1 Q1 Models

1.1 Q1.1

log p(colorless green ideas sleep furiously END") = **-55.923**

1.2 Q1.2

p(colorless green ideas sleep furiously END") = 0
The bigrams ("colorless green", "green ideas", etc.) are not present in the brown corpus so $p(w_t|w_{t-1})$ becomes zero.

1.3 Q1.3

Modelling END is important for linguistic quality because some words never appear at the end of sentence like "the", "very", etc. So the probability of ending a sentence should be less when the previous word is "the", "very", etc. And further in translation for e.g. Hindi has order SOV and English has SVO. Let's take an example of a sentence in English "She loves chocolate" is translated to something like "vah chokalet se pyaar karata hai" where "pyaar" corresponds to "loves", "vah" to "she" and "chokalet" to "chocolate". So rescoring a translation would be better if we model "END".

1.4 Q1.4

When we don't model "END", let's compute the probability of sentence of length T

$$\sum_{w_1 \in V} \sum_{w_2 \in V} ... \sum_{w_T \in V} Pr(w_1, w_2, ..., w_T | w_0 = START)$$
$$= \sum_{w_1 \in V} Pr(w_1|START) \sum_{w_2 \in V} Pr(w_2|w_1) ... \sum_{w_T \in V} Pr(w_T|w_{T-1}) = 1$$

As we see above the Pr(length = T) = 1 so Pr(any length) ¿ 1 so it's not a probabilty distribution. But when we model END as shown below we would get Pr(sentence of over all length) = 1. Hence, modelling END gives a well-formed probability distribution.

$$\sum_{w_1 \in V} \sum_{w_2 \in V} ... \sum_{w_T \in V} Pr(w_1, w_2, ..., w_T, w_{T+1} = END | w_0 = START)$$
$$= \sum_{w_1 \in V} Pr(w_1|START) \sum_{w_2 \in V} Pr(w_2|w_1) ... \sum_{w_T \in V} Pr(w_T|w_{T-1}) Pr(END|w_T) < 1$$

2 Q2

2.1 Q2.1

The LM parameters (V+1) for unigram language model are the probabilty of each vocabulary word including END. To find the Pr(len = n) we need only Pr(END) because we stop drawing next word when we encounter Pr(END) and rest is just 1 - Pr(END). So to get length = n we need to draw anything but END for the first n-1 times.

