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

Assignment 1

Name: Madhu Sivaraj, *NetID:* ms2407

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

1. **(Probability)**

   (a) The definition of variance is $Var[X] = E[(X - E[X])^2]$. I will use linearity of expectation to show that $Var[X] = E[X^2] - E[X]^2$. Let μ be $E[X]$.

   $Var[X] = E[(X - E[X])^2]$ - definition of variance

   $Var[X] = E[(X - \mu)^2]$ - replace $E[X]$ with μ

   $Var[X] = E[X^2 - 2\mu X + \mu^2]$

   $Var[X] = E[X^2] - 2\mu E[X] + \mu^2$ - linearity of expectation

   $Var[X] = E[X^2] - 2\mu^2 + \mu^2$

   $Var[X] = E[X^2] - \mu^2{}_{\text{x}}$

   $Var[X] = E[X^2] - E^2[X]$

   (b) Let X represent the outcome of a fair six-sided die (thus $X = \{1...6\}$).

   Mean(X) = $\frac{1+2+3+4+5+6}{6} = \frac{21}{6} = 3.5$

   Var(X) = $E[X^2] - \mu^2{}_{\text{x}} = 2.91$

   Entropy(X) = $-\sum_{i=1}^{6} \frac{1}{6} log_2 \frac{1}{6} = 2.58$

   (c) Suppose we make the die biased so that it always yields X = 6.

   Mean(X) = $\frac{6+6+6+6+6+6}{6} = \frac{36}{6} = 6$

   Var(X) = $E[X^2] - \mu^2{}_{\text{x}} = (1 * 6^2) - (1^2) = 0$

   Entropy(X) = $-\sum_{i=1}^{6} log_2 1 = 0$

   (d) $H(p_{\text{x}}, Cat(\frac{1}{2}, 0, 0, 0, \frac{1}{2}, 0)) = +\infty$

   H(p_{x}, Cat($\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{10}$)) = 2.65

   $H(p_{\text{x}}, Cat(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}))$ = 2.58

2. **(Linear Algebra)**

(a) $\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1\\4\end{bmatrix} = \begin{bmatrix}-3\end{bmatrix}$

(b) $\begin{bmatrix}1 & 4\\2 & 5\\3 & 6\end{bmatrix}^T \begin{bmatrix}0 & 1 & 1 & -1\\2 & 2 & 1 & -2\\3 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}13 & 5 & 3 & -2\\28 & 14 & 9 & -8\end{bmatrix}$

(c) $\begin{bmatrix}395 & -773 & 171 & 597 & 36\\777 & 888 & -999 & 100 & 1\\-18 & -2 & -35 & 53 & 99\\587 & 11 & 236 & 71 & 316\\391 & 7 & 35 & 128 & 6\end{bmatrix}^T \begin{bmatrix}0\\1\\0\\1\\0\end{bmatrix} = \begin{bmatrix}-176\\988\\51\\82\\135\end{bmatrix}$

(d) $\begin{bmatrix}1 & 4\\2 & 5\\3 & 6\end{bmatrix} \begin{bmatrix}0 & 1 & 1 & -1\\2 & 2 & 1 & -2\\3 & 0 & 0 & 1\end{bmatrix}^T = Invalid$

3. **(Optimization)**

(a) $\arg\min_{x\in\mathbb{R}} \frac{1}{2}(x-3)^2 + 2 = 2$ (where $X = 3$)

$\arg\min_{x\in\mathbb{R}} \frac{1}{3}(x-3)^3 + 2 = -\infty (where X is -\infty)$

(b) First Derivative of (1)
$\frac{du}{dx} = x - 3$

Second Derivative of (1)
$\frac{d^2u}{dx^2} = 1$

The local minimum is found at the point where the first derivative of a function is equal to 0 and the second derivative is positive. When X=3, the first derivative (X-3) equals 0 and the second derivative ($(x-3)^2$) is positive. Thus, the minimum is at the point (3,2) where X=3.

(c) First Derivative of (2)
$\frac{du}{dx} = (x-3)^2$

Second Derivative of (2)
$\frac{d^2u}{dx^2} = 2x - 6$

As mentioned above, the local minimum is found at the point where the first derivative of a function is equal to 0 and the second derivative is positive. Looking at this equation, we know this is a cubic function. Thus, all values less than 3, the inflection point, will be negative. So, we can conclude that the minimum will be at -∞.

