POS Tagging

POS Tagging = Assigning word class information to words

ex: the man bought a book

determiner noun verb determiner noun

Linguistic Questions

▶ How do we divide the text into individual word tokens?
▶ How do we choose a tagset to represent all words?
▶ How do we select appropriate tags for individual words?

Tagsets

Size of tagsets

▶ English:
  - TOSCA 32
  - Penn treebank 36
  - BNC C5 61
  - Brown 77
  - LOB 132
  - London-Lund Corpus 197
  - TOSCA-ICE 270
▶ Romanian: 614
▶ Hungarian: ca. 2 100

Annotating POS Tags

Two fundamentally different approaches:

▶ Start from scratch, find characteristics in words or context (= rules) which give indication of word class
  ▶ e.g., if word ends in "ion", tag it as noun
▶ Accumulate lexicon, disambiguate words with more than one tag
  ▶ e.g., possible categories for "about": preposition, adverb, particle
Automatic POS Tagging

Assumption: local context is sufficient

Examples:

- for the man: noun or verb?
- we will man: noun or verb?
- I can put: verb base form or past?
- re-cap real quick: adjective or adverb?

Bigram Tagging

- Basic assumption: POS tag only depends on word itself and on the POS tag of the previous word
- Use lexicon to retrieve ambiguity class for words
  - e.g., word: beginning, ambiguity class: [JJ, NN, VBG]
  - For unknown words: use heuristics, e.g. all open class POS tags
- Disambiguation: look for most likely path through possibilities

Bigram Tagging – Example

time flies like an arrow

S NN VBZ IN DT NN E

VB NNS VB JJ RB

Bigram Tagging – Probability Table

Probabilities (in %s from 0 to 100):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P(time</td>
<td>7.0727</td>
<td>0.0005</td>
<td>0.0003</td>
<td>0.4754</td>
<td>0.1610</td>
<td>2.512</td>
<td>0.0215</td>
<td>0.702</td>
</tr>
<tr>
<td>P(time</td>
<td>0.0005</td>
<td>0.5294</td>
<td>0.0566</td>
<td>0.0003</td>
<td>0.0566</td>
<td>0.0003</td>
<td>0.0215</td>
<td>0.0005</td>
</tr>
<tr>
<td>P(fliesVBZ</td>
<td>0.4754</td>
<td>0.3905</td>
<td>0.153</td>
<td>0.512</td>
<td>0.0003</td>
<td>0.0566</td>
<td>0.0215</td>
<td>0.702</td>
</tr>
<tr>
<td>P(fliesNN</td>
<td>0.1610</td>
<td>0.0566</td>
<td>0.0003</td>
<td>0.0566</td>
<td>0.0003</td>
<td>0.0566</td>
<td>0.0215</td>
<td>0.702</td>
</tr>
<tr>
<td>P(jj)</td>
<td>2.512</td>
<td>0.0003</td>
<td>0.0566</td>
<td>0.0566</td>
<td>0.0003</td>
<td>0.0566</td>
<td>0.0215</td>
<td>0.702</td>
</tr>
<tr>
<td>P(rb)</td>
<td>0.0215</td>
<td>0.702</td>
<td>0.0003</td>
<td>0.0566</td>
<td>0.0003</td>
<td>0.0566</td>
<td>0.0215</td>
<td>0.702</td>
</tr>
<tr>
<td>P(pitimeNN</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
</tr>
<tr>
<td>P(pitineNN</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0003</td>
</tr>
<tr>
<td>P(arrowNN</td>
<td>0.0215</td>
<td>0.702</td>
<td>0.0003</td>
<td>0.0566</td>
<td>0.0003</td>
<td>0.0566</td>
<td>0.0215</td>
<td>0.702</td>
</tr>
</tbody>
</table>

Bigram Tagging – Probabilities

\[
P(t_1 \ldots t_5) = P(t_1 | S)P(w_1 | t_1)P(t_2 | t_1)P(w_2 | t_2) \ldots
\]