$$Pr(notEND) = 1 - Pr(END)$$
$$Pr(len = n) = (1 - Pr(END))^n Pr(END)$$

2.2 Q2.2

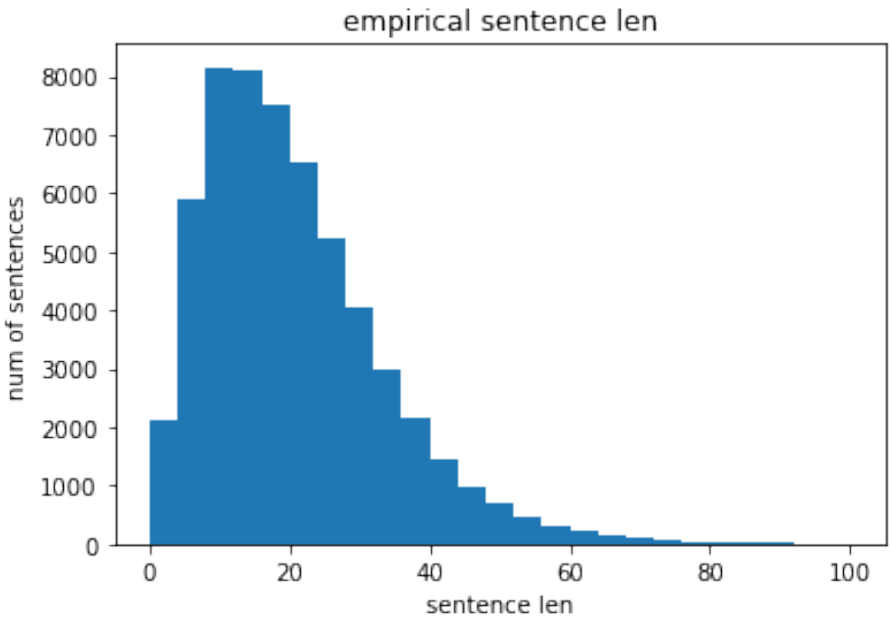

Figure 1: Dataset Sentence Length Histogram

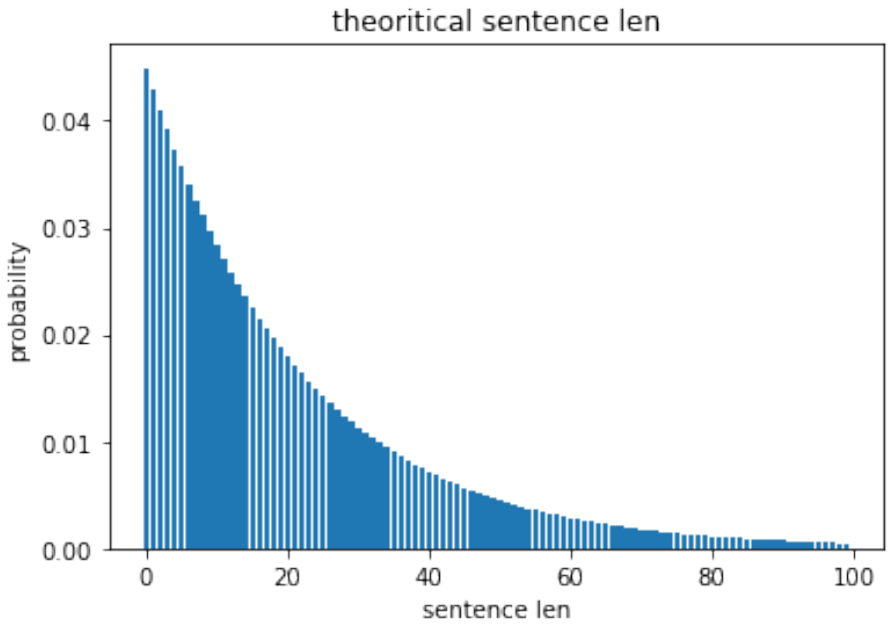

Figure 2: Theoritical Sentence Length Histogram

These distributions are different one is almost guassian like and other is geometric strictly decreasing. The reason is that unigram model doesn't take into account any contextual history.

3 Q3

3.1 Q3.1

The parameters in the Saul and Pereira's "aggregate bigram" model are $P(z|w_{t-1})$ and $P(w_t|z)$ each having dimensions v*k so we have **2*v*k** total parameters (or v*(k-1) + k*(v-1) because of multinomial parameters). Let's call them transition $P(z|w_{t-1})$ and emission $P(w_t|z)$ probabilities. Transition probabilities mean what is the class that it belongs to. Emission probabilities give us the distribution over words given a class.

3.2 Q3.2

The basic equation is: $Pr(w_t|w_{t-1}) = \sum_{z=1}^{z=K} Pr(w_t|z)Pr(z|w_{t-1})$

Using markov assumption, log product is sum over each log, and the above equation we get:

$$\log p(w_1, w_2, ..., w_{T+1} = END|w_0 = START) = \sum_{t=0}^{t=T} \log p(w_{t+1}|w_t)$$

3.3 Q3.3

E-Step

```
for bigram in corpus.sentences():
    prev_word, next_word = bigram
    normalize_sum = 0
    for c in 0..K:
        posterior[c][bigram] = transition[prev_word][c]*emision[c][next_word]
        normalize_sum += posterior[c][bigram]
    posterior[bigram] /= normalize_sum
```

The E-step is calculating the expected probability distribution over latent classes given that prev_word was followed by next_word.

3.4 Q3.4

M-Step

```
for bigram in corpus.sentences():
    prev_word, next_word = bigram
    for c in 0..K:
        transition[prev_word][c] = bigram.frequency() * posterior[c][bigram]
        emision[c][next_word] = bigram.frequency() * posterior[c][bigram]
    for word in corpus.words():
        transition[word] /= sum(transition[word], axis=1)
```

```
for c in 0..K:
    emision[c] /= sum(emision[c], axis=1)
```

The M-step is calculates the probability distribution over latent classes given that prev_word i.e. the transition probabilities and then given a latent class distribution what is the distribution over the next_word i.e. emission probabilities.

4 Q4

4.1 Q4.1

The numerator in both the LM parameters calculates the following:

$$\sum_w N(w_{t-1}, w) * Pr(z|w_{t-1}, w)$$

$$\sum_w N(w, w_t) * Pr(z|w, w_t)$$

If we $\sum_z$ in above equations for a given word (i.e sum a row) we are summing over all the occurences of the word in a bigram which is simply all occurences of a word.

4.2 Q4.2

All the numbers were the same. Below is a quantitative way.

```
empirical = np.sum(np.abs(np.sum(cnt_category_bigram, axis=1)
actual =  self.dataset.bigram_freq))
assert empirical == actual
```

4.3 Q4.3

Yes they sum to 1. For transition probs by summing over latent classes we should get 1 for each word. For emission probs by summing over all words we should get 1 for each class. They sum to 1 because it's a probability distribution.

4.4 Q4.4

Yes, the marginal log-likelihood in our case average log-likelihood per token is increasing. So the model is learning sentences that are more likely from the dataset.

4.5 Q4.5

Let's assume worst case that there is nothing in the dataset that our model can learn.
Then $Pr(w_t|w_{t-1}) = \frac{1}{V}$, $\log Pr(w_t|w_{t-1}) = -\log(V) = -10.81$.
And the sum over all words in all sentences and the division by number of tokens cancel out.
So $ToKLL \geq -10.81$.

