Assignment 1 - Language Modeling
Report (15CS10053)

- Execute the script as:
  python Assignment_1_15CS10053 <*path to input test file*>
- For the output scores please refer ‘**output.txt**’

Task 1

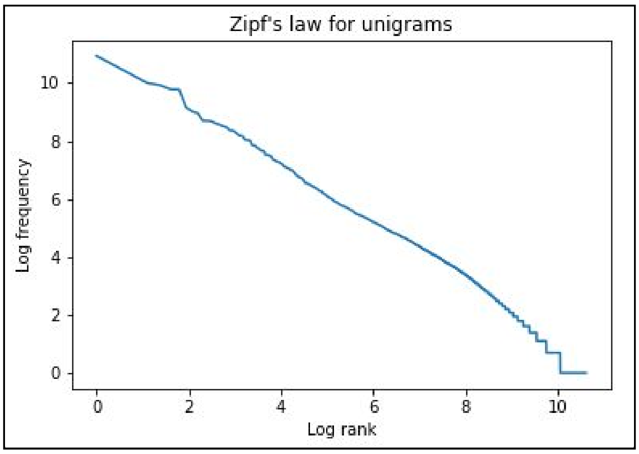
- Created the n-gram language models (n=1,2,3) with padding for n>1.
- Verification of **Zipf’s Law**:

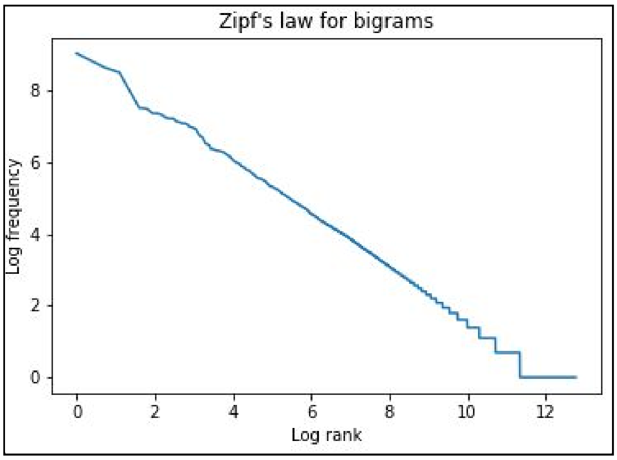
Zipf’s law states that the frequency of a word (f) is inversely related to its position in the list or rank (r).

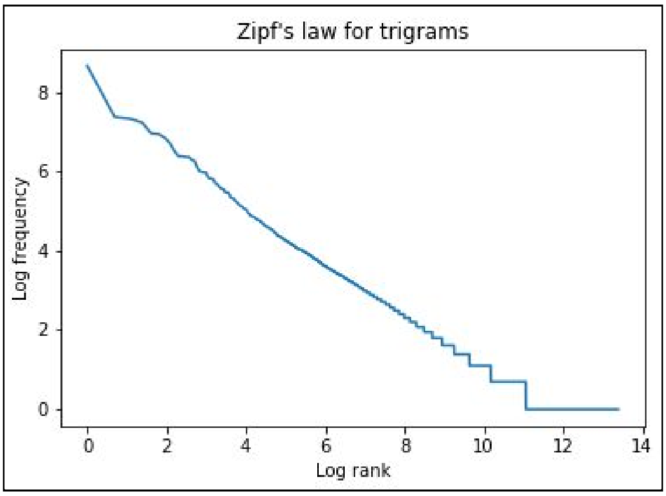
$$f \propto 1/r$$

To verify it, we plot logarithm of word frequency (f) versus logarithm of word rank (r) which should be a straight line with negative slope.

$$\mathbf{\log(f) + \log(r) = Constant}$$







- For the trigram model, many test-sequences give 0 probability since the absence of even one trigram in training model dictionary assigns the probability of the complete test-sequence as zero. Such trigrams give negative INF(infinity) as log-likelihood and positive INF as perplexity score
- The top n-grams consist of stopwords like **‘the’ (56448), ‘of’(31276), ‘and’(22092), ‘to’(20341), ‘a’(17780)**, etc. which is expected.

Task 2

- Applying Laplacian smoothing assigns a non-zero probability to every test-sequence
- We add alpha ∈ (0,1] to numerator and alpha*|N| to the denominator. |N| is the number of unique unigrams (vocabulary size)
- *As weight k increases, log-likelihood decreases*. Given below are the log-likelihood values for the test sentence “he lived a good life”

Language model	Log-likelihood (k=0)	Log-likelihood (k=0.0001)	log-likelihood(k=0.001)	log-likelihood(k=0.01)	log-likelihood(k=0.1)	log-likelihood(k=1)
unigram	-32.6957	-32.6957	-32.6960	-32.6981	-32.7196	-32.9292
bigram	-26.7533	-26.8354	-27.4257	-29.9723	-35.8388	-44.3439
trigram	-inf	-47.8799	-47.1901	-49.4380	-54.5686	-60.8976

Task 3

- Assign the probability mass of n-grams that occurs r+1 in the training corpus to those n-grams that occurs occurs r times. This is done for all r except $r_{max.}$ The probability distribution of the most frequent bigrams and trigrams remain the same as before.

- It is not possible to apply Good Turing smoothing to unigram language model because all possible uni-grams are seen at-least once. We have no way to predict new unseen words which might come up in test sequences. Unigrams which occur 0 times is 0. Hence, the effective count of unigrams not seen in corpus is not defined (0* = 1*(unigrams which occur once)/0).

  Effect on GT smoothing on test sentence : “he lived a good life”

Language model	Log-likelihood (w/o smoothing)	Log-likelihood (with GT smoothing)
bigram	-26.7533	-66.2061
trigram	-inf	-142.4721

Task 4

- Interpolation ensembles different n-gram models. This has a smoothing effect. For e.g. Test bigrams not seen in corpus might occur as uni-grams in corpus. Therefore, the unigram model will assign the test sequence some non-zero probability.

- As lambda increases, contribution of bigram model increases. Since bigram model takes a small context into account, the test-sequence probability increases with lambda.
  Effect on interpolation on test sentence : “he lived a good life”

Language model	Log-likelihood (lambda=0)	Log-likelihood (lambda = 0.2)	Log-likelihood (lambda = 0.5)	Log-likelihood (lambda = 0.8)
bigram	-26.7533	-32.4952	-29.3943	-27.6202