Natural Language Processing – Assignment 2

Mias Ghantous – 213461692

Faisal Omari – 325616894

Due Date: 11/02/2024

In our work we programed a class that construct 3 dictionaries the first is a counter of unigrams $frequency\_dictionary$, the second is the bigram counter $frequency\_dictionary\_2\_words$, and finally for the trigram $frequency\_dictionary\_3\_words$.

We did all of this in the constructer which takes the type of the protocol as a string and the path of the corpus (We used the example corpus that were attached with the assignment), also the constructer saves the corpus size.

Part 1:

Calculate_prop_of_sentence:

1) We calculate the prop of the sentence with the help of the dictionaries that we constructed in the constructer.
2) For the linear smoothing to avoid the division by 0 we decided to use the "get" function to see if we have the key and if not put 1 in the denominator and 0 in the numerator if we are not calculating the unigram, and if we are calculating the unigram then do Laplace smoothing.
3) For the first and the second word we decided to include them in the probability by calculating the unigram and the bigram probabilities for the first and the second words, and calculate the trigram probability for all the other words.
4) If the sentence has less than 3 tokens, then the previous note take care of it.
5) For the λ_s in the linear smoothing we decided that for the tri gram 0.7 for the bigram 0.2 for the unigram 0.1

$generate\_next\_token$:

first we take the last 2 words from the original sentence because these are the words that are going to affect the probability of the next token, and then we try all the possible tokens that appear in the corpus (we can access them from $frequncy\_dictionary.keys()$) and use the previous function that we implemented to calculate the probability and return the token with the max probability.

Part 2:

$get\_k\_n\_collocations$:
we create a dictionary as a counter of the occurrences of a collocation (as a string), we read the right data from the corpus for each sentence in the corpus, get the tokens, if we have less than n tokens then don't do anything. The first collocation is from 0 to n-1 increase the counter of the collocation and then go to the next collocation by deleting the first token and add the next token in the sentence do this until the sentence has no more tokens. In the end we use the "heapq" library to get the required.

For the print part of the section we implemented a function called Q2_text, we made 2 modules one for committee and one for plenary and create the required text with the help of the 2 modules and the previous function.

Part 3:

For this part we programed function called Q3_text, we iterate throw every word in each sentence if the word is $[*]$ then calculate the next token for both types according to the model we've implemented and fill the generated token in the right place replacing it with the $[*]$, while if the word is not $[*]$ then we continue until we see another $[*]$ or till the end of the corpus, and at the end save the text in a txt file.

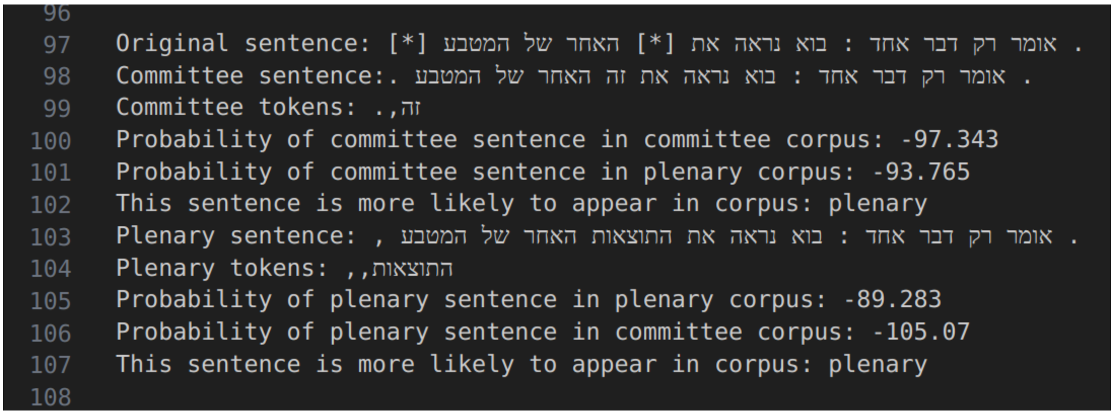
Part 4:

1) In our 2 modules sometimes we have same token predictions but for the most part we don't have the same results, why? Because we have completely separate data and 2 different ways of talking, and that explain the difference, because each one of the corpus types which are the committee and plenary may talk about different topics which leads to different words type used in the sentences, and because of that the model has been trained on one type only, so that do affect the model results.

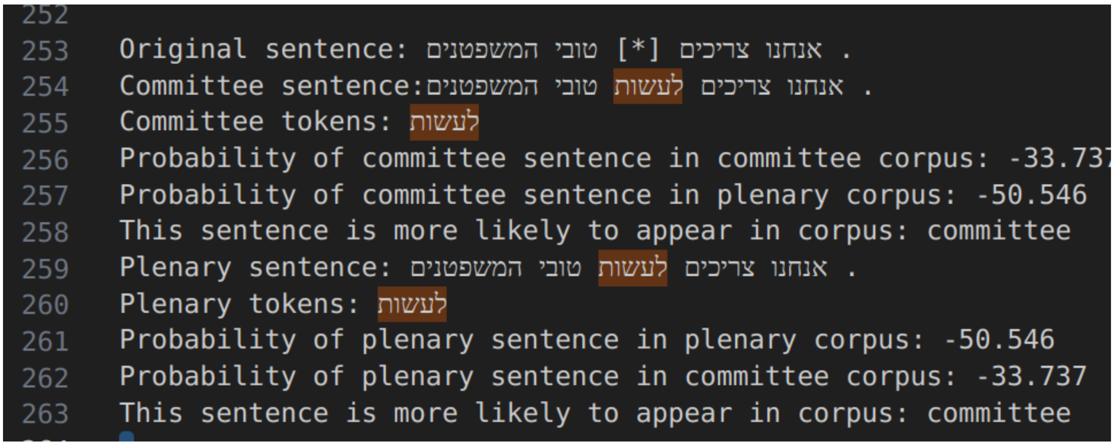
But as we saw in the previous assignment that there will still be some words that must be completed with another word regular which create some type of daily used phrases, for example: the very common collocation like "את זה" of course both the models are going to predict the same thing "זה", and this property leads to the similarity that we see between the models.