4.6 Q4.6

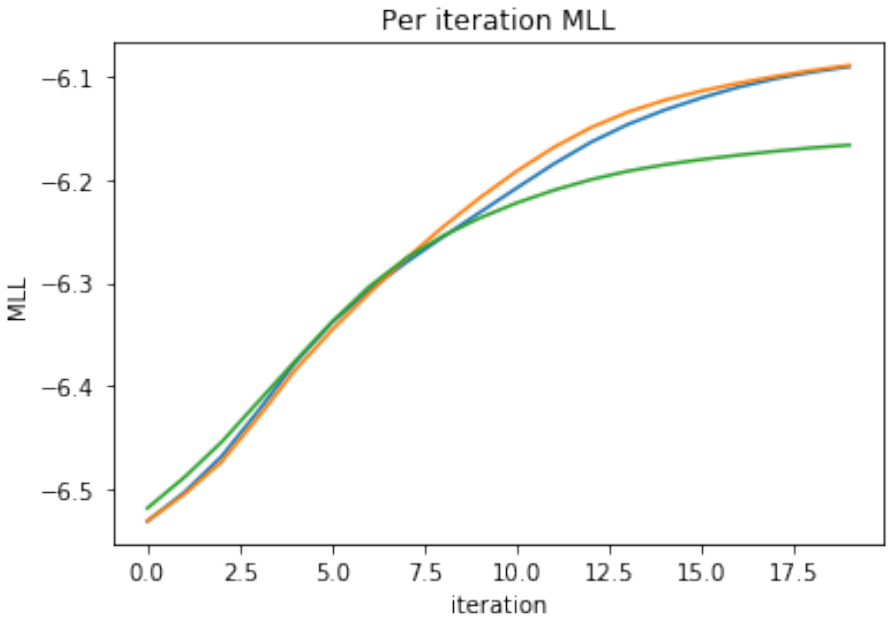

Figure 3: Per iteration MLL

4.7 Q4.7

$$\begin{aligned} J(Q,\theta) &= E_Q[\log P_\theta(w,z) - \log Q(Z)] \\ &= \sum_z P(z|w,\theta) \log P_\theta(w,z) - \sum_z P(z|w,\theta) \log P(z|w,\theta) \end{aligned}$$

If we observe what EM is doing. Then it expects the value of $P(z|w,\theta)$ and maximizes $P_\theta(w,z)$. Since $P(z|w,\theta)$ is common so it cancels out but EM maximizes the $\log P_\theta(w,z)$ term and Hence the overall objective increases.

5 Q5

5.1 Q5.1

The ratio of probabilities that I am getting is = 29475222.7841 [Note: I am including START and END] It is 100 times better in my model [Note: I am taking the best of 3].

5.2 Q5.2

Over classical bigram markov model the latent one doesn't have see a bigram to assign probability i.e. it can assign a probability to the next word from a given word even if this bigram was not there in the dataset. It can learn which class to assign to a word and based on that class predicts the next word. We can also see it as bottlnecking information.

5.3 Q5.3

Example 1:
"this thing ain't for free"
"free thing ain't for this"
Got **right**, probability ratio 10.96 is best and 1.34 is worst. [Note: Results on 3 randomly initialized models]

Example 2:
"society is the pillar of hope"
"pillar is the hope of society"
Got **right**, probability ratio 17.468 is best and 2.482 is worst. [Note: Results on 3 randomly initialized models]

Example 3:
"satisfaction is the death of desire"
"death is the satisfaction of desire"
Got **wrong**, probability ratio 0.792 is best and 0.425 is worst. [Note: Results on 3 randomly initialized models]. So it fools every model.

5.4 Q5.4

I created an example "satisfaction is the death of desire" and it's reordering "death is the satisfaction of desire" which fails the test. Changing the order of nouns while keeping the sentence structure fools the model. My hypothesis was that the model can learn noun subcategories (proper, common, abstract, countable, compound etc.) but in the above example demonstrates that the model doesn't know which noun is applicable to which. This model can't be improved for the above example in my understanding.

5.5 Q5.5

Yes! The model is quite sensitive to initialization. Random initialization can affect rare words (bigrams) more as they are updated depending on number of occurences. In "hope is the pillar of society" vs "pillar is the hope of society" the ratios are 4.67, 24.39 and 0.585. So there is lot of variance.

6 Q6

As was shown in Pereira (2000) and Pater's blog the restructuring of sentence which is grammatically incorrect fools such markov models. Markov models will assign low probabilty to long well-formed string than the short ill-formed string. Modelling language by focussing only on statistical information theory way won't lead to the final goal of NLP i.e. understanding language. These models cannot capture regularities and long term dependencies (global correlations) that is why after few words the grammar starts breaking even if model thinks it is generating highly probable sentences.

Since language is recursive structuring syntactic information into models via generative models that can learn their own rules for PCFG would lead to better grounding and understanding. And as Feyman famously said that "What I cannot create, I do not understand".