Tagging

Steven Bird    Ewan Klein    Edward Loper

University of Melbourne, AUSTRALIA
University of Edinburgh, UK
University of Pennsylvania, USA

April 28, 2008
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: e.g. Penn Treebank (45 tags), Susanne (353 tags).
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).
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

- 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).

- 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
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
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 (e.g., adjectives often followed by nouns, personal pronouns often followed by verbs, possessive pronouns by nouns)
- Pronunciations can be dependent on part of speech, e.g., 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
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
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
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 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 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 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

<table>
<thead>
<tr>
<th>Prepositions</th>
<th>on, under, over, to, with, by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determiners</td>
<td>the, a, an, some</td>
</tr>
<tr>
<td>Pronouns</td>
<td>she, you, I, who</td>
</tr>
<tr>
<td>Conjunctions</td>
<td>and, but, or, as, when, if</td>
</tr>
<tr>
<td>Auxiliary Verbs</td>
<td>can, may, are</td>
</tr>
<tr>
<td>Particles</td>
<td>up, down, at, by</td>
</tr>
<tr>
<td>Numerals</td>
<td>one, two, first, second</td>
</tr>
</tbody>
</table>
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
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
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
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
Open classes

**nouns**  Proper nouns (**Scotland**, **BBC**),
common nouns:
  - count nouns (**goat**, **glass**)
  - mass nouns (**snow**, **pacificism**)

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

<table>
<thead>
<tr>
<th>RP</th>
<th>Particle</th>
<th>up, off</th>
</tr>
</thead>
<tbody>
<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: POS 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)
Tagging

• Definition: POS 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 (e.g., prepositional phrase attachment)
Tagging

• Definition: POS 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)
Tagging

- Definition: POS Tagging is the assignment of a single part-of-speech tag to each word (and punctuation marker) in a corpus. For example:
  
  "The guys that make traditional hardware are really being obsoleted by microprocessor-based machines," said Mr. Benton.

- 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 (e.g., prepositional phrase attachment).
Tagging

• Definition: POS 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
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
import nltk
mytagger = nltk.DefaultTagger('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:

text = 'There are 11 players in a football team'
text_tokens = text.split()
# ['There', 'are', '11', 'players', 'in', 'a', 'football

import nltk
mytagger = nltk.DefaultTagger('NN')
for t in mytagger.tag(text_tokens):
    print t
# ('There', 'NN')
# ('are', 'NN')
# ('11', 'NN')
# ('players', 'NN')
# ('in', 'NN')
# ('a', 'NN')
# ('football', 'NN')

A regular expression tagger

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

default_pattern = (r'.*', 'NN')
cd_pattern = (r'^[0-9]+(\.[0-9]+)?$', 'CD')
patterns = [cd_pattern, default_pattern]
NN_CD_tagger = nltk.RegexpTagger(patterns)
re_tagged = 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:

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

Training a UnigramTagger on the Penn Treebank:

```python
# sentences 0–2999
train_sents = nltk.corpus.treebank.tagged_sents()[:3000]
# from sentence 3000 to the end
test_sents = nltk.corpus.treebank.tagged_sents()[3000:]
unigram_tagger = nltk.UnigramTagger(train_sents)
```
A unigram tagger

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

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

Training a UnigramTagger on the Penn Treebank:

```python
# sentences 0-2999
train_sents = nltk.corpus.treebank.tagged_sents()[:3000]
# from sentence 3000 to the end
test_sents = nltk.corpus.treebank.tagged_sents()[3000:]

unigram_tagger = nltk.UnigramTagger(train_sents)
```
Unigram tagging

>>> sent = "Mr. Jones saw the book on the shelf"
>>> unigram_tagger.tag(sent.split())
[['Mr.', 'NNP'), ('Jones', 'NNP'), ('saw', 'VBD'), ('the', 'DT'), ('book', 'NN'), ('on', 'IN'), ('the', 'DT'), ('shelf', 'NN')]

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 = nltk.UnigramTagger(train_sents, cutoff=0, backoff=NN_CD_tagger)
>>> unigram_tagger.tag(sent.split())
[['Mr.', 'NNP'), ('Jones', 'NNP'), ('saw', 'VBD'), ('the', 'DT'), ('book', 'NN'), ('on', 'IN'), ('the', 'DT'), ('shelf', 'NN')]
Unigram tagging

```python
>>> sent = "Mr. Jones saw the book on the shelf"
>>> unigram_tagger.tag(sent.split())
[('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 = nltk.UnigramTagger(train_sents, cutoff=0, backoff=NN_CD_tagger)
>>> unigram_tagger.tag(sent.split())
[('Mr.', 'NNP'), ('Jones', 'NNP'), ('saw', 'VB'), ('the', 'DT'), ('book', 'NN'), ('on', 'IN'), ('the', 'DT'), ('shelf', 'NN')]
```
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

• 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

- 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

- 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

- 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.

We can make print more prettily:

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

>>> 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%
```
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 * nltk.tag.accuracy(tagger, data))

>>> 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%
```
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 * nltk.tag.accuracy(tagger, data))

>>> 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
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 (eg NNP vs. NN vs. JJ).

