Part-of-speech tagging (1)

Steve Renals
s.renals@ed.ac.uk

ICL — 19 October 2006
Parts of Speech
   Introduction
   Open and closed classes
   Tagsets

PoS Tagging in NLTK
   Tagging
   Simple taggers
   Unigram taggers

Rule-based tagging

Evaluating taggers
   Accuracy and gold standard
   Error analysis

Summary
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), e.g., determiners, pronouns, prepositions.
- Closed class words are usually *function words* — frequently occurring, grammatically important, often short (e.g., 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)

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Example</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 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))
# ['There', 'are', '11', 'players', 'in', 'a', 'football', 'team']
```

```python
from nltk_lite import tag
mytagger = tag.Default('NN')
for t in mytagger.tag(text_tokens):
    print t
# ('There', 'NN')
# ('are', 'NN')
# ...
```
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')
# ...
```

Steve Renals s.renals@ed.ac.uk
A regular expression tagger

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

default_pattern = (r'.*$', 'NN')
cd_pattern = (r'^[0-9]+\.(?:[0-9]+)?$\s$', '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')]
Unigram tagger trained on Penn Treebank

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

$$\text{tag}(w) = \arg \max_{t_i} P(t_i|w)$$
Unigram tagger trained on Penn Treebank

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

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

from nltk_lite import tokenize, tag
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)
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

>>> 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:

>>> 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')]

Steve Renals s.renals@ed.ac.uk
Rule-based tagging using constraints

- Lexicon based, listing morphological and syntactic features for each word: includes inflected and derived forms, with a separate entry for each PoS:
  - show/V: PRESENT -SG3 VFIN
  - show/N: NOMINATIVE SG
Rule-based tagging using constraints

- Lexicon based, listing morphological and syntactic features for each word: includes inflected and derived forms, with a separate entry for each PoS:
  - show/V: PRESENT -SG3 VFIN
  - show/N: NOMINATIVE SG

- Multi-stage tagging:
  1. Return all possible POS tags (and associated features) for each word
  2. Apply constraints (rules) to remove parts-of-speech inconsistent with the context

- More details in Jurafsky and Martin (1st ed. sec 8.4; 2nd ed. sec 5.4)
Transformation-based tagging

- A rule-based system...

- But the rules are learned from a corpus

- Basic approach: start by applying general rules, then successively refine with additional rules that correct the mistakes

- Learn the rules from a corpus, using a set of rule templates, e.g.:

  - Change tag a to b when the following word is tagged z

- Choose the best rule each iteration

- (see module nltk_lite.tag.brill, also sec 5.5/8.5 in J&M)
Transformation-based tagging

- A rule-based system...
- ...but the rules are learned from a corpus
Transformation-based tagging

- A rule-based system...
- ...but the rules are learned from a corpus
- Basic approach: start by applying general rules, then successively refine with additional rules that correct the mistakes
- Learn the rules from a corpus, using a set of rule templates, eg: Change tag a to b when the following word is tagged z
- Choose the best rule each iteration
- (see module nltk_lite.tag.brill), also sec 5.5/8.5 in J&M

Steve Renals s.renals@ed.ac.uk
Evaluating taggers

- Basic idea: compare the output of a tagger with a human-labelled *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.

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

>>> print_accuracy(NN_CD_tagger, test_sents)
18.2%

>>> print_accuracy(unigram_tagger, train_sents)
93.7%

>>> print_accuracy(unigram_tagger, test_sents)
84.0%
```
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))
```

```plaintext
>>> print_accuracy(NN_CD_tagger, test_sents)
18.2%

>>> print_accuracy(unigram_tagger, train_sents)
93.7%

>>> print_accuracy(unigram_tagger, test_sents)
84.0%
```
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))
```

```bash
>>> print_accuracy(NN_CD_tagger, test_sents)
18.2%
>>> print_accuracy(unigram_tagger, train_sents)
93.7%
>>> print_accuracy(unigram_tagger, test_sents)
84.0%
```
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 4.4 of the NLTK tutorial on Tagging.
Summary

- **Reading**: Jurafsky and Martin (1st ed: chapter 8; 2nd ed: chapter 5); NLTK tagging tutorial
- Parts of speech and tagsets
- Tagging
- Constructing simple taggers in NLTK
- Rule-based tagging
- Evaluating taggers
- Next two lectures: statistical tagging using HMMs/n-grams