1 Finite State Transducers

1. The difference between inflectional and derivational morphology, noting their relation with word class

Inflectional morphology is when endings are added to indicate grammatical form and function. For example, they may be used to indicate a grammatical function like plural. The word class is the same both before and after the ending is added. Derivational morphology adds suffixes, affixes or infixes to a stem to create a new word with new meaning. In derivational morphology, the class changes after morphemes are added.

2. What is the difference between a FSA and a FST?

An FSA is a graph representing a regular expression that recognizes strings that are part of the language. An FST is a finite state transducer and it both recognizes and generates output. It can be used to parse the input or to transform it into another expression.

3. Write a transducer or transducers for...

   a) Explicitly and concisely state how FSTs are composed if there are more than one FST.
   b) Do not mix initial, final state. Also do not abuse final state.
   c) Do not miss the explanation for arcs or definition symbols.
d) Be careful about the tricky pitfalls (e.g. A666 output for exercise 3.5).

4. Solution for consonant doubling.

Disclaimer:
The following solution assumes that the input is a morphological representation (e.g., "bat $\wedge$ ed", where $\wedge$ represents a morpheme boundary), and the output is a fully spelled word (e.g., "batted"). It also assumes the input is not empty. An arc notated with a sequence of characters indicates that a sequence of states recognizing and outputting those characters has been collapsed into one edge.

2 Naïve Bayes Classifier

The topic of the test set document is about Pittsburgh.

$$P(Pittsburgh) = 3/4$$

$$P(\neg Pittburgh) = 1/4$$
Use add-one smoothing to prevent zero probability.

\[
P(\text{“Pittsburgh”} \mid \text{Pittsburgh}) = (4 + 1)/(7 + 6) = 5/13
\]
\[
P(\text{“CMU”} \mid \text{Pittsburgh}) = (1 + 1)/(7 + 6) = 2/13
\]
\[
P(\text{“Pittsburgh”} \mid \neg \text{Pittsburgh}) = (1 + 1)/(3 + 6) = 2/9
\]
\[
P(\text{“CMU”} \mid \neg \text{Pittsburgh}) = (0 + 1)/(3 + 6) = 1/9
\]

Probability comparison:

\[
P(\text{Pittsburgh} \mid \text{“Pittsburgh Pittsburgh Pittsburgh CMU”}) = \frac{3/4 \times 5/13 \times 5/13 \times 5/13 \times 2/13}{5/13 \times 5/13 \times 5/13 \times 2/13} = 0.0066
\]
\[
P(\neg \text{Pittsburgh} \mid \text{“Pittsburgh Pittsburgh Pittsburgh CMU”}) = \frac{1/4 \times 2/9 \times 2/9 \times 2/9 \times 1/9}{2/9 \times 2/9 \times 2/9 \times 1/9} = 0.0003
\]

3 Noisy Channel Model

1. \(Y\) is sequence of the POS tags and \(X\) is sequence of words.

2. \(\arg \max_Y P(Y)P(X \mid Y)\)

3. Suppose we are translating form \(L_1\) language to \(L_2\) language. In this context, if we have \(Y \rightarrow \text{Channel} \rightarrow X\), \(Y\) is the word sequence of \(L_2\) language and \(X\) is the word sequence of \(L_1\) language. The prior would be the \(P(Y)\), which is the language model of \(L_2\). And the channel model is modeled by \(P(X \mid Y)\), which is the translation model/channel model. We try to maximize the probability of \(L_2\) language given \(L_1\) language.

4 \textit{N}-gram Models and Smoothing

1. With the sequence go-la-tu ti-bu-do da-ro-pi ti-bu-do da-ro-pi pa-bi-ku da-ro-pi pa-bi-ku:

\[ P(ku \mid bi) = \frac{\#\{bi \: ku\}}{\#\{bi\}} = \frac{2}{2} = 1.0 \]

\[ P(do \mid bu) = \frac{\#\{bu \: do\}}{\#\{bu\}} = \frac{2}{2} = 1.0 \]

\[ P(da \mid ku) = \frac{\#\{ku \: da\}}{\#\{ku\}} = \frac{1}{2} \]

\[ P(ti \mid pi) = \frac{\#\{pi \: ti\}}{\#\{pi\}} = \frac{1}{3} \]

(One of the instances of ku is followed by the STOP symbol.)

2. With add-one smoothing, we get:

\[ P(ku \mid bi) = \frac{\#\{bi \: ku\} + 1}{\#\{bi\} + |V|} = \frac{3}{2 + 12} = \frac{3}{14} \]

\[ P(da \mid ku) = \frac{\#\{ku \: da\} + 1}{\#\{ku\} + |V|} = \frac{2}{2 + 12} = \frac{1}{7} \]

(There are 12 unique syllable sounds. These estimates assume that we are not adding 1 to each \( P(\text{STOP} \mid s) \); if we are, then 13 should be added to the denominators instead of 12.)

3. With so little data, the pseudocounts (the 1’s we add to numerators and the \( |V| \) we add to denominators) overwhelm the actual data; all the probabilities end up being very close, even for the low-probability transitions. A simple solution is to multiply these numbers by a small constant factor (say, 0.02) so that the data still dominates the probabilities.