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

1. **(Relative Frequency Lemma)**

(a) $\frac{\partial}{\partial\lambda}\sum_i^n c_i * logq_i - \lambda(1 - \sum_i^n q_i = 0$

When simplified: $\frac{c_i}{ln2*q_i} = 0$ so $\lambda = -\frac{n}{ln2}$

From the assignment, we know:
$\sum_i^n q_\text{i} = 1$ which is equal to $-\frac{n}{ln2*\lambda}$

By substituting lambda for $-\frac{n}{ln2}$, we get $-\frac{n}{ln2*-\frac{n}{ln2}}$ and end with +1, thus proving Lemma 1.

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

   (a) Use Lemma 1 to show that an empirical MLE (6) of the trigram language model.

   (b) (START, START, the) = $\frac{3}{3} = 1$

   (START, the, cat) = 13
   (START, the, dog) = $\frac{1}{3}$
   (START, the, mouse) = $\frac{1}{3}$
   (the, cat, STOP) = 1
   (the, mouse, STOP) = 1
   (mouse, screamed, STOP) = 1
   (mouse, STOP, STOP) = 1
   (cat, STOP, STOP) = 1
   (screamed, STOP, STOP) = 1
   (cat, ate, the) = 1
   (the, cat, ate) = 1
   (ate, the, mouse) = 1
   (cat, the, ignored) = 1
   (the, dog, ignored) = 1
   (dog, ignored, the) = 1
   (the, mouse, screamed) = 1

   (c) Perplexity = $e^{entropy} = e^{-ln(1/3)+2ln(1/3)} \approx 12 nats$

   (d) Perplexity = $e^{crossentropy} = e^{-((2/3)ln(1/3)+(2/3)ln(1/3)+2ln(1/3))} \approx 17 nats$

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. I implemented the count_ngrams function in Tokenizer. Can be found in util.py file.

```
def count_ngrams(self, toks, n=3):
      ngram_counts = [Counter() for _ in range(n)]
      for i in range(len(toks)):
          for j in range(n):
              if i - j >= 0:
                  ngram = tuple(toks[i-j:i+1])
                  ngram_counts[j][ngram] += 1
      return ngram_counts
```

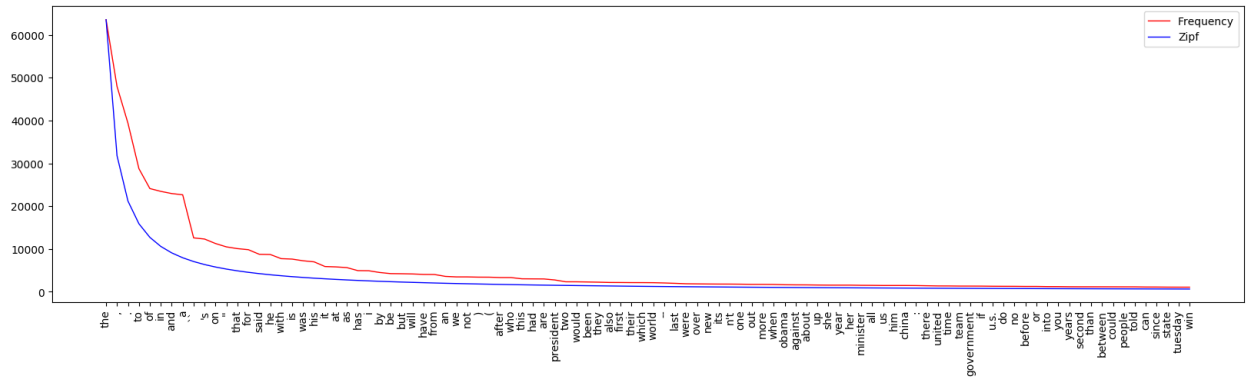
   I tested my function by running the **python3 assignment1.py** command, and my implementation was successful as I was able to pass the assert statements found in test_ngram_counts() function.

2. For each choice of tokenizer (basic, nltk, wp, bpe) , I found the vocabulary size, using all training data and without thresholding, by using the command **python3 assignment1.py –tok tokenizer-choice**. The vocabulary size for basic was 69148. The vocabulary size for nltk was 41844. The vocabulary size for wp was 15716. The vocabulary size for bpe was 22581.

