Part of Speech Tagging

MAS.S60
Catherine Havasi
Rob Speer
Tagsets

• What is a tagset?
• Standards and tagging
• Tags for parts of speech:
  – Nouns, verbs, adverbs, adjectives, articles, etc
  – Subtagging
    • nouns can be singular or plural
    • verbs have tenses
  – Different tagsets have different focuses
Tags are cryptic

>>> text = nltk.word_tokenize("And now for something completely different")
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),
('completely', 'RB'), ('different', 'JJ')]

• Brown and Treebank established some cryptic tags; everyone tends to use Treebank’s
  – CC = coordinating conjunction
  – RB = adverb
  – IN = preposition
  – NN = noun
  – JJ = adjective
NLTK can help

• We don’t even remember what all the tags mean sometimes
• but nltk.help.upenn_tagset(tag) does!
Homographs

>>> text = nltk.word_tokenize("They refuse to permit us to obtain the refuse permit")

>>> nltk.pos_tag(text)
[('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'), ('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')]
Tags in NLTK

```python
>>> tagged_token = nltk.tag.str2tuple('fly/NN')
>>> tagged_token
('fly', 'NN')
>>> tagged_token[0]
'fly'
>>> tagged_token[1]
'NN'
```

• Tags are tuples
• Tags can be converted by NLTK between tagsets
Making tuples from a corpus

```python
>>> sent = '''
... The/AT grand/JJ jury/NN commented/VBD on/IN a/AT number/NN of/IN
... other/AP topics/NNS ,/, AMONG/IN them/PPO the/AT
Atlanta/NP and/CC
... Fulton/NP-tl County/NN-tl purchasing/VBG departments/NNS
which/WDT it/PPS
... said/VBD `\``/`` ARE/BER well/QL operated/VBN and/CC
follow/VB generally/RB
... accepted/VBN practices/NNS which/WDT inure/VB to/IN
the/AT best/JJT
... interest/NN of/IN both/ABX governments/NNS ''/'' ./.
... '''

>>> [nltk.tag.str2tuple(t) for t in sent.split()]
[('The', 'AT'), ('grand', 'JJ'), ('jury', 'NN'), ('commented', 'VBD'),
('on', 'IN'), ('a', 'AT'), ('number', 'NN'), ... ('.', '.')]```
Many Tag Sets

• Different corpora have different conventions for tagging.
• There are ISO standards for tagging...
• There were many boring meetings
• NLTK made a simplified, unified tagset
• ... which no one uses.
## Simplified Tagset of NLTK

<table>
<thead>
<tr>
<th>Tag</th>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>adjective</td>
<td>new, good, high, special, big, local</td>
</tr>
<tr>
<td>ADV</td>
<td>adverb</td>
<td>really, already, still, early, now</td>
</tr>
<tr>
<td>CNJ</td>
<td>conjunction</td>
<td>and, or, but, if, while, although</td>
</tr>
<tr>
<td>DET</td>
<td>determiner</td>
<td>the, a, some, most, every, no</td>
</tr>
<tr>
<td>EX</td>
<td>existential</td>
<td>there, there's</td>
</tr>
<tr>
<td>FW</td>
<td>foreign word</td>
<td>dolce, ersatz, esprit, quo, maitre</td>
</tr>
<tr>
<td>MOD</td>
<td>modal verb</td>
<td>will, can, would, may, must, should</td>
</tr>
<tr>
<td>N</td>
<td>noun</td>
<td>year, home, costs, time, education</td>
</tr>
<tr>
<td>NP</td>
<td>proper noun</td>
<td>Alison, Africa, April, Washington</td>
</tr>
<tr>
<td>NUM</td>
<td>number</td>
<td>twenty-four, fourth, 1991, 14:24</td>
</tr>
<tr>
<td>PRON</td>
<td>pronoun</td>
<td>he, their, her, its, my, I, us</td>
</tr>
<tr>
<td>P</td>
<td>preposition</td>
<td>on, of, at, with, by, into, under</td>
</tr>
<tr>
<td>TO</td>
<td>the word to</td>
<td>to</td>
</tr>
<tr>
<td>UH</td>
<td>interjection</td>
<td>ah, bang, ha, whee, hmpf, oops</td>
</tr>
<tr>
<td>V</td>
<td>verb</td>
<td>is, has, get, do, make, see, run</td>
</tr>
<tr>
<td>VD</td>
<td>past tense</td>
<td>said, took, told, made, asked</td>
</tr>
<tr>
<td>VG</td>
<td>present participle</td>
<td>making, going, playing, working</td>
</tr>
<tr>
<td>VN</td>
<td>past participle</td>
<td>given, taken, begun, sung</td>
</tr>
<tr>
<td>WH</td>
<td>wh determiner</td>
<td>who, which, when, what, where, how</td>
</tr>
</tbody>
</table>
Tagging in Other Languages

Bangla: কুঁড়ে হেরেন্দ্র লি/’NN' আকার/’NN' বাংলার/’NNP' বা/’CC' ভারতে/’NNP' ?/None ন্য/’JJ' ?/None এঁচলের/’NN' প্রচল তি/’JJ' কুঁড়ে/’NN' ঘর/’NN' নয়/’VM' [ক/’SYM'

Hindi: पाकिस्तान/’NNP' की/’PREP' पूर्व/’JJ' प्रधानमंत्री/’NN' राष्ट्रपति/’NNPC' भुज्दो/’NNP' पर/’PREP' लगे/’VFM' पहचान/’NN' के/’PREP' आरोपो/’NN' के/’PREP' खिलाफ/’PREP' भुज्दो/’NNP' द्वारा/’PREP' दायर/’NVB' की/’VFM' गई/’VAUX' एवं अतिक्रमण/’NN' की/’PREP' सुनचाई/’NN' में गलवार/’NN' को/’PREP' वकीलो/’NN' की/’PREP' हठलाल/’NN' के/’PREP' कारण/’PREP' स्थिति/’JVBJ' कर/’VFM' दी/’VAUX' गई/’VAUX' और/’PUNC'

Marathi: ग्रामीण/’JJ' जिल्हा/’NN' बांधावहें/’NNPC' भोसले/’NNP' यांच्या/’PRP' ?/None ह्यांतरेच्या/’NN' फर्शाची/’NN' आज/’NN' वै?/None क/’NN' वाली/’VM' ./’SYM'

Telugu: గ్రామయం/’NN' ఆపం/’PREP' దేశం/’VJJ' ఒందు/’NN' మి/’PREP' తెలుసుతుంది/’NN'
>>> from nltk.corpus import brown
>>> brown_news_tagged =
brown.tagged_words(categories='news',
simplify_tags=True)
>>> tag_fd = nltk.FreqDist(tag for (word, tag) in
brown_news_tagged)
>>> tag_fd.keys()
['N', 'P', 'DET', 'NP', 'V', 'ADJ', ',', '.', 'CNJ',
'PRO', 'ADV', 'VD', ...]
Finding the Verbs

