CS 533: Natural Language Processing **(Due: 02/04/20)**

Assignment 1

Name: Keya Desai, *NetID:* kd706 *Students discussed with:* Twisha Naik, Prakruti Joshi

Problem 1: Preliminaries ((1+1+1+1) + (1+1+1+1) + (1+1+1) = 11 points)

1. **(Probability)**

   (a) Given that, $Var(x) = E[(X - \mu_x)^2]$

$$\begin{aligned} \implies Var(x) &= E[X^2 + E(X)^2 - 2XE(X)] \\ &= E(X^2) + E(E(X)^2) - 2E(X)E(E(X)) \\ &= E(X^2) + E(X)^2 - 2E(X)^2 \\ &= E(X^2) - E(X)^2 \\ &= E(X^2) - \mu_x^2 \end{aligned} \tag{1}$$

   (b) E(X) = 3.5
   Var(X) = 2.91
   H(X) = 2.58

   (c) E(X) = 1
   Var(X) = 5
   H(X) = 0

(d) i) ∞
   ii)2.65
   iii)2.58

2. **(Linear Algebra)**

   (a) i) -3

   ii) $\begin{bmatrix} 13 & 5 & 3 & -2 \\ 28 & 14 & 9 & -8 \end{bmatrix}$

   iii) $\begin{bmatrix} -176 \\ 988 \\ 51 \\ 82 \\ 135 \end{bmatrix}$

   iv) Invalid

3. **(Optimization)**

   (a) Equation (1): x = 3
   Equation (2): x = $-\infty$

   (b) f'(x) = x - 3
   f"(x) = 1
   Since f"(x) >0, x at which first derivative = 0, gives the minimum value of f(x).
   $\implies$ x = 3

   (c) f'(x) = $(x - 3)^2$
   f'(x) = 0 $\implies$ x = 3.
   f"(x) = 2(x - 3)
   At x = 3, f"(x) = 0. Since f"(x) =0, the test fails. As x decreases, f(x) decreases further. Since x is a real number the minimum value it can take is $-\infty$.

Problem 2: n-Gram Models (4 + (2+1+1+1) = 9 points)

1. **(Relative Frequency Lemma)**

   (a) To solve the system:

$$\frac{\partial}{\partial \lambda} \sum_{i \in [n]} c_i log q_i - \lambda (1 - \sum_{i \in [n]} q_i) = 0 \tag{2}$$

$$\frac{\partial}{\partial q_j} \sum_{i \in [n]} c_i log q_i - \lambda (1 - \sum_{i \in [n]} q_i) = 0 \quad \forall j \in [n] \tag{3}$$

   Taking derivative of the (2), the first term gets cancelled. Second term:

$$-1 + \sum_{i \in [n]} q_i = 0 \implies \sum_{i \in [n]} q_i = 1 \tag{4}$$

   Taking derivative of the (3),

$$\frac{c_i}{q_i} + \lambda = 0 \implies c_i = -q_i \times \lambda \qquad \forall i \in [n] \tag{5}$$

   Summing over the terms in (5), we get

$$\sum_{i \in [n]} c_i = -\lambda \sum_{i \in [n]} q_i \tag{6}$$

   Using (4) and given that $\sum_{i \in [n]} c_i$ = N,

$$\lambda = -N \tag{7}$$

   Substituting value of (7) in (5),

$$q_i = \frac{c_i}{N} \qquad \forall i \in [n] \tag{8}$$

   Hence proving lemma 1.

2. **(Maximum Likelihood Estimation (MLE) of the Trigram Language Model)**

   (a) Estimate of MLE of language model is given by:

$$\hat{q}^{MLE} = argmax \frac{1}{N} \sum_{i=1}^{N} log q_X(x^{(i)}) \tag{9}$$

   The probabilities of a trigram language model is:

$$t(x_{1n}) = \prod_{j=1}^{M+1} t(x_j \mid x_{j-2}, x_{j-1}) \tag{10}$$

   Equation (9) is of the form defined in lemma 1 with:

$$\begin{aligned} c_i &= x^{(i)} \\ &= \#(x, x', x") \\ N &= \sum c_i \\ &= \sum_{x"} \#(x, x', x") \\ &= \#(x, x') \end{aligned}$$

(11)

Hence,

$$\begin{aligned} \hat{t}^{MLE} &= c_i/N \\ &= \frac{\#(x, x', x")}{\#(x, x')} \end{aligned} \tag{12}$$

