School of Computing and Information Systems
The University of Melbourne
COMP90042
NATURAL LANGUAGE PROCESSING (Semester 1, 2024)

Workshop exercises: Week 4

Discussion

1. What is a **POS tag**?
   (a) POS tag (by hand) the following sentence: `Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.` according to the Penn Treebank tags. (Note that some of the tags are somewhat obscure.)
   (b) What are some common approaches to POS tagging? What aspects of the data might allow us to predict POS tags systematically?
2. What are the assumptions that go into a **Hidden Markov Model**? What is the time complexity of the **Viterbi algorithm**? Is this practical?
   (a) How can an HMM be used for POS tagging a text? For the purposes of POS tagging:
      i. How can the initial state probabilities π be estimated?
      ii. How can the transition probabilities A be estimated?
      iii. How can the emission probabilities B be estimated?
   (b) Estimate π, A and B for POS tagging, based on the following corpus:

```
1. silver-JJ wheels-NNS turn-VBP
2. wheels-NNS turn-VBP right-JJ
3. right-JJ wheels-NNS turn-VBP
```

3. Consider using the following Hidden Markov Model to tag the sentence `silver wheels turn`:
   $\pi[\text{JJ}, \text{NNS}, \text{VBP}] = [0.3, 0.4, 0.3]$

A	JJ	NNS	VBP
JJ	0.4	0.5	0.1
NNS	0.1	0.4	0.5
VBP	0.4	0.5	0.1

B	silver	wheels	turn
JJ	0.8	0.1	0.1
NNS	0.3	0.4	0.3
VBP	0.1	0.3	0.6

   (a) Visualise the HMM as a graph.
   (b) Use the **Viterbi algorithm** to find the most likely tag for this sequence.

Programming

1. In the iPython notebook `05-pos-tagging`:
   - Why does the bigram tagger — when used without "backoff" — perform worse than the unigram tagger? Find some examples of tokens which are tagged differently by the two models; give evidence from the training corpus as to why they are tagged differently.

- The notebook demonstrates that it helps to use the unigram tagger as a back-off model for the bigram tagger. Why does that mesh with our intuition? Find some examples of tokens that are tagged incorrectly by the unigram model, but correctly by the (backed-off) bigram model.

2. In the iPython notebook `06-hmm`:
   - The Viterbi algorithm is implemented with loops. Try to implement Viterbi using recursion instead.
   - Can you see the difference between the speed of the Viterbi algorithm and the exhaustive search over the lattice? How much faster is Viterbi than exhaustive search on an example problem? (hint: *time* or *clock* functions from the *time* package can be useful)

Catch-up

- Revise the terms **noun**, **verb**, **adjective**, and **determiner**.
- What is **lemmatisation**? Why would it be easier if we knew in advance that a token was a noun (or verb, etc.)?
- Who left waffles on the Falkland Islands? Why?
- What is the difference between a **discriminative** and a **generative** model?

Get ahead

- NLTK has extensive support for POS tagging. Have a read through Chapter 5 of the NLTK book. (`http://www.nltk.org/book/ch05.html`)