

ASSIGNMENT-1 REPORT

TASK-1:

Byte Pair Tokenizer:

- 1) As a first step, we are reading the corpus file, and calculating the frequency of each word in the corpus.
- 2) Post this, we are maintaining the frequency of all possible substrings derived from word frequency. Suppose that 'apple' occurs in the corpus 10 times, then we will also increment the frequencies of substrings like 'ap', 'app', 'ppl' and others by 10.
- 3) Once we have the above data, we take the initial vocabulary as all individual lowercase alphabets, and the '\$' symbol (end of token symbol). We also create a separate table that has frequencies of all substrings of size 2 (the first merge would combine 2 letters, hence size 2).
- 4) From the latest table, we obtain the most frequent possible merged token, and add it to the vocabulary. We also update this table with the frequencies of all possible token combinations that can be obtained in the next merge operation. We also maintain a merge rule list to store the current merge rule.
- 5) After running our algorithm for 'k' iterations, we obtain 'k' merge rules, and 27+k tokens in the vocabulary, which are written to output text files as expected. This concludes the learn_vocabulary() method.
- 6) In the tokenize() method, we pass a list of sample input sentences as arguments. These sentences are tokenized independently, and the obtained tokens are written on a single line for 1 sentence.
- 7) This method basically separates all the words based on spaces. Then subword tokenization is done based on the learnt vocabulary. So, starting

from the first position in a word, the token with maximum length that matches the substring's part is taken, and then the position index is incremented accordingly.

- 8) There are 2 tokens.txt files, one has repeated tokens, as obtained from merging of more frequent lesser length tokens, and the second file contains unique tokens from the previous list.

Task-2

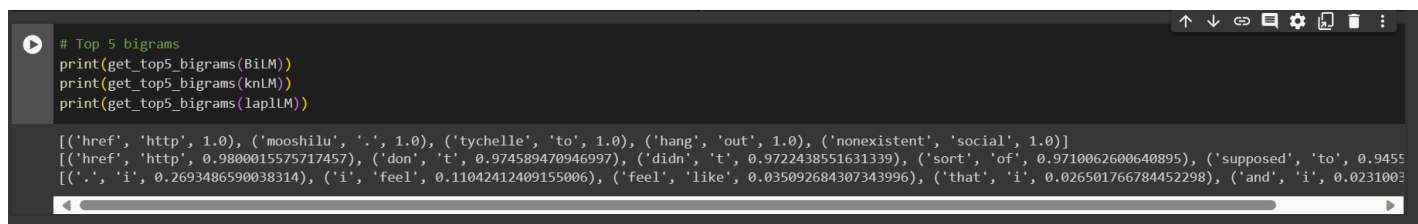
- 1) Bigram LM Implementation: In this implementation, we split the corpus into its constituent words, and maintain a list of unique words, count of unique words, and all words in order. Then we go through the list of all words in order, and increment the count for a pair of words as and when we encounter them. Based on these counts, we apply the formula:

$$P(w_i|w_{i-1}) = (\text{count}(w_{i-1}w_i) / \text{count}(w_{i-1}))$$

And print the top 5 bigrams based on probability.

- 2) Smoothing Implementation: Firstly, we have implemented the Laplace smoothing. In this case, we have taken all possible bigrams from the words in the corpus and set their initial count to 1. Then, based on occurrences in the corpus, the count of the bigrams is updated.
Next, we have implemented the Kneser Ney smoothing. As discussed in class, this algorithm works on the basis of distributing probability to 0 probability bigrams from those bigrams which have minimal non-zero probabilities. The probabilities of these are then updated with those from the next minimal probabilistic bigrams, and this process continues on.

The top 5 bigrams obtained before adding the emotion component is shown in the below screenshot:



```
# Top 5 bigrams
print(get_top5_bigrams(BiLM))
print(get_top5_bigrams(knLM))
print(get_top5_bigrams(lapLM))

[('href', 'http', 1.0), ('mooshilu', '.', 1.0), ('tychelle', 'to', 1.0), ('hang', 'out', 1.0), ('nonexistent', 'social', 1.0)]
[('href', 'http', 0.9800015575717457), ('don', 't', 0.974589470946997), ('didn', 't', 0.972243851631339), ('sort', 'of', 0.9710062600640895), ('supposed', 'to', 0.9455
[('.', 'i', 0.2693486590038314), ('i', 'feel', 0.11042412409155006), ('feel', 'like', 0.035092684307343996), ('that', 'i', 0.026501766784452298), ('and', 'i', 0.0231003
```

The first line shows the 5 bigrams in the standard case, without any smoothing. They are {href, http}, {mooshilu, .}, {tychelle, to}, {hang, out}, {nonexistent, social}, all having probability 1.

The Laplace smoothing result is on the last line. The top 5 bigrams are {., i}, {i, feel}, {feel, like}, {that, i}, {and, i}. In this case, 3 of the 5 bigrams have probability less than 0.04.