```python
>>> wsj =
nltk.corpus.treebank.tagged_words(simplify_tags=True)
>>> word_tag_fd = nltk.FreqDist(wsj)
>>> [word + "/" + tag for (word, tag) in
word_tag_fd if tag.startswith('V')]
['is/V', 'said/VD', 'was/VD', 'are/V', 'be/V', 'has/V', 'have/V', 'says/V', 'were/VD', 'had/VD', 'been/VN', "'s/V", 'do/V', 'say/V', 'make/V', 'did/VD', 'rose/VD', 'does/V', 'expected/VN', 'buy/V', 'take/V', 'get/V', 'sell/V', 'help/V', 'added/VD', 'including/VG', 'according/VG', 'made/VN', 'pay/V', ...
```
Evaluation

- Gold standard: Corpus that has been manually annotated.
- Usually have a training and test component; ideally a portion held out for final evaluation
- Compare algorithms against each other
- Inner Annotator Agreement (IAA)
Data Set Partitioning

```python
>>> size = int(len(brown_tagged_sents) * 0.9)
>>> size
4160
>>> size2 = int(len(brown_tagged_sents) * 0.95)
>>> train_sents = brown_tagged_sents[:size]
>>> test_sents = brown_tagged_sents[size:size2]
>>> unigram_tagger = nltk.UnigramTagger(train_sents)
>>> unigram_tagger.evaluate(test_sents)
0.81202033290142528
```
Tagging in NLTK

• The “Default” tagger tags each token with the most common tag.
Default Tagger in Action

```python
>>> tags = [tag for (word, tag) in 
brown.tagged_words(categories='news')]
>>> nltk.FreqDist(tags).max()
'NN'
>>> raw = 'I do not like green eggs and ham, I do not like them Sam I am'
>>> tokens = nltk.word_tokenize(raw)
>>> default_tagger = nltk.DefaultTagger('NN')
>>> default_tagger.tag(tokens)
[('I', 'NN'), ('do', 'NN'), ('not', 'NN'), ('like', 'NN'), ('green', 'NN'), ('eggs', 'NN'), ('and', 'NN'), ('ham', 'NN'), (',', 'NN'), ('I', 'NN'), ('do', 'NN'), ('not', 'NN'), ('like', 'NN'), ('them', 'NN'), ('Sam', 'NN'), ('I', 'NN'), ('am', 'NN'), ('!', 'NN')]
>>> default_tagger.evaluate(brown_tagged_sents)
0.13089484257215028
```
Regular Expression Tagging

• Uses human-defined patterns:

