# NLP Applications
# Lecture 5: Tagging with NLTK

Claire Gardent

CNRS/LORIA
Campus Scientifique,
BP 239,
F-54 506 Vandœuvre-lès-Nancy, France

2007/2008







## Parts of speech

- 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).

- 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)

| | | | | | |
|---|---|---|---|---|---|
| CC | Coord Conjuncn | *and,but,or* | NN | Noun, sing. or mass | *dog* |
| CD | Cardinal number | *one,two* | NNS | Noun, plural | *dogs* |
| DT | Determiner | *the,some* | NNP | Proper noun, sing. | *Edinburgh* |
| EX | Existential there | *there* | NNPS | Proper noun, plural | *Orkneys* |
| FW | Foreign Word | *mon dieu* | PDT | Predeterminer | *all, both* |
| IN | Preposition | *of,in,by* | POS | Possessive ending | *'s* |
| JJ | Adjective | *big* | PP | Personal pronoun | *I,you,she* |
| JJR | Adj., comparative | *bigger* | PP$ | Possessive pronoun | *my,one's* |
| JJS | Adj., superlative | *biggest* | RB | Adverb | *quickly* |
| LS | List item marker | *1,One* | RBR | Adverb, comparative | *faster* |
| MD | Modal | *can,should* | RBS | Adverb, superlative | *fastest* |

## The Penn Treebank tagset (2)

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



## The Brown tagset

| | | |
|---|---|---|
| ap | determiner-pronoun, post-determiner | many other next |
| at | article | the an no a every |
| cc | conjunction, coordinating | and or but plus |
| cs | conjunction, subordinating | that as after whether |
| in | preposition | in for by |
| md | modal auxiliary | may might will |
| pn | pronoun, nominal | something, none |
| ppl | pronoun, singular, reflexive | himself myself |
| pp$ | determiner, possessive | its his their my |
| pp$$ | pronoun, possessive | mine his hers theirs yours |
| pps | pronoun, personal, nom, 3rd pers sng | he she |
| ppss | pronoun, personal, nom, not 3rd pers sng | we I you |
| wdt | WH-determiner | what whatever |
| wps | WH-pronoun, nominative | who whoever |



## 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 ./.



## Tagging

- 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 have 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

NLTK provides several means of developing a tagger:

- Assigning a default tag
- Using regular expressions on strings (the shape of the string helps guessing the associated PoS tag
- Unigram tagging: assigning the most probable tag
- Bigram tagging: assigning the most probable tag given a left-adjacent PoS
- Brill tagging: transformation based learning of tags



## Tagging in NLTK

The simplest possible tagger tags everything as a noun:

```
>>> tokens = 'John saw 3 polar bears .'.split()
>>> default_tagger = nltk.DefaultTagger('NN')
>>> default_tagger.tag(tokens)
[('John', 'NN'), ('saw', 'NN'), ('3', 'NN'), ('polar', 'NN
('bears', 'NN'), ('.', 'NN')]
```



## A regular expression tagger

The regular expression tagger assigns tags to tokens on the basis of matching patterns in the token's text. For instance, the following tagger assigns "cd" to cardinal numbers, and "nn" to everything else:

```
>>> default_pattern = (r'.*', 'NN')
>>> cd_pattern = (r' ^[0-9]+(.[0-9]+)?$', 'CD')
>>> patterns = [cd_pattern, default_pattern]
>>> tokens = 'There are 11 players in a football team'.split()
['There', 'are', '11', 'players', 'in', 'a', 'football', 'team']
>>> NN_CD_tagger = nltk.RegexpTagger(patterns)
>>> NN_CD_tagger.tag(tokens)
 [('There', 'NN'), ('are', 'NN'), ('11', 'CD'), ('players', 'NN'
('in', 'NN'), ('a', 'NN'), ('football', 'NN'), ('team', 'NN')]
```

