Solution notes

Natural Language Processing 2005 – Paper 8 Question 14 (AAC)

This question is taken from lecture 3. Part a is basically bookwork. Part b requires that the student understand the probabilities — some intuition about relative frequencies and thought about the meaning of the tags is needed for a good answer. Part c refers to improvements mentioned in the notes, though not discussed in detail: applicability to the errors in (1) and (2) makes this part somewhat more complicated to answer.

(*a*) Describe how the probabilities of the tags are estimated in a basic stochastic POS tagger. [7 marks]

From the lecture notes, the relative probability of a sequence of tags T given a sequence of words W is:

$$P(T|W) = P(T)P(W|T)$$

Under the bigram assumption, $P(T)$ is approximated by $P(t_i|t_{i-1})$. $P(W|T)$ is approximated by $P(w_i|t_i)$. A POS tagger maximises the probabilities over a sequence of tagged words (generally using the Viterbi algorithm for efficiency).

The probabilities are estimated by frequencies in some hand-tagged training data. $P(w_i|t_i)$ is estimated by counting the tags on individual words in the data (i.e., finding the proportion of instances of tag t contributed by each word w). $P(t_i|t_{i-1})$ is estimated by collecting all sequences of tags of length two and calculating the proportion of cases of tag t in which it is followed by tag t'.

(*b*) Explain how the probability estimates from the training data could have resulted in the tagging errors seen in (1) and (2). [6 marks]

In (1), the relative probabilities are $P(\text{Turkey}|\text{NP0})$, $P(\text{NP0}|\text{sentence boundary})$ and $P(\text{VM0}|\text{NP0})$ versus $P(\text{Turkey}|\text{NN1})$, $P(\text{NN1}|\text{sentence boundary})$ and $P(\text{VM0}|\text{NN1})$ (if we assume that the contribution of the possibilities with other POS tags than VM0 for *will* is so small as to be negligible). The lexical probabilities from the training data will depend on its source (newspapers are likely to have more instances of Turkey as NP0 than recipe books). $P(\text{NN1}|\text{sentence boundary})$ is likely to be lower than $P(\text{NP0}|\text{sentence boundary})$ since normally singular nouns have to have a determiner, so if the lexical probabilities are relatively similar, this will cause the error. Case sensitivity is a potential complication which might be mentioned.

In (2), the relevant probabilities are $P(\text{hope}|\text{VVB})$, $P(\text{VVB}|\text{VHB})$ and $P(\text{CJT}|\text{VVB})$ versus $P(\text{hope}|\text{NN1})$, $P(\text{NN1}|\text{VHB})$ and $P(\text{CJT}|\text{NN1})$. In this case, $P(\text{VVB}|\text{VHB})$ should be 0, since the base form should never follow *have*. This zero probability could have been smoothed by the POS tagger. There may well be very few cases of *hope* as a noun in the training data. $P(\text{CJT}|\text{VVB})$ is likely to be higher than $P(\text{CJT}|\text{NN1})$.

(*c*) In what ways can better probability estimates be obtained to improve the accuracy of the basic POS tagger you described in (a)? For each improvement you mention, explain whether you might expect it to improve performance on examples (1) and (2). [7 marks]

1. More training data, training data more similar to the text to be tagged. Potentially this could help with both (1) and (2).
2. Trigrams rather than bigrams. No likely effect on (1) or (2): in (1) there is no tag before the sentence boundary and the trigram sequences are both plausible; in (2) the additional tags don't add any discrimination.
3. Mechanism for handling unknown words, sensitive to morphology (e.g., word ending in -*ed* vs one ending in *ing*). Not likely to be relevant to the example sentences.
4. Backoff: e.g., from trigram probabilities to bigram probabilities. Unlikely to affect performance on (1) and (2).
5. Smoothing of low probabilities. Could affect performance on either (1) or (2) since both errors are likely to have involved some low probabilities. Potentially gives worse performance on (2): i.e., we want VHB to VVB to have a probability of 0.

There are some other ideas which might be mentioned, such as the use of manually specified rules to block impossible transitions.