What is part of speech tagging?

- Given a string:
  
  This is a simple sentence

- Identify parts of speech (syntactic categories):
  
  This/DET is/VB a/DET simple/ADJ sentence/NOUN

- First step towards syntactic analysis

- Illustrates use of hidden Markov models to label sequences

Other tagging tasks

Other problems can also be framed as tagging (sequence labelling):

- Case restoration: If we just get lowercased text, we may want to restore proper casing, e.g. the river Thames

- Named entity recognition: It may also be useful to find names of persons, organizations, etc. in the text, e.g. Barack Obama

- Information field segmentation: Given specific type of text (classified advert, bibliography entry), identify which words belong to which “fields” (price/size/#bedrooms, author/title/year)

- Prosodic marking: In speech synthesis, which words/syllables have stress/intonation changes, e.g. He’s going. vs He’s going?

Parts of Speech

- Open class words (or content words)
  
  - nouns, verbs, adjectives, adverbs
  - mostly content-bearing: they refer to objects, actions, and features in the world
  - open class, since there is no limit to what these words are, new ones are added all the time (email, website).

- Closed class words (or function words)
  
  - pronouns, determiners, prepositions, connectives, ...
  - there is a limited number of these
  - mostly functional: to tie the concepts of a sentence together
How many parts of speech?

• Both linguistic and practical considerations
• Corpus annotators decide. Distinguish between
  – proper nouns (names) and common nouns?
  – singular and plural nouns?
  – past and present tense verbs?
  – auxiliary and main verbs?
  – etc

English POS tag sets

Usually have 40-100 tags. For example,

• Brown corpus (87 tags)
  – One of the earliest large corpora collected for computational linguistics (1960s)
  – A balanced corpus: different genres (fiction, news, academic, editorial, etc)

• Penn Treebank corpus (45 tags)
  – First large corpus annotated with POS and full syntactic trees (1992)
  – Possibly the most-used corpus in NLP
  – Originally, just text from the Wall Street Journal (WSJ)

POS tags in other languages

• Morphologically rich languages often have compound morphosyntactic tags
  \[ \text{Noun+A3sg+P2sg+Nom} \] (J&M, p.196)

• Hundreds or thousands of possible combinations

• Predicting these requires more complex methods than what we will discuss (e.g., may combine an FST with a probabilistic disambiguation system)
Universal POS tags (Petrov et al., 2011)

- A move in the other direction
- Simplify the set of tags to lowest common denominator across languages
- Map existing annotations onto universal tags
  \{VB, VBD, VBG, VBN, VBP, VBZ, MD\} ⇒ VERB
- Allows interoperability of systems across languages
- Promoted by Google and others

Why is POS tagging hard?
The usual reasons!

- Ambiguity:
  glass of water/NOUN vs. water/VERB the plants
  lie/VERB down vs. tell a lie/NOUN
  wind/VERB down vs. a mighty wind/NOUN (homographs)

  How about time flies like an arrow?

- Sparse data:
  - Words we haven’t seen before (at all, or in this context)
  - Word-Tag pairs we haven’t seen before

Relevant knowledge for POS tagging

- The word itself
  - Some words may only be nouns, e.g. arrow
  - Some words are ambiguous, e.g. like, flies
  - Probabilities may help, if one tag is more likely than another

- Local context
  - two determiners rarely follow each other
  - two base form verbs rarely follow each other
  - determiner is almost always followed by adjective or noun
A probabilistic model for tagging

Let’s define a new generative process for sentences.

• To generate sentence of length $n$:

Let $t_0 = <s>

For $i = 1$ to $n$

Choose a tag conditioned on previous tag: $P(t_i|t_{i-1})$

Choose a word conditioned on its tag: $P(w_i|t_i)$

• So, model assumes:
  – Each tag depends only on previous tag: a bigram model over tags.
  – Words are conditionally independent given tags

Probabilistic finite-state machine

• One way to view the model: sentences are generated by walking through states in a graph. Each state represents a tag.

Probabilistic finite-state machine

• When passing through a state, emit a word.

Probabilistic finite-state machine

• Prob of emitting $w$ from state $s$ (emission probability):

$$P(w_i = w|t_i = s)$$

What can we do with this model?

• Simplest thing: if we know the parameters (tag transition and word emission probabilities), can compute the probability of a tagged sentence.

