Assignment 1

Neha Iyer
`SR No. 14942`

1 Task 1

The ngram language model was implemented under following settings

2 Datasets used

1. D1 Brown corpus
2. D2 Gutenberg corpus

3 Data Split

The dataset was divided into train and test sets in following manner:
1. For D1 Brown corpus, the train set was ceated by taking 80% from each category and the test set was created by taking remaining 20% of each category.
2. For D2 Gutenberg corpus, the train set was ceated by taking 80% from each file and the test set was created by taking remaining 20% of each file.

4 Smoother

The smoother used for the language model is stupid backoff as it suitable for large scale data.

5 Handling of unknown words

The cases of unknown words was handled by replacing 500 words that has occured only once in the training set with token "UKN".
The probability of "UKN" was estimated from its count like any other regular word in the training set.
When an unknown word is seen in the test set, it is assigned with the probability of the "UKN" token in the training set.

Case	ngram	Perplexity
S1	3	571.27
S2	3	45.33
S3	3	59.54
S4	3	33.66

Table 1: Perplexity measure for trigram model

Case	ngram	Perplexity
S1	2	52.74
S2	2	56.79
S3	2	90.37
S4	2	67.53

Table 2: Perplexity measure for bigram model

6 Implementation

The ngram model was built under following four settings:
Note that the implementation was performed using Python v3.6

- S1: Train: D1-Train, Test: D1-Test
- S2: Train: D2-Train, Test: D2-Test
- S3: Train: D1-Train + D2-Train, Test: D1-Test
- S4: Train: D1-Train + D2-Train, Test: D2-Test

7 Evaluation metric

For each of the above case, the bigram and trigram model was built and evaluated using perplexity metric.

8 Results

1. Both bigram and trigram model was built for each of the above setting
2. The perplexity measure for both the models is as shown in Table 1 and Table 2
3. The trigram perplexity was turning out to be higher than bigram in case of setting S1
4. The possible reason could be due to overfitting of the model on training data

9 Task 2

1. From the results, it is observed that the trigram model is performing marginally better than the bigram model.
2. In particular the setting S4 is giving the least perplexity measure
3. So we generate the sentence with a trigram model trained on seeting S4
4. On executing the script generate_sentence.sh, the sentence is generated in a file called "output.txt"
5. An example sentence generated is as "very much to be a very good natured and I will"