### Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes

<table>
<thead>
<tr>
<th>Open class (lexical) words</th>
<th>Nouns</th>
<th>Verbs</th>
<th>Adjectives</th>
<th>Adverbs</th>
<th>Numbers</th>
<th>Prepositions</th>
<th>Particles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proper</td>
<td>IBM</td>
<td>Common</td>
<td>Main</td>
<td>Adverbs</td>
<td>Numbers</td>
<td>Prepositions</td>
<td>Particles</td>
</tr>
<tr>
<td></td>
<td>Italy</td>
<td>cat / cats</td>
<td>see</td>
<td>slowly</td>
<td>122,312</td>
<td>to with</td>
<td>off up</td>
</tr>
<tr>
<td></td>
<td></td>
<td>snow</td>
<td>registered</td>
<td></td>
<td>one</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Closed class (functional)</th>
<th>Determiners</th>
<th>Conjunctions</th>
<th>Modals</th>
<th>... more</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>the some</td>
<td>and or</td>
<td>can</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>had</td>
<td></td>
</tr>
</tbody>
</table>

- ... more
### Part-of-Speech Ambiguity

#### Example

<table>
<thead>
<tr>
<th>VBD</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBN</td>
<td>VBZ</td>
</tr>
<tr>
<td>VBP</td>
<td>VBZ</td>
</tr>
<tr>
<td>NNP</td>
<td>NNS</td>
</tr>
<tr>
<td>NN</td>
<td>NNS</td>
</tr>
<tr>
<td>CD</td>
<td>NN</td>
</tr>
</tbody>
</table>

Fed raises interest rates 0.5 percent

Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG
All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN
Chateau/NNP Peters/NNP costs/VBN around/RB 250/CD

#### Two basic sources of constraint:
- Grammatical environment
- Identity of the current word

#### Many more possible features:
- … but we won’t be able to use them for a while

---

| CC     | conjunction, coordinating | and both but either or |
| CD     | numeral, cardinal         | mid-1890 nine-thirty 0.5 one |
| DT     | determiner                | a all an every no that the |
| EX     | existential there         | there |
| FW     | foreign word              | gemeinschaft hund ich jeux |
| IN     | preposition or conjunction, subordinating | among whether out on by if |
| JJ     | adjective or numeral, ordinal | third 3rd-mannered respectable |
| JJR    | adjective, comparative    | braver cheaper taller |
| JJPS   | adjective, superlative    | bravest cheapest tallest |
| MO     | modal auxiliary           | can may might will would |
| NN     | noun, common, singular or mass | cabbage thermostat investment subhumanity |
| NNP    | noun, proper, singular    | Motown Cougar Yvette Liverpool |
| NNPBS  | noun, proper, plural      | Americans Materials States |
| NNS    | noun, common, plural      | undergraduates bric-a-brac averages |
| POS    | genitive marker           | ’s |
| PRP    | pronoun, personal         | hers himself it us them |
| PRPS   | pronoun, possessive       | her his mine my our ours their thy your |
| RB     | adverb                    | occasionally maddeningly adventurously |
| RBR    | adverb, comparative       | further gloomier heavier less-perfectly |
| RBS    | adverb, superlative       | best biggest nearest worst |
| RP     | particle                  | aboard away back by on open through |
| TO     | “to” as preposition or infinitive marker | to |
| UH     | interjection              | huh howdy uh whammo shucks heck |
| VB     | verb, base form           | ask bring fire see take |
| VBD    | verb, past tense          | pleased swiped registered saw |
| VBG    | verb, present participle or gerund | soaring focusing approaching erasing |
| VBN    | verb, past participle     | diapason imitated runneled unsettled |
| VBP    | verb, present tense, not 3rd person singular | twist appear comprise mold postpone |
| VBZ    | verb, present tense, 3rd person singular | bases reconstructs marks uses |
| WDT    | WH-determiner             | that what whatever which whichever |
| WP     | WH-pronoun                | that what whatever which who whom |
| WPS    | WH-pronoun, possessive    | whose |
| WRB    | WH-adverb                 | however whenever wherever why |
Why POS Tagging?