(b) $\hat{t}^{MLE}$(the | BOS, BOS) = $\frac{3}{3} = 1$
$\hat{t}^{MLE}$(dog | BOS, the) = $\frac{1}{3}$
$\hat{t}^{MLE}$(ignored | the, dog) = $\frac{1}{1} = 1$
$\hat{t}^{MLE}$(the | dog, ignored) = $\frac{1}{1} = 1$
$\hat{t}^{MLE}$(cat | ignored,the) = $\frac{1}{1} = 1$
$\hat{t}^{MLE}$(EOS | the, cat) = $\frac{1}{2}$
$\hat{t}^{MLE}$(cat | BOS, the) = $\frac{1}{3}$
$\hat{t}^{MLE}$(ate | the, cat) = $\frac{1}{2}$
$\hat{t}^{MLE}$(the | cat, ate) = $\frac{1}{1} = 1$
$\hat{t}^{MLE}$(mouse | ate, the) = $\frac{1}{1} = 1$
$\hat{t}^{MLE}$(EOS | the, mouse) = $\frac{1}{2}$
$\hat{t}^{MLE}$(mouse | BOS, the) = $\frac{1}{3}$
$\hat{t}^{MLE}$(screamed | the, mouse) = $\frac{1}{2}$
$\hat{t}^{MLE}$(EOS | mouse, screamed) = $\frac{1}{1} = 1$

(c) 1.31

(d) ∞

Problem 3: Programming (2 + 1 + 1 + 1 + 2 + 1 + 3 + 1 = 12 points)

(Code must be submitted as well, with unambiguous commands for replicating reported results.)

1. The missing code in implementation of count_ngrams is:

   ngram = tuple(toks[(i-j):(i+1)])

   It passes the correctness test.

2. The vocabulary size for different tokenizers are as follows:

Tokenizer	**Vocab size**
basic	69148
nltk	41844
wp	15716
bpe	22581

3. From Figure 1, the Zipf's law seem to hold approximatel.

4. For the basic bigram model,
   Train perplexity: 70.967555
   Val Perplexity: inf
   There must be atleast one bigram in the Validation corpus, which is not present in the training corpus. Hence, the probability of that sentence occurring becomes 0, rendering the perplexity to ∞. (Similar to the example in Problem 2, 2nd part).

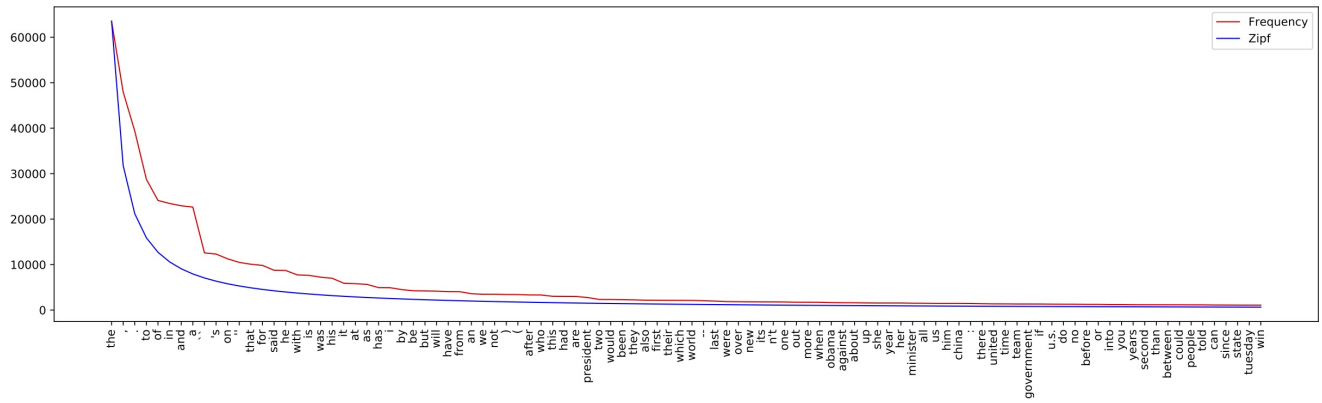

Figure 1: Top 100 most most frequent unigrams using nltk tokenizer

5. From 2, we observe that the training perplexity increases with increase in α with a huge jump from $\alpha = 1$ to $\alpha = 10$. The validation perplexity on the other hand improves with increase in α till $\alpha = 0.01$, then it again starts to increase. The lowest value is observed at $\alpha = 0.01$ with validation perplexity = 232.095.