• Let $S = w_1 \ldots w_n$ be the sentence and $T = t_1 \ldots t_n$ be the corresponding tag sequence. Then

$$p(S, T) = \prod_{i=1}^{n} P(t_i|t_{i-1})P(w_i|t_i)$$
Example: computing joint prob. \( P(S, T) \)

What’s the probability of this tagged sentence?

This/DET is/VB a/DET simple/JJ sentence/NN

• First, add begin- and end-of-sentence \(<s>\) and \(</s>\). Then:

\[
p(S, T) = \prod_{i=1}^{n} P(t_i|t_{i-1})P(w_i|t_i)
= P(\text{DET}|<s>)P(\text{VB}|\text{DET})P(\text{DET}|\text{VB})P(\text{JJ}|\text{DET})P(\text{NN}|\text{JJ})P(</s>|\text{NN})
\cdot P(\text{This}|\text{DET})P(\text{is}|\text{VB})P(\text{a}|\text{DET})P(\text{simple}|\text{JJ})P(\text{sentence}|\text{NN})
\]

• But now we need to plug in probabilities... from where?

Training the model

Given a corpus annotated with tags (e.g., Penn Treebank), we estimate \( P(w_i|t_i) \) and \( P(t_i|t_{i-1}) \) using familiar methods (MLE/smoothing)

![Figure 8.5](Fig from J&M draft 3rd edition)
Training the model

Given a corpus annotated with tags (e.g., Penn Treebank), we estimate $P(w_i|t_i)$ and $P(t_i|t_{i-1})$ using familiar methods (MLE/smoothing).

<table>
<thead>
<tr>
<th></th>
<th>Janet</th>
<th>will</th>
<th>back</th>
<th>the</th>
<th>bill</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNP</td>
<td>0.000032</td>
<td>0</td>
<td>0</td>
<td>0.000048</td>
<td>0</td>
</tr>
<tr>
<td>MD</td>
<td>0</td>
<td>0.308431</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VB</td>
<td>0</td>
<td>0.000028</td>
<td>0.000672</td>
<td>0</td>
<td>0.000028</td>
</tr>
<tr>
<td>JJ</td>
<td>0</td>
<td>0</td>
<td>0.000340</td>
<td>0.000097</td>
<td>0</td>
</tr>
<tr>
<td>NN</td>
<td>0</td>
<td>0.000200</td>
<td>0.000223</td>
<td>0.00006</td>
<td>0.002337</td>
</tr>
<tr>
<td>RB</td>
<td>0</td>
<td>0</td>
<td>0.010446</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.506099</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 8.6 Observation likelihoods $B$ computed from the WSJ corpus without smoothing. (Fig from J&M draft 3rd edition)

Hidden Markov Model (HMM)

HMM is actually a very general model for sequences. Elements of an HMM:

- a set of states (here: the tags)
- an output alphabet (here: words)
- initial state (here: beginning of sentence)
- state transition probabilities (here: $p(t_i|t_{i-1})$)
- symbol emission probabilities (here: $p(w_i|t_i)$)

But... tagging?

Normally, we want to use the model to find the best tag sequence for an untagged sentence.

- Thus, the name of the model: hidden Markov model
  - Markov: because of Markov assumption (tag/state only depends on immediately previous tag/state).
  - hidden: because we only observe the words/emissions; the tags/states are hidden (or latent) variables.
- FSM view: given a sequence of words, what is the most probable state path that generated them?

Formalizing the tagging problem

Normally, we want to use the model to find the best tag sequence $T$ for an untagged sentence $S$:

$$\arg\max_T p(T|S)$$
Formalizing the tagging problem

Normally, we want to use the model to find the best tag sequence $T$ for an untagged sentence $S$:

$$\text{argmax}_T p(T|S)$$

• Bayes’ rule gives us:

$$p(T|S) = \frac{p(S|T)p(T)}{p(S)}$$

• We can drop $p(S)$ if we are only interested in $\text{argmax}_T$:

$$\text{argmax}_T p(T|S) = \text{argmax}_T p(S|T)p(T)$$

Decomposing the model

Now we need to compute $P(S|T)$ and $P(T)$ (actually, their product $P(S|T)P(T) = P(S,T)$).

• We already defined how!

