INF4820 - Excercise 3a

Magnus Andersen

November 6, 2013

1 Building a Hidden Markov Model

1.1 Theory: Hidden Markov Models

a)

Since our 'corpus' is so small, we readily see that the initial probabilities are

$$\begin{aligned} P(D|\langle S\rangle) &= 0.5 \\ P(N|\langle S\rangle) &= 0.5 \\ P(V|\langle S\rangle) &= 0 \\ P(P|\langle S\rangle) &= 0 \end{aligned}$$

The emission probabilities for the word "flies", given each tag, are

$$\begin{aligned} P(\text{flies}|D) &= \frac{C(D,\text{flies})}{C(D)} = \frac{0}{2} = 0 \\ P(\text{flies}|N) &= \frac{C(N,\text{flies})}{C(N)} = \frac{1}{6} = 0.1666... \\ P(\text{flies}|V) &= \frac{C(V,\text{flies})}{C(V)} = \frac{1}{2} = 0.5 \\ P(\text{flies}|P) &= \frac{C(P,\text{flies})}{C(P)} = \frac{0}{2} = 0 \end{aligned}$$

b)

In a general n-gram HMM there will be tons of unobserved data. For example, the word "in" is missing from our training data, but one would assume it should still have a fairly high emission probability from the P tag. In other words, our training data is extremely sparse; in our particular corpus there is a near "inifinite" amount of unobserved words, each of which will have an out-of-the-box probability of zero. Also, the V and P tags/states have an initial probability of zero - not terribly realistic. Other problems may be missing tag bigrams and missing tag/word permutations.

Smoothing is a means to remedy this. Smoothing is essentially a way to ensure that zero count items do not disappear down into the zero-probability sinkhole. A simple example is Laplace smoothing, in which all counts get incremented by one. This will obviously eliminate zero-probability cases. Even though more sophisticated smoothing techniques exist, I assumed that the purpose of this subtask was to simply illustrate the concept - hence me drawing the line at Laplace.

2 Tagging with Viterbi

2.1 Theory: Viterbi

a)

If I've understood it correctly, the trellis is a *numberOfStates-numberOfObservations* sized matrix (I chose to omit $\langle S \rangle$ and $\langle /S \rangle$) where the (as the assignment text implicitly suggests, I think) j-th row indicates state j, and the i-th column indicates that we have seen the first O_1^i observations. As such, the trellis will look like something such as Table 1,

States	**Observations**		
	flies	like	fruit
$s = D$	$v_1(s_1 = D)$	$v_2(s_2 = D)$	$v_3(s_3 = D)$
$s = N$	$v_1(s_1 = N)$	$v_2(s_2 = N)$	$v_3(s_3 = N)$
$s = V$	$v_1(s_1 = V)$	$v_2(s_2 = V)$	$v_3(s_3 = V)$
$s = P$	$v_1(s_1 = P)$	$v_2(s_2 = P)$	$v_3(s_3 = P)$

Table 1: Trellis matrix for "flies like fruit"

with[1]

$$v_1(s_1 = x) = P(s_1 = x | \langle S \rangle) \cdot P(o_1 = \text{flies} | s_1 = x)$$

and[1]

$$v_i(s_i = x) = \max_{k=1}^{L} \left[v_{i-1}(k) \cdot P(x|k) \cdot P(o_i|x) \right], \, i \geq 2$$

and where $s_i = x \in (D, N, V, P)$, $L = 4$ (number of states), $o_i \in$ (flies, like, fruit) and where e.g. $P(x = D|k = 1)$ means the probability of transitioning from the state indexed by 1 to state D, i.e. $P(D|D)$. With this in mind, the first column becomes

$$\begin{aligned}
v_1(D) &= P(D|\langle S \rangle) \cdot P(\text{flies}|D) \\
&= \frac{C(\langle S \rangle, D)}{C(\langle S \rangle)} \cdot \frac{C(D, \text{flies})}{C(D)} \\
&= \frac{1}{2} \cdot \frac{0}{2} = 0 \\
v_1(N) &= P(N|\langle S \rangle) \cdot P(\text{flies}|N) \\
&= \frac{C(\langle S \rangle, N)}{C(\langle S \rangle)} \cdot \frac{C(N, \text{flies})}{C(N)} \\
&= \frac{1}{2} \cdot \frac{1}{6} = \frac{1}{12} \\
v_1(V) &= P(V|\langle S \rangle) \cdot P(\text{flies}|V) \\
&= \frac{C(\langle S \rangle, V)}{C(\langle S \rangle)} \cdot \frac{C(V, \text{flies})}{C(V)} \\
&= \frac{0}{2} \cdot \frac{1}{2} = 0 \\
v_1(P) &= P(P|\langle S \rangle) \cdot P(\text{flies}|P) \\
&= \frac{C(\langle S \rangle, P)}{C(\langle S \rangle)} \cdot \frac{C(P, \text{flies})}{C(P)} \\
&= \frac{0}{2} \cdot \frac{0}{2} = 0
\end{aligned}$$

[1]http://www.uio.no/studier/emner/matnat/ifi/INF4820/h13/undervisningsmateriale/09a_dynamic_handout.pdf

The first element of the second column becomes

$$\begin{aligned}
v_2(D) &= \max_{k=1}^{4} [v_1(k) \cdot P(D|k) \cdot P(\text{like}|D)] \\
&= \max \left[0 \cdot ..., \frac{1}{12} \cdot \frac{C(N,D)}{C(N)} \cdot \frac{C(D,\text{like})}{C(D)}, 0 \cdot ..., 0 \cdot ...\right] \\
&= \max \left[0, \frac{1}{12} \cdot \frac{0}{6} \cdot \frac{0}{2}, 0, 0\right] \\
&= \max [0, 0, 0, 0] \\
&= 0
\end{aligned}$$

and so on.

b)

The main point one should make about the Viterbi algorithm is that it is dynamic: it solves a complex classification problem by breaking said problem down into smaller subproblems and recursively/iteratively solves those to arrive at a solution for the initial, much more complex problem.

The trellis is a matrix in which each cell contains the maximum probability that the i-th state is x, given that we have seen observations from o_1 and up to o_i. That is, if $v_1(x = D) = 0.7$, it means that the maximum probability of tagging o_1 with D is 0.7. If i.e. $v_1(N)$ then happens to be greater than $v_1(D)$, $v_1(N)$ would take precedence over $v_1(D)$, since this in turn means that o_1 is more likely to be N than D, etc.

The complexity of the Viterbi algorithm is $\mathcal{O}(2L^2N + 3L) = \mathcal{O}(L^2N)$ (since L^2 grows faster than L and 2 and 3 are constants), where L is the number of states and N is the length of the input observation sequence. This is because it loops through all of the observations in order, and for each current observation loops all the states, and for each state loops through a set of probabilities of size equal to the number of states (not the entire story, but simplifying $\mathcal{O}(2L^2N+3L)$ reflects this).

The naïve method enumerates all possible permutations of states for a given observation sequence, and computes the probability of each permutation. With L and N as before, this means a complexity of $\mathcal{O}(L^N)$. The naïve method will perform better only if there are only two states (i.e. Viterbi: $L^2 \cdot 2 = 2L^2$ vs näive: L^2). If there are three states (or more) Viterbi always wins (Viterbi: $L^2 \cdot 3 = 3L^2$ and näive: L^3). So Viterbi is obviously to be preferred. The way Viterbi improves upon the näive method is by straight from the get-go being interested only in the optimal path - it simply skips the evaluation of every non-optimal state sequence.