   The following figures are screenshots of my outputs in Terminal which show the vocabulary size for each tokenizer choice.

```
(venv) madhu@Madhumithas-MacBook-Pro code % python3 assignment1.py --tok basic
--------------------------------------------------------------------------------
Using 1000009 tokens for training (100% of 1000009)
--------------------------------------------------------------------------------
Most frequent n-grams for n=1,2,3
(('the',), 62469)               (('of', 'the'), 6657)           (('the', 'united', 'states'), 538)
(('to',), 28616)                (('in', 'the'), 6644)           (('president', 'barack', 'obama'), 323)
(('of',), 24045)                (('to', 'the'), 2720)           (('one', 'of', 'the'), 320)
(('in',), 23200)                (('for', 'the'), 2332)          (('said', 'in', 'a'), 265)
(('and',), 22750)               (('on', 'the'), 2246)           (('the', 'end', 'of'), 264)
(('a',), 22421)                 (('at', 'the'), 2165)           (('in', 'the', 'first'), 225)
(('on',), 11120)                (('in', 'a'), 1944)             (('the', 'white', 'house'), 224)
(('for',), 9716)                (('and', 'the'), 1641)          (('out', 'of', 'the'), 208)
(('that',), 9349)               (('with', 'the'), 1390)         (('as', 'well', 'as'), 207)
(('he',), 7980)                 (('to', 'be'), 1378)            (('a', 'lot', 'of'), 204)
Saved plot at figure.pdf
--------------------------------------------------------------------------------
Using vocab size 10000 (excluding UNK) (original 69148)
(venv) madhu@Madhumithas-MacBook-Pro code % python3 assignment1.py --tok nltk
--------------------------------------------------------------------------------
Using 1137951 tokens for training (100% of 1137951)
--------------------------------------------------------------------------------
Most frequent n-grams for n=1,2,3
(('the',), 63536)               (('.', '``'), 7753)             (('.', '``', 'i'), 1452)
((',',), 47952)                 (('in', 'the'), 6706)           (('said', '.', '``'), 1149)
(('.',), 39355)                 (('of', 'the'), 6659)           ((',', "'", 'he'), 1004)
(('to',), 28752)                ((',', "'"), 4978)              (('.', '``', 'we'), 850)
(('of',), 24104)                (('.', 'the'), 4547)            (('the', 'united', 'states'), 769)
(('in',), 23446)                (('.', "'"), 3089)              ((',', "'", 'said'), 767)
(('and',), 22935)               ((',', 'the'), 2762)            (("'", 'he', 'said'), 720)
(('a',), 22643)                 (('to', 'the'), 2723)           (('.', '``', 'the'), 685)
(('``',), 12585)                (('said', '.'), 2373)           (('he', 'said', '.'), 644)
(("'s",), 12312)                (('for', 'the'), 2348)          (('.', '``', 'it'), 618)
Saved plot at figure.pdf
--------------------------------------------------------------------------------
Using vocab size 10000 (excluding UNK) (original 41844)
(venv) madhu@Madhumithas-MacBook-Pro code % python3 assignment1.py --tok wp
--------------------------------------------------------------------------------
Using 1452469 tokens for training (100% of 1452469)
--------------------------------------------------------------------------------
Most frequent n-grams for n=1,2,3
(('the',), 63869)               (("'", 's'), 12351)             (('o', '##ba', '##ma'), 1653)
(('.',), 50257)                 (('.', '"'), 10877)             (('.', '"', 'i'), 1480)
((',',), 48836)                 (('in', 'the'), 6708)           (('u', '.', 's'), 1478)
(('to',), 29281)                (('of', 'the'), 6659)           (('.', 's', '.'), 1474)
(('a',), 27305)                 ((',', '"'), 5348)              (('said', '.', '"'), 1152)
(('in',), 24570)                (('.', 'the'), 5075)            (('.', '"', 'the'), 1118)
(('of',), 24174)                ((',', 'the'), 2762)            (('t', '##ues', '##day'), 1094)
(('and',), 23385)               (('to', 'the'), 2726)           (('wed', '##nes', '##day'), 1057)
(('"',), 23061)                 (('said', '.'), 2530)           (('sat', '##ur', '##day'), 1036)
(('–',), 21850)                 (('in', 'a'), 2356)             ((',', '"', 'he'), 1023)
Saved plot at figure.pdf
--------------------------------------------------------------------------------
Using vocab size 10000 (excluding UNK) (original 15716)
(venv) madhu@Madhumithas-MacBook-Pro code % python3 assignment1.py --tok bpe
--------------------------------------------------------------------------------
Using 1357070 tokens for training (100% of 1357070)
--------------------------------------------------------------------------------
Most frequent n-grams for n=1,2,3
(('Ġthe',), 62482)              (('.', 'Ġ"'), 7209)             (('.', 'Ġ"', 'i'), 1440)
(('.',), 45027)                 (('Ġof', 'Ġthe'), 6657)         (('Ġu', '.', 's'), 1391)
((',',), 42880)                 (('Ġin', 'Ġthe'), 6644)         (('.', 's', '.'), 1353)
(('Ġto',), 28852)               (('.', 'Ġthe'), 4733)           (('Ġsaid', '.', 'Ġ"'), 1149)
(('Ġof',), 24072)               (('Ġto', 'Ġthe'), 2721)         (('.', 'Ġ"', 'we'), 840)
(('Ġin',), 23409)               ((',', 'Ġthe'), 2720)           ((',', 'Ġ6', '–'), 810)
(('Ġand',), 23185)              (('Ġsaid', '.'), 2529)          (('Ġthe', 'Ġunited', 'Ġstates'), 758)
(('Ġa',), 23144)                (('Ġfor', 'Ġthe'), 2332)        ((',"', 'Ġhe', 'Ġsaid'), 707)
(('–',), 17158)                 ((',', 'Ġand'), 2315)           (('Ġhe', 'Ġsaid', '.'), 691)
(("'s",), 12329)                (('Ġon', 'Ġthe'), 2246)         (('year', '–', 'old'), 683)
Saved plot at figure.pdf
--------------------------------------------------------------------------------
Using vocab size 10000 (excluding UNK) (original 22581)
```