2) The collections meet our expectations because we have a lot of punctuation marks and words that are used in day to day life: "אני, אבל" and appears that we have some words like "חבר, כנסת, היושב, ראש" that we may not use in the day to day life but they are famous because of the type of the corpus (Knesset corpus) so of course they are going to be famous collocations in our corpus.

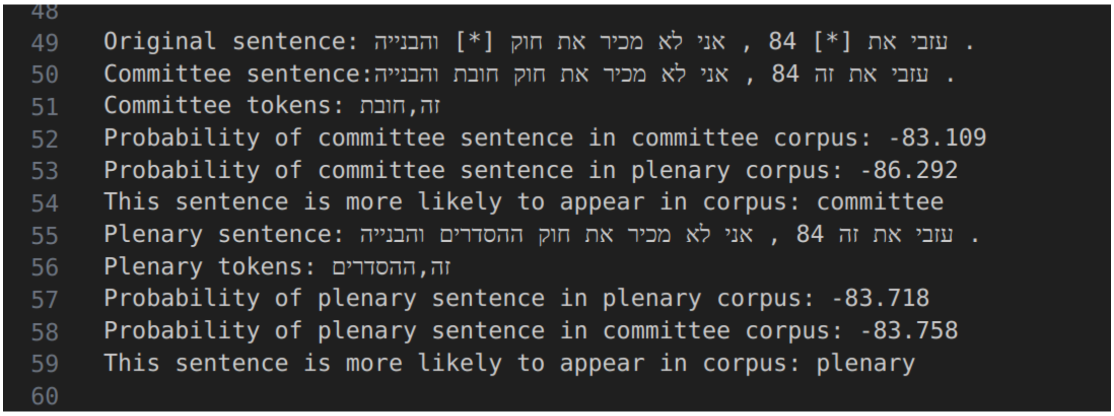
3) We have a lot of fine predictions like:



```
Original sentence: אומר רק דבר אחד : בוא נראה את [*] האחר של המטבע [*] .
Committee sentence:אומר רק דבר אחד : בוא נראה את זה האחר של המטבע . .
Committee tokens: זה,.
Probability of committee sentence in committee corpus: -97.343
Probability of committee sentence in plenary corpus: -93.765
This sentence is more likely to appear in corpus: plenary
Plenary sentence: אומר רק דבר אחד : בוא נראה את התוצאות האחר של המטבע , .
Plenary tokens: התוצאות,,
Probability of plenary sentence in plenary corpus: -89.283
Probability of plenary sentence in committee corpus: -105.07
This sentence is more likely to appear in corpus: plenary
```



```
Original sentence: אנחנו צריכים [*] טובי המשפטנים .
Committee sentence:אנחנו צריכים לעשות טובי המשפטנים .
Committee tokens: לעשות
Probability of committee sentence in committee corpus: -33.737
Probability of committee sentence in plenary corpus: -50.546
This sentence is more likely to appear in corpus: committee
Plenary sentence: אנחנו צריכים לעשות טובי המשפטנים .
Plenary tokens: לעשות
Probability of plenary sentence in plenary corpus: -50.546
Probability of plenary sentence in committee corpus: -33.737
This sentence is more likely to appear in corpus: committee
```



```
Original sentence: עזבי את [*] 84 , אני לא מכיר את חוק [*] והבנייה .
Committee sentence:עזבי את זה 84 , אני לא מכיר את חוק חובת והבנייה .
Committee tokens: זה,חובת
Probability of committee sentence in committee corpus: -83.109
Probability of committee sentence in plenary corpus: -86.292
This sentence is more likely to appear in corpus: committee
Plenary sentence: עזבי את זה 84 , אני לא מכיר את חוק ההסדרים והבנייה .
Plenary tokens: זה,ההסדרים
Probability of plenary sentence in plenary corpus: -83.718
Probability of plenary sentence in committee corpus: -83.758
This sentence is more likely to appear in corpus: plenary
```

But of course, if we get a bigger corpus we are going to have a lot more fine predictions, because it is known for us that the more data we have the more stable model will get, and that also improves the results because of avoiding overfitting on some repeated sentences in specific corpus, or underfitting because of no enough data to capture some repeated sentence phrases from it and predicting with some higher value probability which makes the model more confident and more stable.

4) If we were to use Bigram module then we would have got worst results because then the module would be less knowledgeable on the relations between the words, which results in worst predictions, and that is the main disadvantage in this kind of models, because it builds its prediction on some unstable word counts sentence, and that makes it affected when dealing with complex sentences and makes the predicted token not fitting on the right place and having lower probability than making the model confidence.