- Useful in and of itself
  - Text-to-speech: record, lead
  - Lemmatization: saw[v] → see, saw[n] → saw
  - Quick-and-dirty NP-chunk detection: grep {JJ | NN}* {NN | NNS}

- Useful as a pre-processing step for parsing
  - Less tag ambiguity means fewer parses
  - However, some tag choices are better decided by parsers

```
IN  DT  NNP    NN  VBD  VBN  RP  NN  NNS
The Georgia branch had taken on loan commitments …
```

```
VDN
DT  NN  IN  NN  VBD  NNS  VBD
The average of interbank offered rates plummeted …
```

HMMs

- We want a model of sequences s and observations w

```
s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_n
W_1 \quad W_2 \quad W_n
```

\[
P(s, w) = \prod_{i} P(s_i|s_{i-1})P(w_i|s_i)
\]

- Assumptions:
  - States are tag n-grams
  - Usually a dedicated start and end state / word
  - Tag/state sequence is generated by a markov model
  - Words are chosen independently, conditioned only on the tag/state
  - These are totally broken assumptions: why?
Transitions and Emissions

Transitions $P(s|s')$ encode well-formed tag sequences
- In a bigram tagger, states = tags

In a trigram tagger, states = tag pairs

Transitions
Estimating Transitions

- Use standard smoothing methods to estimate transitions:
  \[ P(t_i \mid t_{i-1}, t_{i-2}) = \lambda_2 \hat{P}(t_i \mid t_{i-1}, t_{i-2}) + \lambda_1 \hat{P}(t_i \mid t_{i-1}) + (1 - \lambda_1 - \lambda_2) \hat{P}(t_i) \]
- Can get a lot fancier (e.g. KN smoothing), but in this case it doesn’t buy much
- One option: encode more into the state, e.g. whether the previous word was capitalized (Brants 00)

Estimating Emissions

- Emissions are trickier:
  - Words we’ve never seen before
  - Words which occur with tags we’ve never seen
  - One option: break out the Good-Turning smoothing
  - Issue: words aren’t black boxes:
    - 343,127.23 11-year Minteria reintroducibly
  - Unknown words usually broken into word classes
    - D*, D±, D+
    - D*±, x†, Xx†, x†“ly”
  - Another option: decompose words into features and use a maxent model along with Bayes’ rule
    \[ P(w \mid t) = P_{\text{MAXENT}}(t \mid w) P(w) / P(t) \]
Better Features

- Can do surprisingly well just looking at a word by itself:
  - Word: the → DT
  - Lowercased word: Importantly → RB
  - Prefixes: unfathomable → JJ
  - Suffixes: Importantly → RB
  - Capitalization: Meridian: CAP → NNP
  - Word shapes: 35-year: d-x → JJ

- Then build a maxent (or whatever) model to predict tag
- Maxent $P(t|w)$: 93.7% / 82.6%

Disambiguation

- Given these two multinomials, we can score any word / tag sequence pair

Fed raises interest rates 0.5 percent .

$P(NNP|\langle\bullet,\bullet\rangle)$ $P(Fed|NNP)$ $P(VBZ|\langle NNP,\bullet\rangle)$ $P(raises|VBZ)$ $P(NN|VBZ,NNP)$…..

- In principle, we’re done – list all possible tag sequences, score each one, pick the best one (the Viterbi state sequence)

```
NNP VBZ NN NNS CD NN logP = -23
NNP NNS NN NNS CD NN logP = -29
NNP VBZ VB NNS CD NN logP = -27
```
Finding the Best Trajectory

- Too many trajectories (state sequences) to list
- Option 1: Beam Search

A beam is a set of partial hypotheses
- Start with just the single empty trajectory
- At each derivation step:
  - Consider all continuations of previous hypotheses
  - Discard most, keep top k, or those within a factor of the best, (or some combination)

Beam search works relatively well in practice
- ... but sometimes you want the optimal answer
- ... and you need optimal answers to validate your beam search

HMM Trellis
The Viterbi Algorithm

- Dynamic program for computing
  \[ \delta_i(s) = \max_{s_0 \cdots s_{i-1} \leq s} P(s_0 \cdots s_{i-1} s, w_i \cdots w_{i-1}) \]
  - The score of a best path up to position \( i \) ending in state \( s \)
  \[ \delta_0(s) = \begin{cases} 1 & \text{if } s = \langle \bullet, \bullet \rangle \\ 0 & \text{otherwise} \end{cases} \]
  \[ \delta_i(s) = \max_{s'} P(s \mid s') P(w \mid s') \delta_{i-1}(s') \]
  - Also store a backtrace
  \[ \psi_i(s) = \arg \max_{s'} P(s \mid s') P(w \mid s') \delta_{i-1}(s') \]
- Memoized solution
- Iterative solution

So How Well Does It Work?