(Note: this is actually \(P(t_1 \ldots t_5 | w_1 \ldots w_5)\))

green = transition probabilities
blue = lexical probabilities

Bigram Tagging – Counter-Examples

- start before
  - start before the course or start before he is done
- real quick
  - re-cap real quick or a real quick lunch
- barely changed
  - he was barely changed or he barely changed his contents
- that beginning
  - that beginning part or that beginning frightened the students or with that beginning early, he was forced ...
**Maximum Likelihood Estimation**

Simplest way to calculate such probabilities from a corpus:

$$P_{\text{MLE}}(t_n | t_{n-1}) = \frac{C(t_n, t_{n-1})}{C(t_{n-1})}$$

$$P_{\text{MLE}}(w_n | t_n) = \frac{C(w_n, t_n)}{C(t_n)}$$

- Uses relative frequency
- Maximizes the probabilities of the corpus

**Motivating Hidden Markov Models**

Thinking back to Markov models: we are now given a sequence of words and want to find the POS tags

- The underlying event of POS tags can be thought of as generating the words in the sentence
- Each state in the Markov model can be a POS tag
  - We don’t know the correct state sequence (Hidden Markov Model (HMM))

This requires an additional emission matrix, linking words with POS tags (cf. $P(\text{arrow} | \text{NN})$)

**Example HMM**

Assume DET, N, and VB as hidden states, with this transition matrix (A):

<table>
<thead>
<tr>
<th></th>
<th>DET</th>
<th>N</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DET</td>
<td>0.01</td>
<td>0.89</td>
<td>0.10</td>
</tr>
<tr>
<td>N</td>
<td>0.30</td>
<td>0.20</td>
<td>0.50</td>
</tr>
<tr>
<td>VB</td>
<td>0.67</td>
<td>0.23</td>
<td>0.10</td>
</tr>
</tbody>
</table>

... emission matrix (B):

<table>
<thead>
<tr>
<th></th>
<th>DET</th>
<th>N</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DET</td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>N</td>
<td>0.0</td>
<td>0.8</td>
<td>0.0</td>
</tr>
<tr>
<td>VB</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

... and initial probability matrix ($\pi$):

<table>
<thead>
<tr>
<th></th>
<th>DET</th>
<th>N</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DET</td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VB</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Using Example HMM**

In order to generate words, we:

1. Choose tag/state from $\pi$
2. Choose emitted word from relevant row of B
3. Choose transition from relevant row of A
4. Repeat #2 & #3, until we hit a stopping point
   - keeping track of probabilities as we go along

We could generate all possibilities this way and find the most probable sequence

- Want a more efficient way of finding most probable sequence

**Example HMM**

Assume DET, N, and VB as hidden states, with this transition matrix (A):

<table>
<thead>
<tr>
<th></th>
<th>DET</th>
<th>N</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DET</td>
<td>0.01</td>
<td>0.89</td>
<td>0.10</td>
</tr>
<tr>
<td>N</td>
<td>0.30</td>
<td>0.20</td>
<td>0.50</td>
</tr>
<tr>
<td>VB</td>
<td>0.67</td>
<td>0.23</td>
<td>0.10</td>
</tr>
</tbody>
</table>

... emission matrix (B):

<table>
<thead>
<tr>
<th></th>
<th>DET</th>
<th>N</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DET</td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>N</td>
<td>0.0</td>
<td>0.8</td>
<td>0.0</td>
</tr>
<tr>
<td>VB</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

... and initial probability matrix ($\pi$):

<table>
<thead>
<tr>
<th></th>
<th>DET</th>
<th>N</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DET</td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VB</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Using Example HMM**

In order to generate words, we:

1. Choose tag/state from $\pi$
2. Choose emitted word from relevant row of B
3. Choose transition from relevant row of A
4. Repeat #2 & #3, until we hit a stopping point
   - keeping track of probabilities as we go along

We could generate all possibilities this way and find the most probable sequence

- Want a more efficient way of finding most probable sequence