CS 533: Natural Language Processing **(Due: 01/31/20)**

Assignment 1

Name: Prakruti Joshi, *NetID:* phj15 *Students discussed with:* Twisha Naik, Keya Desai

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

1. **(Probability)**

   (a)

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

   (b) Mean = 3.5
   Variance = 2.92
   Entropy = 2.58

   (c) Mean = 6
   Variance = 0
   Entropy = 0

   (d) i. $+\infty$
   ii. 2.654
   iii. 2.585

2. **(Linear Algebra)**

   (a) $\begin{bmatrix} -3 \end{bmatrix}$

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

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

   (d) Invalid

3. **(Optimization)**

   (a) i. 3
   ii. -∞

(b)

$$f(x) = (x-3)^2/2 + 2$$

The first derivative of the function is:

$$f`(x) = (x-3)$$

The first derivative is 0 at x = 3. To verify the critical point is a maxima or minima, the second derivative of function is taken.

$$f``(x) = 1 > 0$$

Thus, $x = 3$ is a local minima.

(c)

$$f(x) = (x-3)^3/3 + 2$$

The first derivative of the function is:

$$f'(x) = (x-3)^2$$

The first derivative is 0 at x = 3. To verify the critical point is a maxima or minima, the second derivative of function is taken.

$$f``(x) = 2(x-3)$$

The second derivative is 0 at x=3, thus x=3 cannot be determined as maxima or minima. Since cube of negative value is the negative, minimizing (x-3) will minimize f(x). Since, x ∈ R, x can be as small as possible. Thus, the answer is x = -∞.

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

1. **(Relative Frequency Lemma)**

(a)

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

Taking the derivative, the first equation becomes:

$$(1 - \sum_{i \in [n]} q_i) = 0$$

(-1)

Taking the derivative, the second equation becomes:

$$c_i/q_i + \lambda = 0 \quad \forall i \in [n]$$
$$\implies \lambda * q_i = -(c_i) \quad \forall i \in [n]$$

(0)

Summing over all the values for i,

$$\implies \lambda = -N/1 \quad [Since \sum_{i \in [n]} q_i = 1 \, and \sum_{i \in [n]} c_i = N] \tag{1}$$

Solving the equations, we get (q, λ) as the solution as shown above. Also, λ comes out to be equal to -N, proving the Lemma 1 as:

$$q_i^* = c_i/N \tag{2}$$

using equation 0 derived above.

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

   (a) We define $[n] = V \cup EOS \cup BOS$ and x, x', x" $\in$ [n].
   If we define $C_{x"} = (x, x', x")$, a non-negative scalar associated with each x", then

   $$N = \sum_{x"} (x, x', x") = (x, x')$$

   which is the bigram counts using emperical MLE. Using Lemma 1, it can be shown that an emperical MLE estimate the trigram language model in eq 7 which maximxes the expected loglikelihood similar to eq 3. Thus, the MLE trigram language model is:

   $$\tilde{t}^{MLE}(x, x', x") = q^*_{x"} = C_{x"}/N = (x, x', x")/(x, x')$$

   (b) V = { the, dog, ignored, cat, ate, mouse, screamed } $\cup$ {EOS}

   All non-zero MLE parameter values estimated from the corpus V^+ :

   $\tilde{t}^{MLE}$ (the | BOS, BOS) = 3/3 = 1
   $\tilde{t}^{MLE}$ (dog | BOS, the) = 1/3
   $\tilde{t}^{MLE}$ (ignored | the, dog) = 1/1 = 1
   $\tilde{t}^{MLE}$ (the | dog, ignored) = 1/1 = 1
   $\tilde{t}^{MLE}$ (cat | ignored, the) = 1/1 = 1
   $\tilde{t}^{MLE}$ (EOS | the, cat) = 1/2

   $\tilde{t}^{MLE}$ (cat | BOS, the) = 1/3
   $\tilde{t}^{MLE}$ (ate | the, cat) = 1/2
   $\tilde{t}^{MLE}$ (the | cat, ate) = 1/1 = 1
   $\tilde{t}^{MLE}$ (mouse | ate, the) = 1/1 = 1
   $\tilde{t}^{MLE}$ (EOS | the, mouse) = 1/2

   $\tilde{t}^{MLE}$ (mouse | BOS, the) = 1/3
   $\tilde{t}^{MLE}$ (screamed | the, mouse) = 1/2
   $\tilde{t}^{MLE}$ (EOS | mouse, screamed) = 1/1 = 1

   (c) 1.32

   (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. Implemented *countngrams* in Tokenizer by setting ngram as:

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

   Verified the implementation using *testngramcounts*.