• $P(T)$ is the state transition sequence:

$$P(T) = \prod_i P(t_i|t_{i-1})$$

• $P(S|T)$ are the emission probabilities:

$$P(S|T) = \prod_i P(w_i|t_i)$$

Search for the best tag sequence

• We have defined a model, but how do we use it?

  – given: word sequence $S$
  – wanted: best tag sequence $T^*$

• For any specific tag sequence $T$, it is easy to compute $P(S,T) = P(S|T)P(T)$.

$$P(S|T)P(T) = \prod_i P(w_i|t_i) P(t_i|t_{i-1})$$

• So, can’t we just enumerate all possible $T$, compute their probabilities, and choose the best one?

Enumeration won’t work

• Suppose we have $c$ possible tags for each of the $n$ words in the sentence.

• How many possible tag sequences?
Enumeration won’t work

- Suppose we have \( c \) possible tags for each of the \( n \) words in the sentence.
- How many possible tag sequences?
- There are \( c^n \) possible tag sequences: the number grows **exponentially** in the length \( n \).
- For all but small \( n \), too many sequences to efficiently enumerate.

Finding the best path

- The **Viterbi algorithm** finds this path without explicitly enumerating all paths.
- Our second example of a **dynamic programming** (or **memoization**) algorithm.
- Like min. edit distance, the algorithm stores partial results in a **chart** to avoid recomputing them.

Recap: HMM

- Given a sentence \( O = o_1 \ldots o_T \) with tags \( Q = q_1 \ldots q_T \), compute \( P(O,Q) \) as:

\[
P(O,Q) = \prod_{t=1}^{T} P(o_t|q_t)P(q_t|q_{t-1})
\]

- But we want to find \( \arg \max_Q P(Q|O) \) without enumerating all possible \( Q \)
  - Use Viterbi algorithm to store partial computations.

Tagging example

Words:
Possible tags: (ordered by frequency for each word)

<table>
<thead>
<tr>
<th>&lt;s&gt;</th>
<th>one</th>
<th>dog</th>
<th>bit</th>
<th>&lt;/s&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CD</td>
<td>NN</td>
<td>NN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>VB</td>
<td>VBD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tagging example

Words:
Possible tags: (ordered by frequency for each word)

<table>
<thead>
<tr>
<th></th>
<th>one</th>
<th>dog</th>
<th>bit</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;s&gt;</td>
<td>CD</td>
<td>NN</td>
<td>NN</td>
<td>&lt;/s&gt;</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>VB</td>
<td>VBD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Choosing the best tag for each word independently gives the wrong answer (<s> CD NN NN </s>).
- \( P(VB|\text{bit}) < P(\text{NN}|\text{bit}) \), but may yield a better sequence (<s> CD NN VB </s>)
  - because \( P(\text{VBD}|\text{NN}) \) and \( P(</s>|\text{VBD}) \) are high.

Viterbi: intuition

Words:
Possible tags: (ordered by frequency for each word)

<table>
<thead>
<tr>
<th></th>
<th>one</th>
<th>dog</th>
<th>bit</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;s&gt;</td>
<td>CD</td>
<td>NN</td>
<td>NN</td>
<td>&lt;/s&gt;</td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>VB</td>
<td>VBD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Suppose we have already computed
  a) The best tag sequence for <s> … bit that ends in NN.
  b) The best tag sequence for <s> … bit that ends in VBD.
- Then, the best full sequence would be either
  - sequence (a) extended to include </s>, or
  - sequence (b) extended to include </s>.

Viterbi: high-level picture

• Intuition: the best path of length \( t \) ending in state \( q \) must include the best path of length \( t-1 \) to the previous state. \( (t \) now a time step, not a tag).
Summary

- Parts of speech (syntactic categories) provide the beginning of syntactic analysis, categorizing words by their behaviour.

- Hidden Markov models are a probabilistic model for POS tagging (and other sequence labelling tasks)

- HMM defines the joint probability of (tags, words).

- To find the best tag sequence, use the Viterbi Algorithm (details next time).

Reminders/announcements

- Assignment 1 will hit the website tonight.

- Assignments are intended to be done by pairs of students. If you want to choose your own partner, use the Google spreadsheet (link on Piazza) to sign up by NOON TOMORROW. Otherwise we will assign you a partner.

References