In Kneser Ney smoothing, the top 5 bigrams are {href, http}, {don', t}, {didn', t}, {sort, of}, {supposed, to}, all of them having probabilities in range 0.94-0.98.

If we compare the bigrams and probabilities, we see that Kneser Ney gives much higher probability than Laplace. Note that in both cases, there are no 0 probability bigrams. However, Laplace smoothing highly affects the most probable bigram in the corpus... we can see that {href, http} is not in top 5 in the Laplace case. This indicates that Laplace highly modifies the frequency structure of the words to make them equitable.

In Kneser Ney, bigrams do lose some probability, but it is less than Laplace. This indicates that the frequency structure of the words in the corpus is not impacted as largely as Laplace.

Due to this reason, Kneser Ney seems a better choice for smoothing the corpus.

3) We have used 2 methods for generating emotion oriented sentences:

- a) Using separate probability distribution of words/bigrams for different emotions: We classified each generated sentence into emotions. Each generated sentence is assigned the emotion corresponding to the largest probability. All the bigrams in a given sentence with for example emotion "sadness" will contribute to bigram probability/frequency distribution for sadness. In this way we got 6 separate bigram distributions corresponding to each 6 emotion. Now while generating sentences corresponding to a given emotion we will use the distribution corresponding to that emotion.
- b) Modify the standard probability of bigram model:

$$P(w_i|w_{i-1}) = (count(w_{i-1}w_i)/count(w_{i-1})) + w_1 * e_1 + w_2 * e_2 + w_3 * e_3 + w_4 * e_4 + w_5 * e_5 + w_6 * e_6$$

Here w_1, \dots, w_6 are weights which are calculated based on frequency of bigrams in each emotion sentence and e_1, \dots, e_6 are variables which can have a value 0 or 1 depending on what sentence we want to generate. If we want to generate an emotion of type e_1 , only e_1 will be one while rest will be zero. In this way we can take into account the distribution of bigrams in sentences with different emotions. We also multiply a constant weight = w to all w_1, \dots, w_6 .

Generated Samples for each emotions

1) Anger:

- a) i was just pissed my left me but i dont see what i cant help feeling dazed and helpful to give me would drive over

Reason: “pissed” usually shows anger.

- b) i have another for my protection and being insulted because it scares me if they way its all contribute to some cynicism involved in my thing up for zach in time to be with the work in wide and purpose to do feel angry that we laughed a fraction of jealousy before bed though his words but i should feel like i ve resigned summer moments i began to fail so open arms

2) Fear:

- a) i feel scared that my carcass over so vulnerable

Reason: “scared” and “vulnerable” corresponding to fear.

- b) i have a handful of my hair feel unprotected if possible the thoughts because hes trying to go of the time it go of not going down and a list to share with friends our region manager

3) Joy:

- a) i feel like cupcakes might have a bit stunned and more about that ive been following you it all that rich every time this history stuff i feel incredibly generous this subject

- b) i was feeling very happy and off

Reason: “feeling very happy” corresponds to joy.

4) Love:

- a) i am i really caring towards that is in a week was actually feeling a loving children in your fragrance

Reason: “caring” and “loving” correspond to love.

- b) i feel less loving of the fact ive talked with the lively room simply isn t want to be loving of life to his work is having my family is from try not and i know also it because i am fond of you after i truly caring to shove tender and my beloved darlings off

5) Sadness:

- a) i would be so pathetic feeling rejected its cause bonka neva thanked me than you go to my family if i can be at work from committing suicide or upset about nick not looking and chance to emphasize that i do jack around to feel awkward teenager when in bad impression with girly stuff because i feel rotten if its teeth but yesterday the same batch

Reason: “pathetic feeling”, “committing suicide”, “upset”, “bad impression”, “feel rotten” - all these words correspond to sadness.

- b) i was so unhappy about the kids which is a totally unworthy whenever i need to tell him feel like i get pregnant

6) Surprise:

- a) i should hear that maybe i feel surprised that i love i got really big one way

Reason: “feel surprised” corresponds to surprise.

- b) i am feeling amazed by the longing to become weird dreamworld where they have some reason to new and necessary structure and interacting with your hand on the girls i think of the feelings

4) We generated 6 files for 6 emotion. Then we took the original data and generated samples. For this we first did preprocessing and used `tf_idfvectorizer` to vectorize it into `tf_idf` matrix. After which we trained the model with training data (excluding the 300 data points from `tf_idf` matrix) and then we evaluated it on the 300 samples.

Credit Statement:

- 1) Implementation of BPE Tokenizer (Task 1) : Rohan Gupta (2020113)
- 2) Implementation of Bigram (Standard + Beta Component Modification):
Vatsal Chaudhary (2020549)
- 3) Implementation of Smoothing Algorithms : Jay Saraf (2020438)
- 4) Extrinsic Evaluation : Harsh (2020061)