Outline of today’s lecture

From last time

POS tagging overview

HMMs for POS tagging

Imperfect training data

State-of-the-art in POS tagging
Questions or comments about previous lecture?

Background on syntax and morphology:

- L95 notes (at least http://www.cl.cam.ac.uk/teaching/1516/L95/introling.pdf)
- Exercises for POS tagging (in materials)
- Bender, Emily M. 2013. Linguistic Fundamentals for Natural Language Processing: 100 Essentials from Morphology and Syntax. (more advanced, not just English)
Generative models

- NB is a generative model: we train a model of the joint distribution of observations and classes, $P(\vec{f}, c)$.
- Hence, for multinomial NB, this is equivalent to a unigram model.
- Contrast discriminative models, where we train the posterior distribution of the class given the observation $P(c | \vec{f})$.
- Also: discriminant functions — we just train a mapping from the observation to the class label without the probability.
POS tagging

They can fish.
  ▶ They_PNP can_VM0 fish_VVI ._PUN

Lower ranked:
  ▶ They_PNP can_VVB fish_NN2 ._PUN
  ▶ They_PNP can_VM0 fish_NN2 ._PUN no full parse

*tagset* (CLAWS 5) includes:

<table>
<thead>
<tr>
<th>NN1</th>
<th>singular noun</th>
<th>NN2</th>
<th>plural noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNP</td>
<td>personal pronoun</td>
<td>VM0</td>
<td>modal auxiliary verb</td>
</tr>
<tr>
<td>VVB</td>
<td>base form of verb</td>
<td>VVI</td>
<td>infinitive form of verb</td>
</tr>
</tbody>
</table>
POS lexicon fragment

- they  PNP
- can VM0 VVB VVI NN1
- fish NN1 NN2 VVB VVI

- Lexicon could be acquired from a dictionary/grammar.
- Possible tag sequences could also come from a grammar.
- For ML approach, we want to acquire probabilities of tags and tag sequences from data.
Why POS tag?

Not often considered as a task until early 1990s, but much easier and faster than full parsing:

- Preprocessing before parsing to reduce search space or for unknown words.
- Simple source of syntactic features for other tasks: e.g., named entity recognition (NER).

Sports Direct hit by slide in pound.

- Aiding investigation of language: lexicographers, corpus linguistics.
POS tagging problem specification

- which language? English? Turkish? Japanese?
- tagset?
- genre? newspaper headlines, chemistry texts
- errors in the data?
  He walked in into the room.
- Accuracy for rare words? rare uses of words?

Nearly all published work is on a limited range of standard datasets . . .
POS tagging as a ML problem

- Classification of items in a sequence.
- Almost always treated as supervised learning.
- Available training data is somewhat limited: human annotators require fairly extensive training, annotation guidelines are lengthy, but inter-annotator agreement can be good (especially compared to most semantic tasks).
- Decide on (approximate) model, learn probabilities (efficiently), apply model (efficiently).
Modelling POS tagging as a ML problem

- **HMM**: Hidden Markov Model — POS tags are hidden states.
- **transition** probabilities and **emission** probabilities.
- Standard POS tagging uses HMMs in a simplified way: probabilities taken from annotated corpora (supervised).
- HMMs can be used unsupervised, but performance for POS tagging isn’t good.
- Efficient application via Viterbi algorithm.
- Basic model must be augmented with **smoothing** and treatment of **unknown words**.
Assigning probabilities

Estimate the sequence of $n$ tags as the sequence with the maximum probability, given the sequence of $n$ words:

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n|w_1^n)$$

By Bayes theorem:

$$P(t_1^n|w_1^n) = \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

Tagging a particular sequence of words so $P(w_1^n)$ is constant:

$$\hat{t}_1^n = \arg\max_{t_1^n} P(w_1^n|t_1^n)P(t_1^n)$$
Approximations

Bigram assumption: probability of a tag sequence approximated by the product of the two-tag sequences:

\[ P(t_1^n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1}) \]

Probability of the word estimated on the basis of its own tag alone:

\[ P(w_1^n | t_1^n) \approx \prod_{i=1}^{n} P(w_i | t_i) \]

Hence:

\[ \hat{t}_1^n = \arg \max_{t_1^n} \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1}) \]
More details

- Maximise the overall tag sequence probability — use Viterbi dynamic programming (details in J+M).
- Actual systems use trigrams — smoothing and backoff are critical: insufficient data to use 4-grams etc.
- Unseen words.
- Preprocessing: what is a word? formulae etc
- Genre effects: e.g., tag for ‘I’.
Smoothing

- Training data is always incomplete: some tag sequences are possible but rare, words will not be seen with all their possible POS tags.
- One zero probability turns everything into zeros!
- In some cases, zero probabilities are correct: e.g., probably don’t ever want so to be tagged as a verb.
- But, in general, hold back some probability mass for unseen events.
Smoothing techniques

- Add-one smoothing: simple, often effective (e.g., Naive Bayes, we don’t have real probabilities anyway).
- POS tagging:
  - use backoff for tag sequences: trigram counts modified by bigram and unigram counts with appropriate parameter.
  - e.g., replace all infrequent words (e.g., count less than 5) with UNK.
  - But: rare tags for frequent words?
  - Lots of experimentation . . .
Estimating tags for unknown words

- Distribute the probabilities according to the frequency of open class tags.
- But morphology: e.g., word ending in ‘ing’ can’t be VVD.
- Additional features: incorporating into HMM is messy . . .
- Most language have much richer morphology than English.
Improvements to HMMs

- Speed/accuracy trade-off: e.g., ideally want to incorporate word sequence information:
  
  I have a bad cold . . .
  There is a large cold . . .

- Discriminative models better for proper treatment of additional features (but HMM-based TnT very effective in practice).

- Bidirectional: HMM maximizes over sequence, but fully bidirectional is better.

- Character based models: morphology, capitalization etc.

- Until recently, lots of feature engineering.
POS tagging with LSTMs

Paper by Plank et al (2016), in course readings (details on LSTMs in lecture 7 or 8):

- Different natural languages, different language families.
- LSTMs can make use of pre-trained embeddings (unsupervised).
- Performance is close to the likely ceiling, but still quite low on unseen items in some languages.
- Best LSTM variant clearly better than TnT (c 25% reduction in error rate), but TnT still better with very limited training data.

Question to think about again: why POS tag?
Before next time

- Try POS tagging some of the sentences from the examples handout (manually and with an online system: see L90 notes for some URLs).