- See section 3 of the NLTK tutorial on Tagging.
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
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
Error analysis

- The % correct score doesn’t tell you everything — it is useful know what is misclassified as what
- **Confusion matrix**: A matrix (ntag 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
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 (eg NNP vs. NN vs. JJ).
- See section 3 of the NLTK tutorial on Tagging.
N-gram taggers

- 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 \( t_{i-1} \).
N-gram taggers

• 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:

\[ t_i = \arg \max_j P(w_i|t_j)P(t_j|t_{i-1}) \]

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 \( t_{i-1} \).
N-gram taggers

• 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 \( t_{i-1} \)
N-gram taggers

**Tokens:**
- $w_{n-2}$
- $w_{n-1}$
- $w_n$
- $w_{n+1}$

**Tags:**
- $t_{n-2}$
- $t_{n-1}$
- $t_n$
- $t_{n+1}$

(context)
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:

\[
P(\text{race is } \text{VB}) = P(\text{VB}|\text{TO})P(\text{race}|\text{VB})
\]
\[
P(\text{race is } \text{NN}) = P(\text{NN}|\text{TO})P(\text{race}|\text{NN})
\]
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:

\[
P(\text{race is } VB) = P(\text{VB}|\text{TO})P(\text{race}|\text{VB})
\]
\[
P(\text{race is } NN) = P(\text{NN}|\text{TO})P(\text{race}|\text{NN})
\]
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:

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

\[ P(\text{NN}|\text{TO}) = 0.021 \]
\[ P(\text{VB}|\text{TO}) = 0.34 \]

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

\[ P(\text{race is VB}) = P(\text{VB}|\text{TO})P(\text{race}|\text{VB}) \]
\[ = 0.34 \times 0.00003 = 0.00001 \]

\[ P(\text{race is NN}) = P(\text{NN}|\text{TO})P(\text{race}|\text{NN}) \]
\[ = 0.021 \times 0.00041 = 0.000007 \]
Simple bigram tagging in NLTK

```python
>>> default_pattern = (r'.*', 'NN')
>>> cd_pattern = (r'^[0-9]+\.(.[0-9]+)?$', 'CD')
>>> patterns = [cd_pattern, default_pattern]
>>> NN_CD_tagger = nltk.RegexpTagger(patterns)
>>> unigram_tagger = nltk.UnigramTagger(train_sents, cutoff=0, backoff=NN_CD_tagger)
>>> bigram_tagger = tag.BigramTagger(train_sents, backoff=unigram_tagger)

>>> print_accuracy(bigram_tagger, train_sents)
95.6%
>>> print_accuracy(bigram_tagger, test_sents)
84.2%
```
Limitation of NLTK n-gram taggers

- Does not find the most likely sequence of tags, simply works left to right always assigning the most probable single tag (given the previous tag assignments)
- Does not cope with zero probability problem well (no smoothing or discounting)
- See module `nltk.tag.hmm`
Brill Tagger

- Problem with n-gram taggers: size
  - 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 (painting analogy)
  - 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
Brill Tagger

- Problem with n-gram taggers: size
- 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 (painting analogy)
  - 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
Brill Tagger

• Problem with n-gram taggers: size
• 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 (painting analogy)
• 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
Brill Tagger

- Problem with n-gram taggers: size
- 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 (painting analogy)
  - Learn the rules from a corpus, using a set of rule templates, eg:
    Change tag \texttt{a} to \texttt{b} when the following word is tagged \texttt{z}
  - Choose the best rule each iteration
Brill Tagger

- Problem with n-gram taggers: size
- 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 (painting analogy)
- 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
Brill Tagger

- Problem with n-gram taggers: size
- 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 (painting analogy)
- 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
## Brill Tagger: Example

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Gold</th>
<th>Unigram</th>
<th>Replace NN with VB when the previous word is TO</th>
<th>Replace TO with IN when the next tag is</th>
</tr>
</thead>
<tbody>
<tr>
<td>The President</td>
<td>AT</td>
<td>AT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>said</td>
<td>VBD</td>
<td>VBD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>he</td>
<td>PPS</td>
<td>PPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>will</td>
<td>MD</td>
<td>MD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ask</td>
<td>VB</td>
<td>VB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Congress to</td>
<td>NP</td>
<td>NP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>increase grants</td>
<td>VB</td>
<td>NN</td>
<td>VBD</td>
<td></td>
</tr>
<tr>
<td>to states</td>
<td>IN</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>for</td>
<td>IN</td>
<td>IN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vocational</td>
<td>JJ</td>
<td>JJ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rehabilitation</td>
<td>NN</td>
<td>NN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary

- **Reading:** Jurafsky and Martin, chapter 8 (esp. sec 8.5); Manning and Schütze, chapter 10;
  - Rule-based and statistical tagging
  - HMMs and n-grams for statistical tagging
  - Operation of a simple bigram tagger
  - TnT — an accurate trigram-based tagger
Summary

- **Reading:** Jurafsky and Martin, chapter 8 (esp. sec 8.5); Manning and Schütze, chapter 10;
- Rule-based and statistical tagging
  - HMMs and n-grams for statistical tagging
  - Operation of a simple bigram tagger
  - TnT — an accurate trigram-based tagger
Summary

- **Reading**: Jurafsky and Martin, chapter 8 (esp. sec 8.5); Manning and Schütze, chapter 10;
- Rule-based and statistical tagging
- HMMs and n-grams for statistical tagging
  - Operation of a simple bigram tagger
  - TnT — an accurate trigram-based tagger
Summary

- **Reading:** Jurafsky and Martin, chapter 8 (esp. sec 8.5); Manning and Schütze, chapter 10;
- Rule-based and statistical tagging
- HMMs and n-grams for statistical tagging
- Operation of a simple bigram tagger
- TnT — an accurate trigram-based tagger
Summary

- **Reading**: Jurafsky and Martin, chapter 8 (esp. sec 8.5); Manning and Schütze, chapter 10;
- Rule-based and statistical tagging
- HMMs and n-grams for statistical tagging
- Operation of a simple bigram tagger
- TnT — an accurate trigram-based tagger