CS565: Intelligent Systems and Interfaces

Assignment 1

Topics: Tokenization, N-Grams, Collocation, and Morphological Analysis

Roll no. - 170101022

Name – Devansh Gupta

Hindi Corpus Analysis Google Collab Link:-
https://colab.research.google.com/drive/1UdTBAQsxcp4Qajxg8eY6t1so2GpOh2T4?usp=sharing

English Corpus Analysis Google Collab Link:-
https://colab.research.google.com/drive/1nXuJPw0o9FQ9Zo4svzL-ypH35pKL9_p3?usp=sharing

1.3.1 Analysis using existing NLP tools [15 marks]

Tool used: - NLTK

1. Options: -

- For English: -
  - ♦ Sentence Segmentation:
    - ➢ Method1: I used sent_tokenize() function, which is already trained and thus very well knows to mark the end and beginning of sentence at what characters and punctuation.
    - ➢ Method2: I used PunktSentenceTokenizer() and given our data to train using tokenizer.train(text).
  - ♦ Word Tokenization
    - ➢ Method1: I had used word_tokenize() function, it simply splits words in a sentence.
    - ➢ Method2: I had used TreebankWordTokenizer(), this tokenizer work by separating the words using punctuation and spaces and it does not discard the punctuation, allowing a user to decide what to do with the punctuations at the time of preprocessing.
- For Hindi: -
  - ♦ Sentence Segmentation:
    - ➢ Method1: I had used sentence_tokenize.sentence_split() function.
    - ➢ Method2: I had used standfordnlp.
  - ♦ Word Tokenization
    - ➢ Method1: I had used indic_tokenize.trivial_tokenize() function, it simply splits words in a sentence.
    - ➢ Method2: I had used analyzer.morph_analyze_document(text.split(‘ ‘) function.

2. Difference: -

- For English: -
  - ♦ Sentence Segmentation:

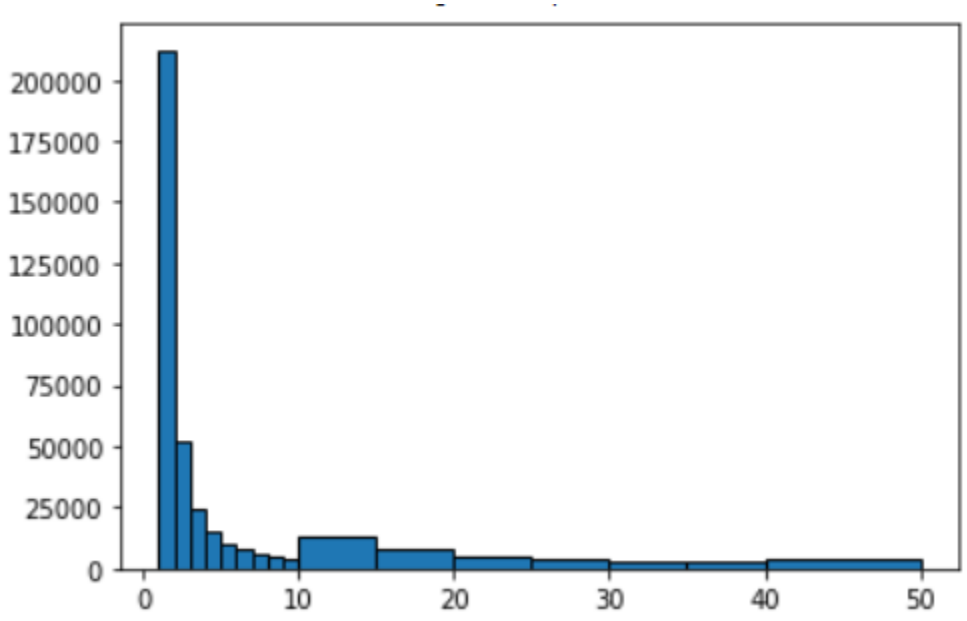
  Difference between method1 and method2 of sentence segmentation was method2 braked sentence at J.J. and continued next line from Thomson while method1 considered J.J. Thomson in one line.
  - ♦ Word Tokenization

  Difference between method1 and method2 of word tokenization was method2 considered word + “.” while method1 considered “.” as different token.

