Sample Midterm: Natural Language Processing

(1) **Language Modeling**

Consider a language model over character sequences that computes the probability of a word based on the characters in that word, so if word $w = c_0, c_1, \ldots, c_n$ then $P(w) = P(c_0, \ldots, c_n)$. Let us assume that the language model is defined as a bigram character model $P(c_i \mid c_{i-1})$ where

$$P(c_0, \ldots, c_n) = \prod_{i=1,2,\ldots,n} P(c_i \mid c_{i-1})$$

Katz backoff smoothing is defined as follows:

$$P_{katz}(c_i \mid c_{i-1}) = \begin{cases} \frac{r^*(c_{i-1},c_i)}{r(c_{i-1})} & \text{if } r(c_{i-1}, c_i) > 0 \\ \alpha(c_{i-1}) P_{katz}(c_i) & \text{otherwise} \end{cases}$$

where $r(\cdot)$ provides the (unsmoothed) frequency from training data and $r^*(\cdot)$ is the Good-Turing estimate of the frequency r.

Provide the equation for $\alpha(c_{i-1})$ that ensures that $P_{katz}(c_i \mid c_{i-1})$ is a proper probability.

Answer:

Step by step derivation below. We are just looking for the end result.

$$\begin{aligned}
\sum_{c_i} \left(\frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})} + \alpha(c_{i-1}) P_{katz}(c_i) \right) &= 1 \\
\sum_{c_i : r(c_{i-1}, c_i) > 0} \frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})} + \alpha(c_{i-1}) \sum_{c_i : r(c_{i-1}, c_i) = 0} P_{katz}(c_i) &= 1 \\
\alpha(c_{i-1}) \sum_{c_i : r(c_{i-1}, c_i) = 0} P_{katz}(c_i) &= 1 - \left(\sum_{c_i : r(c_{i-1}, c_i) > 0} \frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})} \right) \\
\alpha(c_{i-1}) &= \frac{1 - \left(\sum_{c_i : r(c_{i-1}, c_i) > 0} \frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})} \right)}{\sum_{c_i : r(c_{i-1}, c_i) = 0} P_{katz}(c_i)} \\
\alpha(c_{i-1}) &= \frac{1 - \left(\sum_{c_i : r(c_{i-1}, c_i) > 0} \frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})} \right)}{1 - \left(\sum_{c_i : r(c_{i-1}, c_i) > 0} P_{katz}(c_i) \right)}
\end{aligned}$$

Also acceptable is the somewhat less precise answer which assumes $\sum_{c_i} P_{katz}(c_i) = 1$:

$$\alpha(c_{i-1}) = 1 - \sum_{c_i} \frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})}$$

(2) **Hidden Markov Models**:

The probability model $P(t_i \mid t_{i-2}, t_{i-1})$ is provided below where each t_i is a part of speech tag, e.g. the sixth row of the left table below corresponds to $P(\mathrm{D} \mid \mathrm{N}, \mathrm{V}) = \frac{1}{3}$. Also provided is $P(w_i \mid t_i)$ that a word w_i has a part of speech tag t_i, e.g. the seventh line of the right table below corresponds to $P(\text{flies} \mid \mathrm{V}) = \frac{1}{2}$.

$P(t_i \mid t_{i-2}, t_{i-1})$	t_{i-2}	t_{i-1}	t_i
1	`bos`	`bos`	N
$\frac{1}{2}$	`bos`	N	N
$\frac{1}{2}$	`bos`	N	V
$\frac{1}{2}$	N	N	V
$\frac{1}{2}$	N	N	P
$\frac{1}{3}$	N	V	D
$\frac{1}{3}$	N	V	V
$\frac{1}{3}$	N	V	P
1	V	D	N
1	V	V	D
1	N	P	D
1	V	P	D
1	P	D	N
1	D	N	`eos`

$P(w_i \mid t_i)$	t_i	w_i
1	D	an
$\frac{2}{5}$	N	time
$\frac{2}{5}$	N	arrow
$\frac{1}{5}$	N	flies
1	P	like
$\frac{1}{2}$	V	like
$\frac{1}{2}$	V	flies
1	`eos`	`eos`
1	`bos`	`bos`

The part of speech tag definitions are: `bos` (*begin sentence marker*), N (*noun*), V (*verb*), D (*determiner*), P (*preposition*), `eos` (*end of sentence marker*).

a. Provide a Hidden Markov Model (*hmm*) that uses the trigram part of speech probability $P(t_i \mid t_{i-2}, t_{i-1})$ as the transition probability $P_{hmm}(s_j \mid s_k)$ and the probability $P(w_i \mid t_i)$ as the emission probability $P_{hmm}(w_j \mid s_j)$.

Important: Provide the *hmm* in the form of two tables as shown below. The first table contains transitions between states in the *hmm* and the transition probabilities and the second table contains the words emitted at each state and the emission probabilities. Do not provide entries with zero probability.

from-state s_k	to-state s_j	$P(s_j \mid s_k)$

state s_j	emission w	$P(w \mid s_j)$

Hint: In your *hmm* the state $\langle N, \texttt{eos} \rangle$ will have emission of word `eos` with probability 1 and will not have transitions to any other states.

