('Mr.', 'NNP'), ('Jones', 'NNP'), ('saw', 'VBD'), ('the', 'DT'), ('book', 'NN'), ('on', 'IN'), ('the', 'DT'), ('shelf', None)]
```

The UnigramTagger assigns the default tag `None` to words that are not in the training data (eg `shelf`).
Unigram tagging

```python
>>> list(unigram_tagger.tag(tokenize.whitespace("Mr. Jones saw the book on the shelf")))
[('Mr.', 'NNP'), ('Jones', 'NNP'), ('saw', 'VBD'), ('the', 'DT'), ('book', 'NN'), ('on', 'IN'), ('the', 'DT'), ('shelf', None)]
```

The UnigramTagger assigns the default tag None to words that are not in the training data (eg shelf)
We can combine taggers to ensure every word is tagged:

```python
>>> unigram_tagger = tag.Unigram(cutoff=0, backoff=NN_CD_tagger)
>>> unigram_tagger.train(train_sents)
>>> list(unigram_tagger.tag(tokenize.whitespace("Mr. Jones saw the book on the shelf")))
[('Mr.', 'NNP'), ('Jones', 'NNP'), ('saw', 'VBD'), ('the', 'DT'), ('book', 'VB'), ('on', 'IN'), ('the', 'DT'), ('shelf', 'NN')]
```
Evaluating taggers

▶ Basic idea: compare the output of a tagger with a human-labelled \textit{gold standard}
▶ Need to compare how well an automatic method does with the agreement between people
▶ The best automatic methods have an accuracy of about 96-97\% when using the (small) Penn treebank tagset (but this is still an average of one error every couple of sentences...)
▶ Inter-annotator agreement is also only about 97\%
▶ A good unigram baseline (with smoothing) can obtain 90-91\%
Evaluating taggers in NLTK

NLTK provides a function `tag.accuracy` to automate evaluation. It needs to be provided with a tagger, together with some text to be tagged and the gold standard tags.
Evaluating taggers in NLTK

NLTK provides a function `tag.accuracy` to automate evaluation. It needs to be provided with a tagger, together with some text to be tagged and the gold standard tags. We can make print more prettily:

```python
def print_accuracy(tagger, data):
    print '%3.1f%%' % (100 * tag.accuracy(tagger, data))
```
Evaluating taggers in NLTK

NLTK provides a function `tag.accuracy` to automate evaluation. It needs to be provided with a tagger, together with some text to be tagged and the gold standard tags. We can make print more prettily:

```python
def print_accuracy(tagger, data):
    print '%3.1f%%' % (100 * tag.accuracy(tagger, data))
```

```python
>>> print_accuracy(NN_CD_tagger, test_sents)
15.0%
>>> print_accuracy(unigram_tagger, train_sents)
93.8%
>>> print_accuracy(unigram_tagger, test_sents)
82.8%
```
Error analysis

► The % correct score doesn’t tell you everything — it is useful to know what is misclassified as what

► **Confusion matrix**: A matrix (ntags x ntags) where the rows correspond to the correct tags and the columns correspond to the tagger output. Cell \((i, j)\) gives the count of the number of times tag \(i\) was classified as tag \(j\)

► The leading diagonal elements correspond to correct classifications

► Off diagonal elements correspond to misclassifications

► Thus a confusion matrix gives information on the major problems facing a tagger (e.g., NNP vs. NN vs. JJ)

► See section 3 of the NLTK tutorial on Tagging
Summary

- **Reading**: Chapter 8 of Jurafsky and Martin
- Parts of speech and tagsets
- Tagging
- Constructing simple taggers in NLTK
- Evaluating taggers
- Next lecture: n-grams