```python
>>> 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 num.
...    (r'.*', 'NN')  # nouns (default)
... ]
```
RE Tagger in Action

```python
>>> regexp_tagger = nltk.RegexpTagger(patterns)
>>> regexp_tagger.tag(brown_sents[3])
[('``', 'NN'), ('Only', 'NN'), ('a', 'NN'), ('relative', 'NN'),
('handful', 'NN'), ('of', 'NN'), ('such', 'NN'), ('reports',
'NNS'), ('was', 'NNS'), ('received', 'VBD'), ('''', 'NN'), (',',
'NN'), ('the', 'NN'), ('jury', 'NN'), ('said', 'NN'), (',', ',
'NN'), (```, 'NN'), ('considering', 'VBG'), ('the', 'NN'),
('widespread', 'NN'), ...]
```

```python
>>> regexp_tagger.evaluate(brown_tagged_sents)
0.20326391789486245
```
Unigram Tagger

• Otherwise known as the “lookup tagger”
• Stores and looks up the tag for each word

```python
>>> fd = nltk.FreqDist(brown.words(categories='news'))
>>> cfd = nltk.ConditionalFreqDist(brown.tagged_words(categories='news'))
>>> most_freq_words = fd.keys()[:100]
>>> likely_tags = dict((word, cfd[word].max()) for word in most_freq_words)
>>> baseline_tagger = nltk.UnigramTagger(model=likely_tags)
>>> baseline_tagger.evaluate(brown_tagged_sents)
0.45578495136941344
```
Unigram Tagger Performance

Lookup Tagger Performance with Varying Model Size

Performance vs. Model Size graph showing the performance of a lookup tagger improving with increasing model size.
Smaller Data Unigram

- Store less data – only pay attention to non-nouns?
- Saves ~18%
- Does this matter?
General N-Gram Tagging

- Current word plus context: the n-1 previous POS tags.
The Bigram Tagger

```python
>>> bigram_tagger = nltk.BigramTagger(train_sents)
>>> bigram_tagger.tag(brown_sents[2007])
[('Various', 'JJ'), ('of', 'IN'), ('the', 'AT'), ('apartments', 'NNS'), ('are', 'BER'), ('of', 'IN'), ('the', 'AT'), ('terrace', 'NN'), ('type', 'NN'), (',', ','), ('being', 'BEG'), ('on', 'IN'), ('the', 'AT'), ('ground', 'NN'), ('floor', 'NN'), ('so', 'CS'), ('that', 'CS'), ('entrance', 'NN'), ('is', 'BEZ'), ('direct', 'JJ'), (',', ',')]
```

```python
>>> unseen_sent = brown_sents[4203]
>>> bigram_tagger.tag(unseen_sent)
[('The', 'AT'), ('population', 'NN'), ('of', 'IN'), ('the', 'AT'), ('Congo', 'NP'), ('is', 'BEZ'), ('13.5', None), ('million', None), (',', None), ('divided', None), ('into', None), ('at', None), ('least', None), ('seven', None), ('major', None), (',', None), ('culture', None), ('clusters', None), (',', None), ('and', None), ('innumerable', None), ('tribes', None), ('speaking', None), ('400', None), ('separate', None), ('dialects', None), (',', None)]
```
Evaluating the Bigram Tagger

```python
>>> bigram_tagger.evaluate(test_sents)
0.10276088906608193
```

Ick.
Why so bad?

- The bigram tagger has no **backoff**
- The unigram and regex taggers and a default assumption...
- Backoff as a baseline (Semeval)
Layering taggers

• Use the benefits of several types of taggers
  – Try the bigram tagger
  – When it is unable to find a tag, use the unigram tagger
  – If that fails, then use the default tagger
Evaluating the Layered Tagger

```python
>>> t0 = nltk.DefaultTagger('NN')
>>> t1 = nltk.UnigramTagger(train_sents, backoff=t0)
>>> t2 = nltk.BigramTagger(train_sents, backoff=t1)
>>> t2.evaluate(test_sents)
0.84491179108940495
```
That’s awesome, right?
What’s wrong with this picture?

• Size of their n-gram table language model
• Context!
  – Words, not just pos, matter in context
• Understandable rules
The Brill Tagger

• A “Transformation based” tagger
• Guess the tag of each word – then go back and fix mistakes
• Compiles a list of correctional rules
Brill Output

```python
>>> nltk.tag.brill.demo()
Training Brill tagger on 80 sentences...
Finding initial useful rules...
    Found 6555 useful rules.