2. Vocabulary size for different tokenizers (using all training data and without thresholding):
   (a) nltk : 41844
   (b) basic: 69148
   (c) BertTokenizer (wp) : 15716
   (d) RobertaTokenizer (bpe): 22581

3. Yes. The Zipf's law does (approximtely) hold.

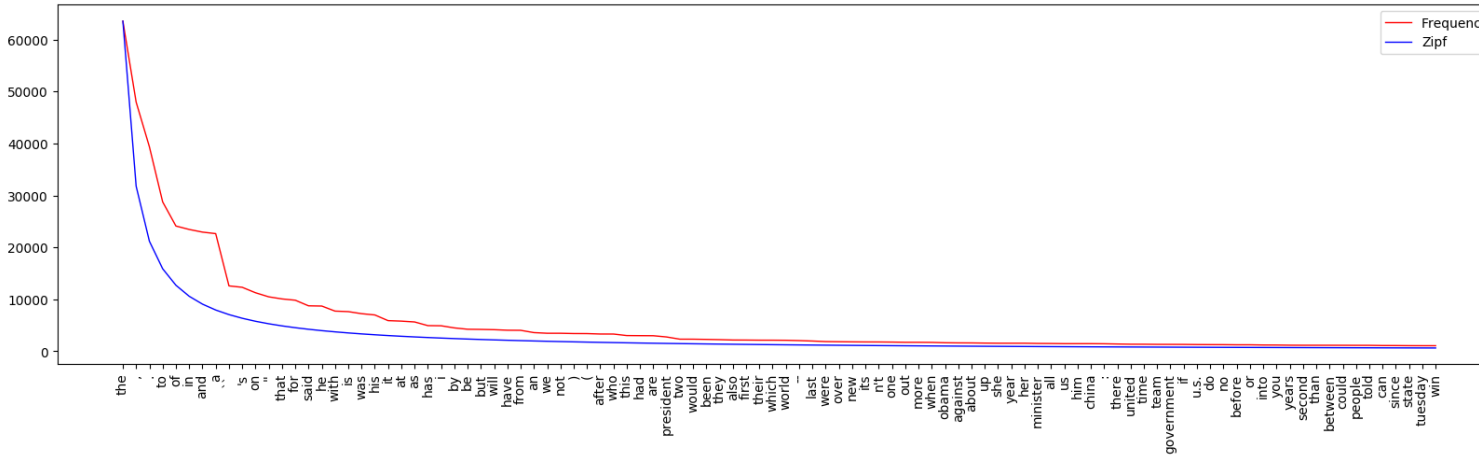

Figure 1: Top-100 most frequent unigrams using nltk tokenizer plotted against the Zipf distribution curve.

4. Training perplexity: 70.967
   Validation perplexity: ∞
   The validation set has bigram pairs which are not present in the training corpus. Without any smoothing, the bigram language model estimates the probabilities for such pairs as 0 based on the MLE counts. Thus, the perplexity for validation corpus becomes infinite for such bigram pairs as the probability is estimated as zero.

5. The plot of perplexity for different values of alpha is shown in Fig.2 The validation perplexity decreases and starts increasing after a local optima of alpha $= 10^{-2}$. This fixes the problem encountered in 4, as the bigram terms which were not encountered in the training corpus are non-zero due to the extra alpha parameter. Thus, the new terms in validation corpus have some non-zero probability. This ensures that the perplexity is finite.

6. The plot of perplexity for different training fractions of training data is shown in Fig.3 The validation perplexity should decrease as the training fraction is increased. This is because the model has more corpus to better estimate the probabilities. The training perplexity remains pretty much the same. The plot in Fig.3 matches our intuition as the validation perplexity decreases with the increase in training fraction.