Answer: Here are the two tables that define the HMM, the transition table on the left and the emission table on the right:

from-state s_k	to-state s_j	$P(s_j \mid s_k)$
bos, bos	bos, N	$P(N \mid bos, bos)$ 1
bos, N	N, N	$P(N \mid bos, N)$ $\frac{1}{2}$
bos, N	N, V	$P(V \mid bos, N)$ $\frac{1}{2}$
N, N	N, V	$P(V \mid N, N)$ $\frac{1}{2}$
N, N	N, P	$P(P \mid N, N)$ $\frac{1}{2}$
N, V	V, D	$P(D \mid N, V)$ $\frac{1}{3}$
N, V	V, V	$P(V \mid N, V)$ $\frac{1}{3}$
N, V	V, P	$P(P \mid N, V)$ $\frac{1}{3}$
V, D	D, N	$P(N \mid V, D)$ 1
V, V	V, D	$P(D \mid V, V)$ 1
N, P	P, D	$P(D \mid N, P)$ 1
V, P	P, D	$P(D \mid V, P)$ 1
P, D	D, N	$P(N \mid P, D)$ 1
D, N	N, eos	$P(eos \mid D, N)$ 1

state s_j	emission w	$P(w \mid s_j)$
bos, bos	bos	1
bos, N	time	$\frac{2}{5}$
bos, N	arrow	$\frac{2}{5}$
bos, N	flies	$\frac{1}{5}$
N, N	time	$\frac{2}{5}$
N, N	arrow	$\frac{2}{5}$
N, N	flies	$\frac{1}{5}$
N, V	like	$\frac{1}{2}$
N, V	flies	$\frac{1}{2}$
V, D	an	1
V, V	like	$\frac{1}{2}$
V, V	flies	$\frac{1}{2}$
N, P	like	1
V, P	like	1
P, D	an	1
D, N	time	$\frac{2}{5}$
D, N	arrow	$\frac{2}{5}$
D, N	flies	$\frac{1}{5}$

b. Based on your *hmm* constructed in 2a. what is the state sequence with the highest probability for the following observation sequence:

`bos bos` time flies like an arrow `eos`

Answer:

bos	time	flies	like	an	arrow	eos	
(bos,bos) 1	(bos,N) $\times 1 \times \frac{2}{5}$	(N,V) $\times\frac{1}{2} \times \frac{1}{2}$	(V,P) $\times\frac{1}{3} \times 1$	(P,D) $\times 1 \times 1$	(D,N) $\times 1 \times \frac{2}{5}$	(N,eos) $\times 1 \times 1$	$= \frac{1}{75}*$
(bos,bos) 1	(bos,N) $\times 1 \times \frac{2}{5}$	(N,V) $\times\frac{1}{2} \times \frac{1}{2}$	(V,V) $\times\frac{1}{3} \times \frac{1}{2}$	(V,D) $\times 1 \times 1$	(D,N) $\times 1 \times \frac{2}{5}$	(N,eos) $\times 1 \times 1$	$= \frac{1}{150}$
(bos,bos) 1	(bos,N) $\times 1 \times \frac{2}{5}$	(N,N) $\times\frac{1}{2} \times \frac{1}{5}$	(N,P) $\times\frac{1}{2} \times 1$	(P,D) $\times 1 \times 1$	(D,N) $\times 1 \times \frac{2}{5}$	(N,eos) $\times 1 \times 1$	$= \frac{1}{125}$
(bos,bos) 1	(bos,N) $\times 1 \times \frac{2}{5}$	(N,N) $\times\frac{1}{2} \times \frac{1}{5}$	(N,V) $\times\frac{1}{2} \times \frac{1}{2}$	(V,D) $\times\frac{1}{3} \times 1$	(D,N) $\times 1 \times \frac{2}{5}$	(N,eos) $\times 1 \times 1$	$= \frac{1}{750}$

(3) Part-of-speech Tagging:

Consider the task of assigning the most likely part of speech tag to each word in an input sentence. We want to get the best (or most likely) tag sequence as defined by the equation:

$$T^* = \arg\max_{t_0,\ldots,t_n} P(t_0, \ldots, t_n \mid w_0, \ldots, w_n)$$

a. Write down the equation for computing the probability $P(t_0, \ldots, t_n \mid w_0, \ldots, w_n)$ using Bayes Rule and a trigram probability model over part of speech tags.

Answer:

$$\begin{aligned} P(t_0, \ldots, t_n \mid w_0, \ldots, w_n) &= P(w_0, \ldots, w_n \mid t_0, \ldots, t_n) \times P(t_0, \ldots, t_n) \\ &= \left(\prod_{i=0}^{n} P(w_i \mid t_i)\right) \times \left(P(t_0) \times P(t_1 \mid t_0) \times \prod_{i=2}^{n} P(t_i \mid t_{i-2}, t_{i-1})\right) \end{aligned}$$

b. We realize that we can get better tagging accuracy if we can condition the current tag on the previous tag and the next tag, i.e. if we can use $P(t_i \mid t_{i-1}, t_{i+1})$. Thus, we define the best (or most likely) tag sequence as follows:

$$\begin{aligned} T^* &= \arg\max_{t_0,\ldots,t_n} P(t_0,\ldots,t_n \mid w_0,\ldots,w_n) \\ &\approx \arg\max_{t_0,\ldots,t_n} \prod_{i=0}^{n+1} P(w_i \mid t_i) \times P(t_i \mid t_{i-1}, t_{i+1}) \text{ where } t_{-1} = t_{n+1} = \text{none} \end{aligned}$$

Explain why the Viterbi algorithm cannot be directly used to find T^* for the above equation.