## A more sophisticated regular expression tagger

Guess the correct tag for words based on the presence of certain prefix or suffix strings.

```
 >>> patterns = [
...     (r'.*ing$', 'VBG'),               # gerunds
...     (r'.*ed$', 'VBD'),                # simple past
...     (r'.*es$', 'VBZ'),                # 3rd singular present
...     (r'.*ould$', 'MD'),               # modals
...     (r'.*\'s$', 'NN$'),               # possessive nouns
...     (r'.*s$', 'NNS'),                 # plural nouns
...     (r'^-?[0-9]+(.[0-9]+)?$', 'CD'),  # cardinal numbers
...     (r'.*', 'NN')                     # nouns (default)
... ]
>>> regexp_tagger = nltk.RegexpTagger(patterns)
>>> regexp_tagger.tag(nltk.corpus.brown.sents('a')[3])
[('``', 'NN'), ('Only', 'NN'), ('a', 'NN'), ('relative', 'NN'),
('handful', 'NN'), ('of', 'NN'), ('such', 'NN'), ('reports', 'NN
('was', 'NNS'), ('received', 'VBD'), ("''", 'NN'), (',', 'NN'),
('the', 'NN'), ('jury', 'NN'), ('said', 'NN'), (',', 'NN'), ('``
('considering', 'VBG'), ('the', 'NN'), ('widespread', 'NN'), ...
```



## Unigram taggers

- A UnigramTagger assigns to each token its most likely tag
- A UnigramTagger is based on a table of unigram probabilities
- This table is derived from an annotated corpus
- Before a "UnigramTagger" can be used to tag data, it must be trained on a *training corpus*. This corpus is used to determine which tags are most common for each word.
- Annotated data can also be used to evaluate the tagger



## A unigram tagger

The NLTK UnigramTagger class implements a unigram tagging algorithm

```
>>> brown_a = nltk.corpus.brown.tagged_sents('a')
>>> unigram_tagger = nltk.UnigramTagger(brown_a)
>>> sent = nltk.corpus.brown.sents('a')[2007]
>>> unigram_tagger.tag(sent)
[('Various', None), ('of', 'IN'), ('the', 'AT'), ('apartments',
('of', 'IN'), ('the', 'AT'), ('terrace', 'NN'), ('type', 'NN'),
('being', 'BEG'), ('on', 'IN'), ('the', 'AT'), ('ground', 'NN'),
('so', 'QL'), ('that', 'CS'), ('entrance', 'NN'), ('is', 'BEZ'),
>>> nltk.tag.accuracy(unigram_tagger, brown_a)
0.8550331165343994
```

The UnigramTagger assigns the default tag `None` to words that are not in the training data (eg *Various*)



## Using a backoff

We can combine taggers to ensure every word is tagged:

```
>>> t0 = nltk.DefaultTagger('NN')
>>> t1 = nltk.UnigramTagger(brown_a, backoff=t0)
```

## Evaluating taggers

| Sentence | Gold Standard | Unigram Tagger |
|---|---|---|
| The | at | at |
| President | nn-tl | nn-tl |
| said | vbd | vbd |
| he | pps | pps |
| will | md | md |
| ask | vb | vb |
| Congress | np | np |
| to | to | to |
| increase | vb | *nn* |
| grants | nns | nns |
| to | in | *to* |
| states | nns | nns |
| . | . | . |

The tagger correctly tagged 10 out of 12 words.
Accuracy = 10/12 or 83%



## 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 `nltk.tag.accuracy` to automate evaluation.

```
>>> acc = nltk.tag.accuracy(unigram_tagger, train_sents
>>> print 'Accuracy = %4.1f%%' % (100 * acc)
Accuracy = 81.8%
```



## Evaluating taggers in NLTK

Training and testing corpus should be distinct:

```
>>> nltk.tag.accuracy(unigram_tagger, train_sents)
93.8%
>>> nltk.tag.accuracy(unigram_tagger, test_sents)
82.8%
```

## Error analysis

- The % correct score doesn't tell you everything — it is useful 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 (eg NNP vs. NN vs. JJ)
- See section 3 of the NLTK tutorial on Tagging



## Other approaches to POS Tagging

- Rule-based (briefly discussed in Jurafsky and Martin,, sec 8.4)
- N-gram / HMM based tagging (eg TnT)
- Transformation-based learning (the Brill tagger)



