Introduction to NLP

Assignment-1

Tokenization,Language Modelling and Smoothing

ADITYA SHARMA

2021201016

Task1: Tokenization

- First all the Mentions has been replaced with <MENTION>
- All the Url have been replaced with <URL>
- All the hashtag have been replaced with <HASHTAG>
- All the numbers(containing digits) have been replaced with <NUM>
- All the punctuations have been replaced with empty strings.

Task2: Ngram Model

- In the scope of assignment, we are supposed to do 4-gram modelling.
- Firstly, I have stored all the possible 4-gram tokens with their frequency of how many times that word has come.
- Then I have kept the store of the number of distinct words that can start from that word and number of words that come after this word.
- I have also stored the count of all the words that have ended with this word as a reverse count of that word.

Task3: smoothing

- **Kneser Ney Smoothing**

Formula Used:

$$P_{\mathrm{KN}}(w_i|w_{i-n+1}^{i-1}) = \underbrace{\frac{\max(c_{KN}(w_{i-n+1}^{i}) - d, 0)}{c_{KN}(w_{i-n+1}^{i-1})}}_{\texttt{firstTerm}} + \underbrace{\lambda(w_{i-n+1}^{i-1})}_{\texttt{lambda}} \underbrace{P_{KN}(w_i|w_{i-n+2}^{i-1})}_{\texttt{Pcont}}$$

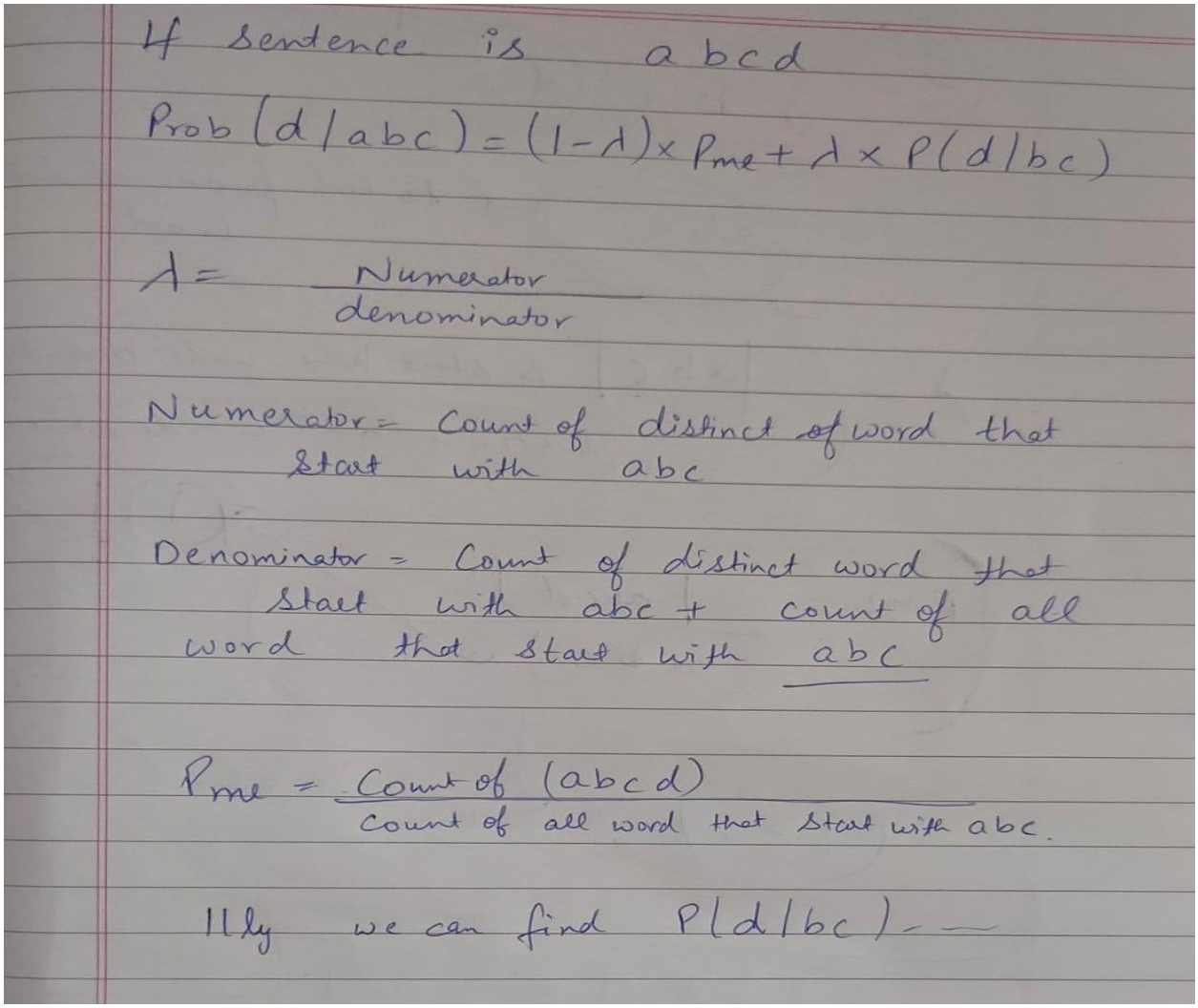
This will give us the probability of a word that can occur after some word as a sentence.

$$c_{KN}(\cdot) = \begin{cases} \text{count}(\cdot) & \text{for the highest order} \\ \text{continuationcount}(\cdot) & \text{for lower orders} \end{cases}$$

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} |\{w : c(w_{i-1}, w) > 0\}|$$

$$P_{\text{CONTINUATION}}(w_i) = \frac{|\{w_{i-1} : C(w_{i-1} w_i) > 0\}|}{\sum_{w'_i} |\{w'_{i-1} : c(w'_{i-1} w'_i) > 0\}|}$$

- **Witten-Bell Smoothing**



Task 4: Perplexity Score Calculation:

I have calculated perplexity by using this formula

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_N)}}$$

That is f= **loat(1)/float(math.exp(float(probability)/float(n))),**
Here n is the length of the sentence.

Task 5: Observation and Analysis

For Pride and Prejudice:

Kneser-Ney

LM1_train_perplexity:3.2879167993992175
LM1_test_perplexity: 104.90032662442454

Witten-Bell

LM2_train_perplexity:1.6510589552417863
LM2_test_perplexity: 102.28541933886075

For Ulysses

Kneser-Ney

LM3_train_perplexity:2.5956186862220143
LM3_test_perplexity: 172.04404059332091

Witten-Bell

LM4_train_perplexity:1.4410041807855742
LM4_test_perplexity: 201.55030630381583

Kneser-Ney took more time as compared to Witten Bell because in knesser-Ney we require to calculate all possible prefixes in the PContination Count. So,Witten-bell is more conservative.

Neural Language Model

Task1: Tokenization

- First all the Mentions has been replaced with <MENTION>
- All the Url have been replaced with <URL>
- All the hashtag have been replaced with <HASHTAG>
- All the numbers(containing digits) have been replaced with <NUM>
- All the punctuations have been replaced with empty strings.

Task2: Splitting the dataset

- The dataset has been split in the order of 70:15:15 for Train : Test : Validation.

Task 3 : Building the Neural Model :

The following are used :

- Loss = ‘categorical_crossentropy’
- Optimizer = ‘adam’
- Activation = ‘softmax’
- Metrics = [‘accuracy]

Task 4 : Calculating the Perplexities:

I have used the formula to calculate the perplexity

= float(1)/float(math.exp(float(probability)/float(n))),