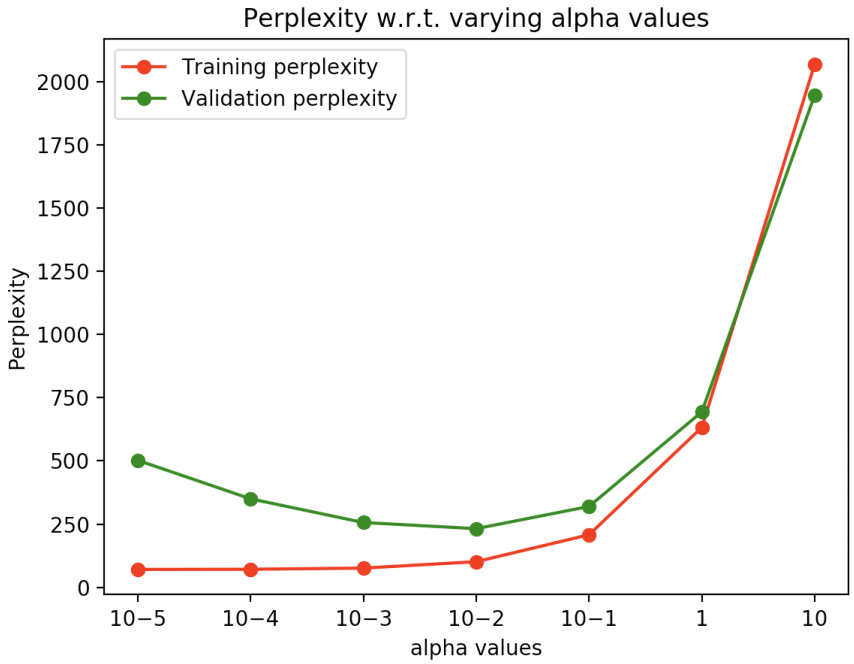

Figure 2: Perplexity for different values of alpha in Laplace smoothing

7. The plot of perplexity for different values of alpha is shown in Fig.3 The training and validation perplexity seems to decreases as the value of beta increases. The beta parameter adjusts the weight given to the bigram and unigram estimates. The optimal value for beta in this case turns out to be around 0.8.

8. The best validation perplexity : 197.471
   Parameters used:
   alpha = 0.01, beta = 0.8

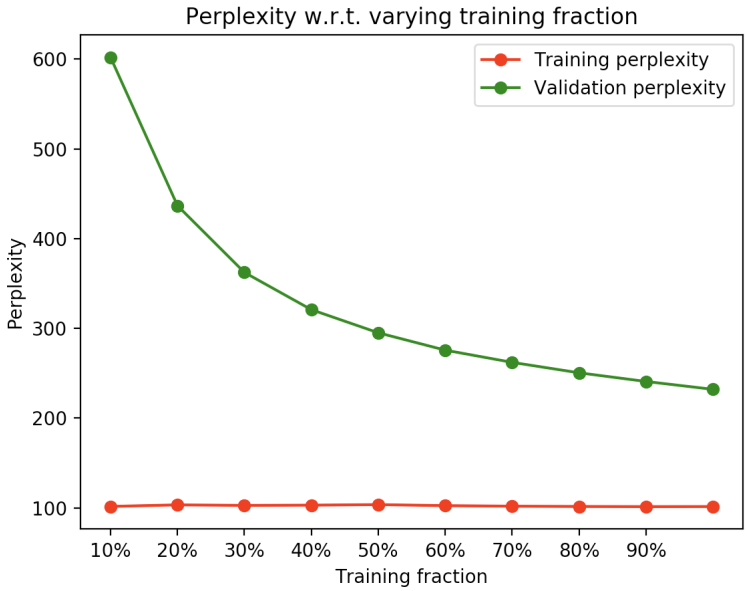

Figure 3: Perplexity for different training fraction

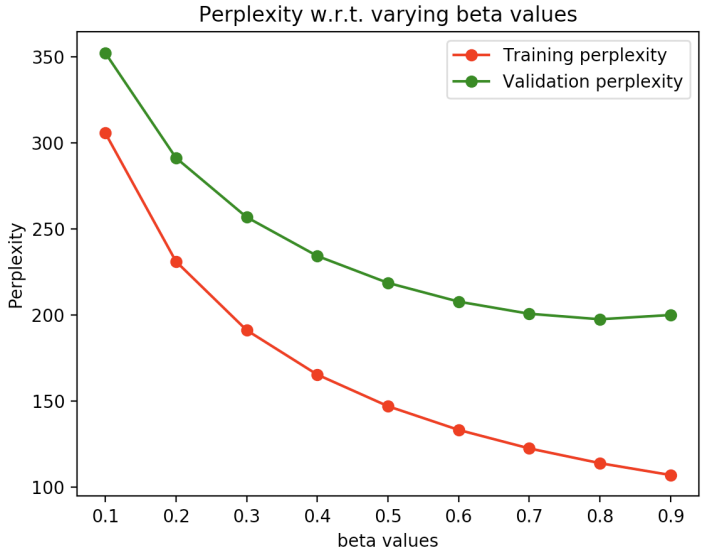

Figure 4: Perplexity for different beta fraction