## n-gram based tagging

- Also referred to as hidden Markov model (HMM) tagging
- Basic idea: Choose the tag that maximises:

$$P(\text{word}|\text{tag}) \cdot P(\text{tag}|\text{previous n tags})$$

- For a bigram model the best tag at position $i$ is:

$$t_i = \arg\max_{t_j} P(w_i|t_j)P(t_j|t_{i-1})$$

  Assuming that you know the previous tag, $t_{i-1}$.
- Interpretation: choose the tag $t_i$ that is most likely to generate word $w_i$ given that the previous tag was $t_{i-1}$



## Example (J+M, p304)

Secretariat/NNP is/VBZ expected/VBZ to/TO race/VB tomorrow/NN

People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT race/NN for/IN outer/JJ space/NN

- "race" is a verb in the first, a noun in the second.
- Assume that race is the only untagged word, so we can assume the tags of the others.
- Probabilities of "race" being a verb, or race being a noun in the first example:

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

## Example (continued)

$$P(NN|TO) = 0.021$$
$$P(VB|TO) = 0.34$$

$$P(\text{race}|NN) = 0.00041$$
$$P(\text{race}|VB) = 0.00003$$

$$\begin{aligned} P(\text{race is } VB) &= P(VB|TO)P(\text{race}|VB) \\ &= 0.34 \times 0.00003 = 0.00001 \\ P(\text{race is } NN) &= P(NN|TO)P(\text{race}|NN) \\ &= 0.021 \times 0.00041 = 0.000007 \end{aligned}$$



## Limitation of n-gram taggers

- Size of n-gram table (*Language model*): hundred of millions of entries
- Context limited to tags; Information about preceding or neighbouring word is often useful



## Brill tagging

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



## Transformation-based tagging

Basic idea:

1. Label every word with its most likely tag (Unigram tagging)
2. Generate finite set of transformation rules (*In context C, replace T1 with T2*)
3. Select best rule i.e., rule that most improve tagging
4. Re-tag the data using this rule

Repeat 1-3 until stopping criteria is met (e.g., insufficient improvment over previous pass).
Output: ordered list of transformations that can be applied to a new corpus; tagging procedure.

## The transformations

- TBL considers every possible transformation in order to pick the best one
- Hence the set of possible transformations must be finite
- This is done by defining a set of templates; every transformation is an instantiation of one such template



## Brill set of templates

Change tag a to tag b when:

- the preceding/following word is tagged z
- the second preceding/following word is tagged z
- one of the three 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 second preceding/following word is tagged w

The set of possible transformations is the set of all possible instantiations of the above rules given a finite set of tags.



## Selecting the best transformation

- The best transformation is the transformation with highest net benefit
- The net benefit of a transformation is the number of incorrect tags corrected minus the number of correct tags modified



## Example output

```
Loading tagged data...
Training unigram tagger: [accuracy: 0.820940]
Training Brill tagger on 37168 tokens...

Iteration 1: 1482 errors; ranking 23989 rules;
  Found: "Replace POS with VBZ if the preceding word is tagged PRP"
  Apply: [changed 39 tags: 39 correct; 0 incorrect]

Iteration 2: 1443 errors; ranking 23662 rules;
  Found: "Replace VBP with VB if one of the 3 preceding words is tagg
  Apply: [changed 36 tags: 36 correct; 0 incorrect]

  ...
Iteration 20: 1138 errors; ranking 20717 rules;
  Found: "Replace RBR with JJR if one of the 2 following words is tag
  Apply: [changed 14 tags: 10 correct; 4 incorrect]

Iteration 21: 1128 errors; ranking 20569 rules;
  Found: "Replace VBD with VBN if the preceding word is tagged VBD"
[insufficient improvement; stopping]

Brill accuracy: 0.835145
```

## Summary

- Parts of speech and tagsets
- Tagging
- Constructing simple taggers in NLTK
- Evaluating taggers
- Other types of taggers (n-gram, TBL)



## Reading


- Jurafsky and Martin, chapter 8 (esp. sec 8.5);
- Manning and Schütze, chapter 10
  Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999.
- T. Brants (2000). "TnT – a statistical part-of-speech tagger". In Proceedings of the 6th Applied NLP Conference, ANLP-2000.
  `http://uk.arxiv.org/abs/cs.CL/0003055`
- Schuetze, H. Distributional Part-of-Speech Tagging. Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics, 1995.
  `ftp://csli.stanford.edu/pub/prosit/DisPosTag.ps`