3. The following figure presents a plot of the top-100 most frequent unigrams. This figure was created using the command **python3 assignment1.py –figure_file q3_top100.png**.



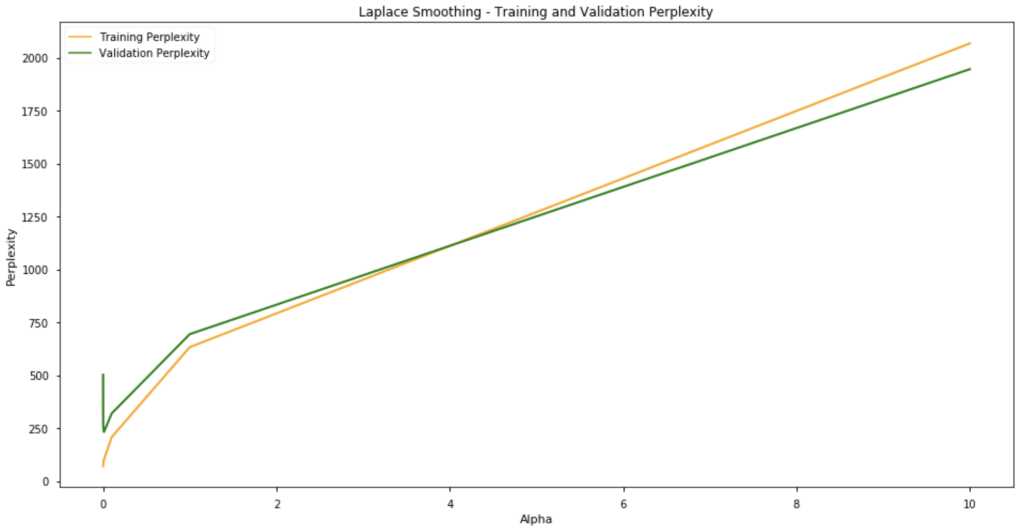
Zipf's law seems to hold based on my findings from this graph. Zipf's law states that given some corpus of natural language utterances, the frequency of any unigram is inversely proportional to its rank in the frequency table. Therefore, the most frequent unigram will occur approximately twice as often as the second most frequent unigram, three times as often as the third most frequent unigram, and onwards. Based on my figure, this holds true as the drop between the most frequent unigram, the, and the second unigram, ., is much greater than the drop from tuesday to win, which are the two least frequent unigrams in this set of the top-100 most frequent.

4. After training and testing the basic bigram model by running **assignment1.py**, I receive a training validation of 70.967555 and a validation perplexity of infinity. See the following figure for a screenshot of my terminal output containing these metrics.