- Choose the most common tag
  - 90.3% with a bad unknown word model
  - 93.7% with a good one

- TnT (Brants, 2000):
  - A carefully smoothed trigram tagger
  - Suffix trees for emissions
  - 96.7% on WSJ text (SOA is ~97.2%)

- Noise in the data
  - Many errors in the training and test corpora

  DT NN IN NN VBD NNS VBD
  The average of interbank offered rates plummeted ...
  - Probably about 2% guaranteed error from noise (on this data)
### Overview: Accuracies

- **Roadmap of (known / unknown) accuracies:**
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(t|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - MEMM tagger: 96.9% / 86.9%
  - Cyclic tagger: 97.2% / 89.0%
  - Upper bound: ~98%

### What’s Next for POS Tagging

- **Better features!**
  - We could fix this with a feature that looked at the next word
  - We could fix this by linking capitalized words to their lowercase versions
- **Solution:** maximum entropy sequence models (next class)
  - Taggers are already pretty good on WSJ journal text…
  - What the world needs is taggers that work on other text!
Common Errors

- Common errors [from Toutanova & Manning 00]

```
<table>
<thead>
<tr>
<th></th>
<th>JJ</th>
<th>NN</th>
<th>NNP</th>
<th>NNPS</th>
<th>RB</th>
<th>RP</th>
<th>IN</th>
<th>VB</th>
<th>VBD</th>
<th>VBN</th>
<th>VBP</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>JJ</td>
<td>0</td>
<td>177</td>
<td>56</td>
<td>0</td>
<td>61</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>108</td>
<td>0</td>
<td>488</td>
</tr>
<tr>
<td>NN</td>
<td>244</td>
<td>0</td>
<td>103</td>
<td>0</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td>29</td>
<td>5</td>
<td>6</td>
<td>19</td>
<td>525</td>
</tr>
<tr>
<td>NNP</td>
<td>107</td>
<td>106</td>
<td>0</td>
<td>132</td>
<td>5</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>427</td>
</tr>
<tr>
<td>NNPS</td>
<td>1</td>
<td>0</td>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>142</td>
</tr>
<tr>
<td>RB</td>
<td>72</td>
<td>21</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>138</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>298</td>
</tr>
<tr>
<td>RP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>104</td>
</tr>
<tr>
<td>IN</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>169</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>323</td>
</tr>
<tr>
<td>VB</td>
<td>17</td>
<td>64</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>85</td>
<td>189</td>
</tr>
<tr>
<td>VBD</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>143</td>
<td>2</td>
<td>166</td>
</tr>
<tr>
<td>VBN</td>
<td>101</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>108</td>
<td>0</td>
<td>1</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>VBP</td>
<td>5</td>
<td>34</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>49</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>626</td>
<td>536</td>
<td>348</td>
<td>144</td>
<td>317</td>
<td>122</td>
<td>279</td>
<td>102</td>
<td>140</td>
<td>269</td>
<td>108</td>
<td>3651</td>
</tr>
</tbody>
</table>
```

- Official knowledge
- Made up the story
- Recently sold shares

Sequence-Free Tagging?

- What about looking at a word and its environment, but no sequence information?
  - Add in previous / next word: the __
  - Previous / next word shapes: X __ X
  - Occurrence pattern features: [X: x X occurs]
  - Crude entity detection: __ ..... (Inc.|Co.)
  - Phrasal verb in sentence: put ...... __
  - Conjunctions of these things

- All features except sequence: 96.6% / 86.8%
- Uses lots of features: > 200K
- Why isn’t this the standard approach?
Maxent Taggers

- One step up: also condition on previous tags

\[ P(t|w) = \prod_i P_{ME}(t_i|w, t_{i-1}, t_{i-2}) \]

- Train up \( P(t|w, t_{i-1}, t_{i-2}) \) as a normal maxent problem, then use to score sequences
- This is referred to as a maxent tagger [Ratnaparkhi 96]
- Beam search effective! (Why?)
- What’s the advantage of beam size 1?