B
S Fr O         Score = Fixed - Broken
C i o t R     Fixed = num tags changed incorrect -> correct
O x k h u     Broken = num tags changed correct -> incorrect
r e e e l     Other = num tags changed incorrect -> incorrect
c d n r e

12 13 1 4   NN -> VB if the tag of the preceding word is 'TO'
8 9 1 23  NN -> VBD if the tag of the following word is 'DT'
8 8 0 9  NN -> VBD if the tag of the preceding word is 'NNS'
6 9 3 16  NN -> NNP if the tag of words i-2...i-1 is 'NONE'
5 8 3 6  NN -> NNP if the tag of the following word is 'NNP'
5 6 1 0  NN -> NNP if the text of words i-2...i-1 is 'like'
5 5 0 3  NN -> VBN if the text of the following word is '*-1'
...

>>> print(open("errors.out").read())

<table>
<thead>
<tr>
<th>left context</th>
<th>word/test-&gt;gold</th>
<th>right context</th>
</tr>
</thead>
<tbody>
<tr>
<td>, in/IN the/DT guests/NNS</td>
<td>Then/NN-&gt;RB</td>
<td>,/, in/IN the/DT guests/NNS</td>
</tr>
<tr>
<td>'/POS honor/NN ,/, the/DT</td>
<td>'/VBD-&gt;POS</td>
<td>honor/NN ,/, the/DT</td>
</tr>
<tr>
<td>NN ,/, the/DT</td>
<td>NN -&gt; VBD</td>
<td>,/, the/DT speed</td>
</tr>
<tr>
<td>the/DT</td>
<td>DT speedway/NNS</td>
<td>speedway/JJ-&gt;NN</td>
</tr>
<tr>
<td>NNS hauled/VBD</td>
<td>NN</td>
<td>hauled/NN-&gt;VBD</td>
</tr>
<tr>
<td>DT hauled/VBD</td>
<td>/NN-&gt;VBD</td>
<td>out/NNP-&gt;RP</td>
</tr>
<tr>
<td>hauled/VBD</td>
<td>/NN-&gt;VBD</td>
<td>four/NNP-&gt;CD</td>
</tr>
<tr>
<td>out/RP</td>
<td>/NN-&gt;VBD</td>
<td>drivers/NNP-&gt;NNS</td>
</tr>
<tr>
<td>four/CD</td>
<td>/NN-&gt;VBD</td>
<td>crews/NN-&gt;NNS</td>
</tr>
<tr>
<td>drivers/NNS</td>
<td>/NN-&gt;VBD</td>
<td>official/NNP-&gt;JJ</td>
</tr>
<tr>
<td>crews/NN</td>
<td>/NN-&gt;VBD</td>
<td>After/VBD-&gt;IN</td>
</tr>
<tr>
<td>NNS and/CC</td>
<td>/NN-&gt;VBD</td>
<td>Fortune/IN-&gt;NNP</td>
</tr>
<tr>
<td>even/RB the/DT</td>
<td>/NN-&gt;VBD</td>
<td>schoolboys/NNP-&gt;NNS</td>
</tr>
<tr>
<td>ter/IN the/DT race/NN ,/s</td>
<td>/NN-&gt;VBD</td>
<td>cars/NN-&gt;NNS</td>
</tr>
<tr>
<td>s/NNS cheered/VBD like/IN</td>
<td>/NN-&gt;VBD</td>
<td>over/IN the/DT cars/NNS</td>
</tr>
</tbody>
</table>
| colboys/NNS | /NN->VBD | and/CC drivers/NNS ,/.
```
Evaluating the Brill tagger

• Train over 3500 Treebank sentences
  – Unigram: 89.0%
  – Bigram: 89.7%
  – Brill: 90.1%

• Incremental improvements and NLP publishing
In-class lab: improve the Brill tagger

• The Brill tagging demo is easy to modify
• Look at the error list: you probably see some simple rules that would help
  – Which rules?
  – Where in the chain do you add them?
• Try out taggers with different rules. Who can get the most accuracy?
Taggers are just classifiers

• What the Brill tagger is doing may seem familiar from the classifier lecture
• The modern approach: just turn your training data into features and throw them into a good classifier
• We don’t have time to train a good classifier, so here’s Naïve Bayes again
92% accuracy!

• Imagine what we could do with a better classifier
  – Hidden Markov Models
  – Maximum Entropy decision trees
MaxEnt gets > 99% correct

• ...on the Wall Street Journal
• This is the tagger built in as nltk.pos_tag()
Slide & Graphic Thanks

- Some slides, code, graphics: NLTK & Ed Loper
- Powerpoint of NLTK code: Lillian Cassel