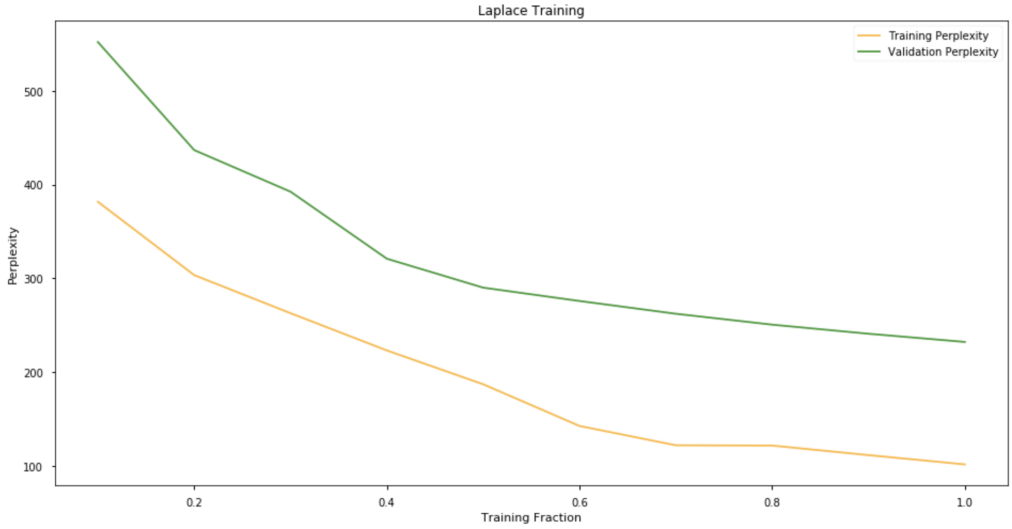
```
(venv) madhu@Madhumithas-MacBook-Pro code % python3 assignment1.py
--------------------------------------------------------------------------------
Using 1137951 tokens for training (100% of 1137951)
--------------------------------------------------------------------------------
Most frequent n-grams for n=1,2,3
(('the',), 63536)              (('.', '``'), 7753)            (('.', '``', 'i'), 1452)
((',',), 47952)                (('in', 'the'), 6706)          (('said', '.', '``'), 1149)
(('.',), 39355)                (('of', 'the'), 6659)          ((',', "'", 'he'), 1004)
(('to',), 28752)               ((',', "'"), 4978)             (('.', '``', 'we'), 850)
(('of',), 24104)               (('.', 'the'), 4547)           (('the', 'united', 'states'), 769)
(('in',), 23446)               (('.', "'"), 3089)             ((',', "'", 'said'), 767)
(('and',), 22935)              ((',', 'the'), 2762)           (("'", 'he', 'said'), 720)
(('a',), 22643)                (('to', 'the'), 2723)          (('.', '``', 'the'), 685)
(('``',), 12585)               (('said', '.'), 2373)          (('he', 'said', '.'), 644)
(("'s",), 12312)               (('for', 'the'), 2348)         (('.', '``', 'it'), 618)
Saved plot at figure.pdf
--------------------------------------------------------------------------------
Using vocab size 10000 (excluding UNK) (original 41844)
/Users/madhu/Desktop/code/util.py:90: RuntimeWarning: divide by zero encountered in log
  logprob += np.log(self.bi_prob[idx, corpus[i+1]])
Train perplexity: 70.967555
Val Perplexity: inf
```

My validation perplexity was infinity (inf) because in the plain bigram model, when we do not see a word in training, we assign it a probability of 0. When calculating perplexity, the sum of the log of all probabilities is taken, so when there is a probability of 0, the perplexity ends up being infinity.

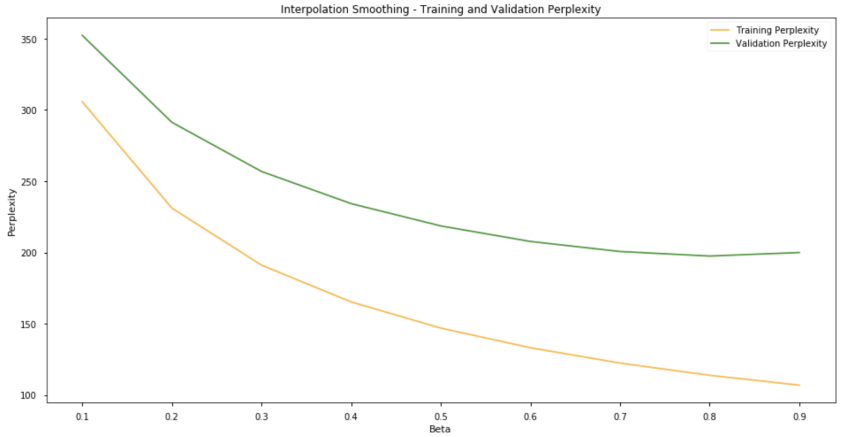
5. Below is a figure of the perplexity on both train and validation sets as a function of alpha after Laplace smoothing has been implemented. The graph grows somewhat smoothly after the initial jump seen through alpha values of 0.00001 to 1. Laplace smoothing fixes the issue with the validation perplexity from question 4 by assigning every word a small probability rather than 0. This helps to avoid receiving a perplexity of infinity.



6. The perplexity decreases as we increase the fraction of training data each time. This is intuitive because we expect the model to perform better with more data.



7. The training and validation perplexity decrease as the beta values increase. The figure's regression models exponential decay.



8. Minimum Validation Perplexity: 197.47 from Interpolation Smoothing