4. With three hours of speech, the babies would hear a representative sample of all the syllable transitions. At that point, smoothing would only make the model’s estimation of the probabilities less accurate.
5. The probability table has one entry $P(s_1, s_2)$ for every pair of syllables $s_1$ and $s_2$. There are 12 different syllables, so that gives 144 entries in the table. In addition, we have an entry for $P(s|\text{START})$ and $P(\text{STOP}|s)$ for each syllable $s$, which gives another 24 parameters. Thus, the complete bigram model has 168 parameters.

6. You could use a bigram model to assign word boundaries between each sequential pair of syllables $s_1$ and $s_2$ for which the transition probability $P(s_2|s_1)$ is low (below some threshold). For this to work, you would first need to have identified the phones in the word (the basic sound units), grouped them into phonemes (e.g., in English aspirated and unaspirated $p$'s are interchangeable), and grouped the phonemes into syllables.

5 POS-tagging and HMMs

This solution comes from Jurafsky and Martin, lightly edited.

The Church (1988) and HMM taggers will perform differently when, given two tags $t_1$ and $t_2$ and a word $w$,

$$P(t_1 | w) > P(t_2 | w)$$

but,

$$P(w | t_1) < P(w | t_2)$$

This happens, for example, with words like manufacturing which was associated with the following probabilities in a sample of text from the Wall Street Journal:

\[
\begin{align*}
P(\text{VBG} | \text{manufacturing}) &= 0.231 \\
P(\text{NN} | \text{manufacturing}) &= 0.769 \\
P(\text{manufacturing} | \text{VBG}) &= 0.004 \\
P(\text{manufacturing} | \text{NN}) &= 0.001
\end{align*}
\]

Thus, if we are looking at the words and we see manufacturing, we expect this word to receive the tag NN, not the tag VBG. But if we are looking at the tags, we expect manufacturing to be produced more often from a VBG state
than from an NN state. Given a word like this, we can construct situations
where either the Church (1988) tagger or the HMM tagger produces the wrong
result by building a simple transition table where all transitions are equally
likely, e.g.:

\[ P(\text{NN} | \langle s \rangle) = P(\text{VBG} | \langle s \rangle) = 0.5 \]

Then the HMM model will select the VBG label:

\[
P(\text{manufacturing} | \text{NN})P(\text{NN} | \langle s \rangle) = (0.001)(0.5) = 0.0005
\]
\[
P(\text{manufacturing} | \text{VBG})P(\text{VBG} | \langle s \rangle) = (0.004)(0.5) = 0.002
\]

while the Church (1988) tagger will select the NN label:

\[
P(\text{NN} | \text{manufacturing})P(\text{NN} | \langle s \rangle) = (0.769)(0.5) = 0.3845
\]
\[
P(\text{VBG} | \text{manufacturing})P(\text{VBG} | \langle s \rangle) = (0.231)(0.5) = 0.1155
\]

If we have a phrase like *Manufacturing plants are useful*, then the Church
(1988) tagger has the better answer, while if we have a phrase like *Manufacturing plants that are brightly colored is popular*, then the HMM tagger has
the better answer.

6 Solution: Syntax and Parsing

1. All the rules must be of the form \( A \rightarrow a \) or \( B \rightarrow CD \) where \( a \) is a
terminal and \( B, C, \) and \( D \) are non-terminals. This form, into which any
CFG can be translated without loss, is called Chomsky Normal Form.

2. The following rules can be inferred from the provided chart:

(a) \( S \rightarrow NP \ VP \)
(b) \( VP \rightarrow V \ NP \)
(c) \( VP \rightarrow VP \ PP \)
(d) \( NP \rightarrow Det \ N \)
(e) $PP \rightarrow P\ NP$

(f) $VP \rightarrow eats$

(g) $NP \rightarrow she$

(h) $Det \rightarrow a$

(i) $V \rightarrow eats$

(j) $P \rightarrow with$

(k) $N \rightarrow fish$

(l) $N \rightarrow fork$

Alternatively, this grammar could be represented as follows:

(a) $S \rightarrow NP\ VP$

(b) $VP \rightarrow V\ NP \mid VP\ PP \mid eats$

(c) $NP \rightarrow Det\ N$

(d) $PP \rightarrow P\ NP$

(e) $NP \rightarrow she$

(f) $Det \rightarrow a$

(g) $V \rightarrow eats$

(h) $P \rightarrow with$

(i) $N \rightarrow fish \mid fork$

3. In this chart (which is only one of four different types of charts that can be used to represent the operation of the CYK algorithm) the span of terminals dominated by a parent can be easily described informally: for every occupied cell above the bottom row of the chart (the row second from the bottom in the table), the span of dominated cells runs from the column in which the parent lies to the column where a diagonal running southeast from the parent to the bottom row of the chart. For other representation of CYK charts, this relationship must be described differently.
4. A parse tree of the sentence has the following structure:

```
S
  NP       VP
     |       |
    she    VP
          |       |
         VP   PP
           |     |     |
          V   P   N
           |     |     |     |
          eats Det N with Det N
            a    fish      a    fork
```

5. The CYK algorithm, when applied to parsing, is robust in the face of ambiguity but, in its classical incarnation, CYK only retains one parse. The typical way of keeping alternate parses is by adding a three dimensional table of “backpointers.”