Answer: The standard way of using Viterbi would be to have states in the HMM as pairs of tags and then we store the best path upto each tag pair $\langle t_{i-1}, t_i \rangle$ for time step i and then we can efficiently compute the best score upto state t_{i+1} recursively as follows:

$$\text{Viterbi}[i+1, \langle t_{i+1}, t_i \rangle] = \max_{\langle t_{i-1}, t_i \rangle} (\text{Viterbi}[i, \langle t_{i-1}, t_i \rangle] \times P(w_{i+1} \mid t_{i+1}) \times P(t_{i+1} \mid \langle t_{i-1}, t_i \rangle))$$

For t_i we can only condition on the previous tag t_{i-1} in the Viterbi algorithm since we have entered the score for each possible t_{i-1} in the Viterbi matrix. Crucially Viterbi proceeds from left to right, and so it cannot condition on a tag t_{i+1} which occurs on the right (and is not still part of the Viterbi table). Running Viterbi from right to left also does not work, due to tag t_{i-1} which occurs on the left. As a result, we cannot simultaneously consider t_{i-1} and t_{i+1} for Viterbi computation for the trigram probability $P(t_i \mid t_{i-1}, t_{i+1})$.

c. BestScore is the score for the maximum probability tag sequence for a given input word sequence.

$$\text{BestScore} = \max_{t_0,\ldots,t_n} P(t_0,\ldots,t_n \mid w_0,\ldots,w_n)$$

It is a bit simpler to compute than Viterbi since it does not compute the best sequence of tags (no back pointer is required). For the standard trigram model $P(t_i \mid t_{i-2}, t_{i-1})$:

$$\text{BestScore} = \max_{t_0,\ldots,t_n} \prod_{i=0}^{n+1} P(w_i \mid t_i) \times P(t_i \mid t_{i-2}, t_{i-1})$$

Assuming that $t_{-1} = t_{-2} = t_{n+1} = \text{none}$, we can compute BestScore recursively from left to right as follows:

$$\begin{aligned} \text{BestScore}[i+1, t_{i+1}, t_i] &= \max_{t_{i-1}, t_i} (\ \text{BestScore}[i, t_{i-1}, t_i] \times P(w_{i+1} \mid t_{i+1}) \times P(t_{i+1} \mid t_{i-1}, t_i)\ ) \\ &\quad \text{for all } -1 \le i \le n \\ \text{BestScore} &= \max_{\langle t, \text{none} \rangle} \text{BestScore}[n+1, \langle t, \text{none} \rangle] \end{aligned}$$

This algorithm for computing BestScore is simply the recursive forward algorithm for HMMs but with the sum replaced by max.

Provide an algorithm in order to compute BestScore for the improved trigram model $P(t_i \mid t_{i-1}, t_{i+1})$:

$$\text{BestScore} = \max_{t_0,\ldots,t_n} \prod_{i=0}^{n+1} P(w_i \mid t_i) \times P(t_i \mid t_{i-1}, t_{i+1}) \text{ where } t_{-1} = t_{n+1} = \text{none}$$

As before assume that: $P(t_0 \mid t_{-1} = \text{none}, t_1) \approx P(t_0 \mid t_1)$ and $P(t_n \mid t_{n-1}, t_{n+1} = \text{none}) \approx P(t_n \mid t_{n-1})$

You can provide either pseudo code, a recursive definition of the algorithm, or a recurrence relation.

Hint: The first step would be to extend the recursive BestScore algorithm given above to read the input from right to left.

Answer: We can define a dynamic programming algorithm that can be used to efficiently compute the best sequence of tags for the given equation. The algorithm is shown below as a recursive function (each call to this function can be stored in a matrix to give a polynomial time algorithm). We assume the input is padded with none's so that we can get probabilities $P(t_0 \mid t_{-1} = \text{none}, t_1)$ and $P(t_n \mid t_{n-1}, t_{n+1} = \text{none})$.

```
function bestScore()
    return NotViterbi($n + 2$, none, none, none)

function NotViterbi($i, t_{i-2}, t_{i-1}, t_i$)
    if ($i - 1 == -1$)
        if ($t_{i-2}, t_{i-1}, t_i$ == none, none, none) return 1
        else return 0
    return $\max_{t_{i-1}}$ NotViterbi($i - 1, t_{i-3}, t_{i-2}, t_{i-1}$) $\times P(w_{i-1} \mid t_{i-1}) \times P(t_{i-1} \mid t_{i-2}, t_i)$
```