   Suppose that there exists a bigram which exists in validation corpus but does not exist in the training corpus. This implies that #(x, x') will be equal to 0. But by implementing Laplace smoothing, the probability of that bigram will not be equal to zero because of the factor of α. Similarly, if #(x) was also zero, the probability will not be rendered ∞ because of the addition of $\alpha|V|$ term in the denominator. Hence, implementing Laplace smoothing fixes the issue of validation perplexity = ∞.

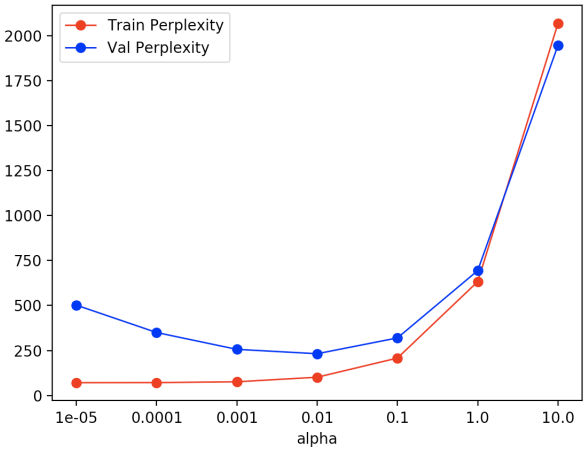

Figure 2: Training and validation Perplexity v/s alpha values in Laplace smoothing.

6. The training and validation perplexity by varying the fraction of training data is shown in Figure 3. As the training data increases, the validation perplexity should decrease as the model will be able to use more words and sentence and to calculate probabilities closer to the true estimate. This is exactly what we get, as seen in Figure 3.

7. The training and validation perplexity by varying value of β is shown in Figure 4. The training and validation perplexity decreases as the value of β increases. Minimum values of both perplexities are observed at $\beta = 0.9$. β is basically the weightage given to the bigrams in the model and 1- β is the weightage given to the unigrams. Unigrams completely ignores the context of the words while the bigram captures some context of a word w.r.t to its previous word. Hence the bigram model should perform better than the unigram model, which can be observed from the fact that giving more weight to the bigram probability improves the perplexity of training as well as validation.

8. After exploring all the choices, I found the best validation perplexity = 197.470875 by setting $\alpha = 0.01$ and $\beta = 0.8$.

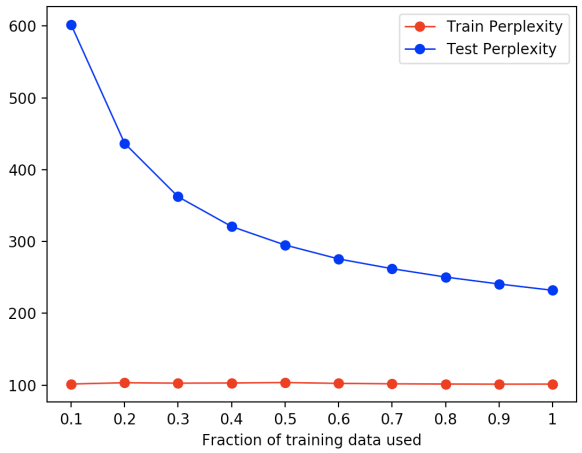

Figure 3: Training and validation perplexity vs variation in fraction of training data used.

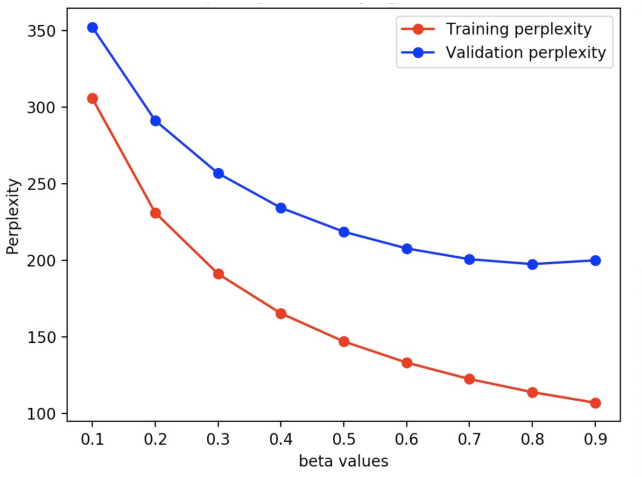

Figure 4: Training and validation Perplexity v/s beta values in Interpolation smoothing.