Feature Templates

- We’ve been sloppy:
  - Features: \(<w_0=\text{future}, t_0=\text{JJ}>\)
  - Feature templates: \(<w_0, t_0>\)

- In maxent taggers:
  - Can now add edge feature templates:
    - \(<t_1, t_0>\)
    - \(<t_2, t_1, t_0>\)
  - Also, mixed feature templates:
    - \(<t_1, w_0, t_0>\)
Decoding

- Decoding maxent taggers:
  - Just like decoding HMMs
  - Viterbi, beam search, posterior decoding
- Viterbi algorithm (HMMs):
  \[ \delta_i(s) = \arg \max_{s'} P(s|s')P(w_{i-1}|s')\delta_{i-1}(s') \]
- Viterbi algorithm (Maxent):
  \[ \delta_i(s) = \arg \max_{s', w} P(s|s', w)\delta_{i-1}(s') \]

TBL Tagger

- [Brill 95] presents a transformation-based tagger
  - Label the training set with most frequent tags
    
    DT MD VBD VBD .
    The can was rusted .

  - Add transformation rules which reduce training mistakes
    
    MD \rightarrow NN : DT _
    VBD \rightarrow VBN : VBD _

  - Stop when no transformations do sufficient good
  - Does this remind anyone of anything?

- Probably the most widely used tagger (esp. outside NLP)
- … but not the most accurate: 96.6% / 82.0 %
### TBL Tagger II

- **What gets learned?** [from Brill 95]

<table>
<thead>
<tr>
<th>Change Tag</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 NN VB</td>
<td>Previous tag is TO</td>
</tr>
<tr>
<td>2 VBP VB</td>
<td>One of the previous three tags is MDP</td>
</tr>
<tr>
<td>3 NN VB</td>
<td>One of the previous two tags is MDP</td>
</tr>
<tr>
<td>4 VB NN</td>
<td>One of the previous two tags is DT</td>
</tr>
<tr>
<td>5 VBD VBN</td>
<td>One of the previous three tags is VBD</td>
</tr>
<tr>
<td>6 VBN VBD</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>7 VBN VBD</td>
<td>Previous tag is NNP</td>
</tr>
<tr>
<td>8 VBD VBN</td>
<td>Previous tag is VBD</td>
</tr>
<tr>
<td>9 VBP VB</td>
<td>Previous tag is TO</td>
</tr>
<tr>
<td>10 POS VBZ</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>11 VB VBP</td>
<td>Previous tag is NNS</td>
</tr>
<tr>
<td>12 VBD VBN</td>
<td>One of the previous three tags is VBP</td>
</tr>
<tr>
<td>13 IN WDT</td>
<td>One of the next two tags is VB</td>
</tr>
<tr>
<td>14 VBD VBN</td>
<td>One of the previous two tags is VB</td>
</tr>
<tr>
<td>15 VB VBP</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>16 IN WDT</td>
<td>Next tag is VBZ</td>
</tr>
<tr>
<td>17 IN DT</td>
<td>Next tag is NN</td>
</tr>
<tr>
<td>18 JJ NNP</td>
<td>Next tag is NNP</td>
</tr>
<tr>
<td>19 IN WDT</td>
<td>Next tag is VBD</td>
</tr>
<tr>
<td>20 JJR RBR</td>
<td>Next tag is JJ</td>
</tr>
</tbody>
</table>

### EngCG Tagger

- **English constraint grammar tagger**
  - [Tapanainen and Voutilainen 94]
  - Something else you should know about
  - Hand-written and knowledge driven
  - “Don’t guess if you know” (general point about modeling more structure!)
  - Tag set doesn’t make all of the hard distinctions as the standard tag set (e.g. JJ/NN)
  - They get stellar accuracies: 98.5% on their tag set
  - Linguistic representation matters…
  - … but it’s easier to win when you make up the rules
CRF Taggers

- Newer, higher-powered discriminative sequence models
  - CRFs (also voted perceptrons, M3Ns)
  - Do not decompose training into independent local regions
  - Can be deathly slow to train – require repeated inference on training set
- Differences tend not to be too important for POS tagging
- However: one issue worth knowing about in local models
  - “Label bias” and other explaining away effects
  - Maxent taggers’ local scores can be near one without having both good “transitions” and “emissions”
  - This means that often evidence doesn’t flow properly
  - Why isn’t this a big deal for POS tagging?

Domain Effects

- Accuracies degrade outside of domain
  - Up to triple error rate
  - Usually make the most errors on the things you care about in the domain (e.g. protein names)
- Open questions
  - How to effectively exploit unlabeled data from a new domain (what could we gain?)
  - How to best incorporate domain lexica in a principled way (e.g. UMLS specialist lexicon, ontologies)