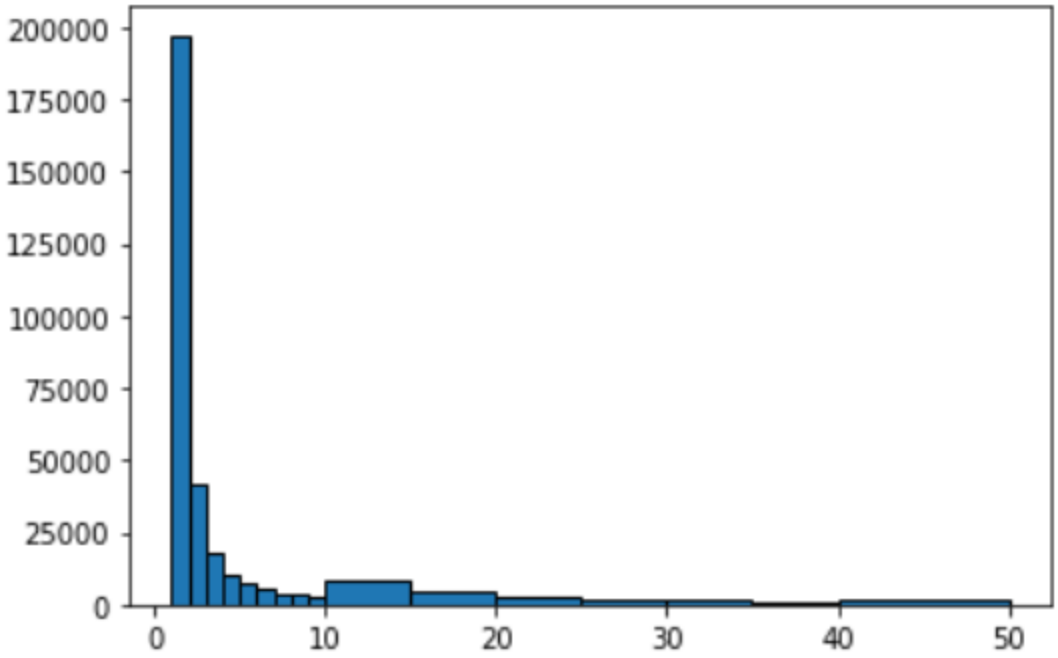
- For Hindi: -
    - ♦ Sentence Segmentation: No difference is observed
    - ♦ Word Tokenization:

    Method1 just split the words on the basis of space but Method2 first split the word on the basis of space then further splits word by morphologically analyzing word itself.

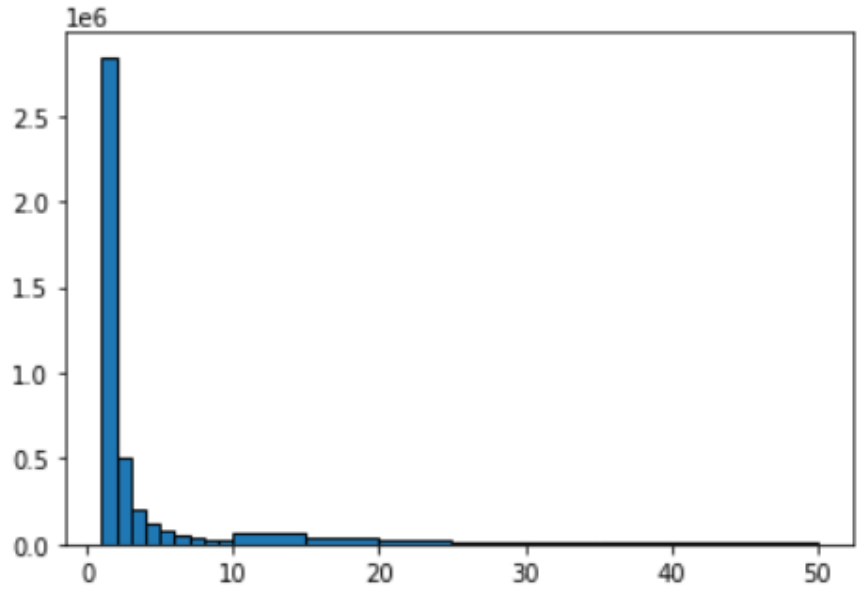
3. Frequency Distribution
    - ♦ Unigram:
        - ➢ Total no. of unigrams (Distinct) was 395,757 in English and 322,350 in Hindi. More than half of them has frequency 1 in both the cases.
        - ➢ Most frequent unigrams were either of articles, preposition and conjunctions. “The” has highest frequency i.e. 1,083,392 in English corpus and 'के' has highest frequency 357,833 in Hindi corpus.
        - ➢ Least frequent unigrams were either of quantity and complex scientific words. “100g” has frequency 1 in English corpus and '“कमीशन' has frequency 1 in Hindi corpus and any more has frequency 1 as mentioned in 1st point in both the corpus
        - ➢ We can characterize most frequent and least frequent. Most frequent are prepositions, articles, conjunctions. Least frequent are quantities and complex scientific words.
        - ➢ **x-axis**- frequency, **y-axis**- no. of Unigrams.
        - ➢ For English: -



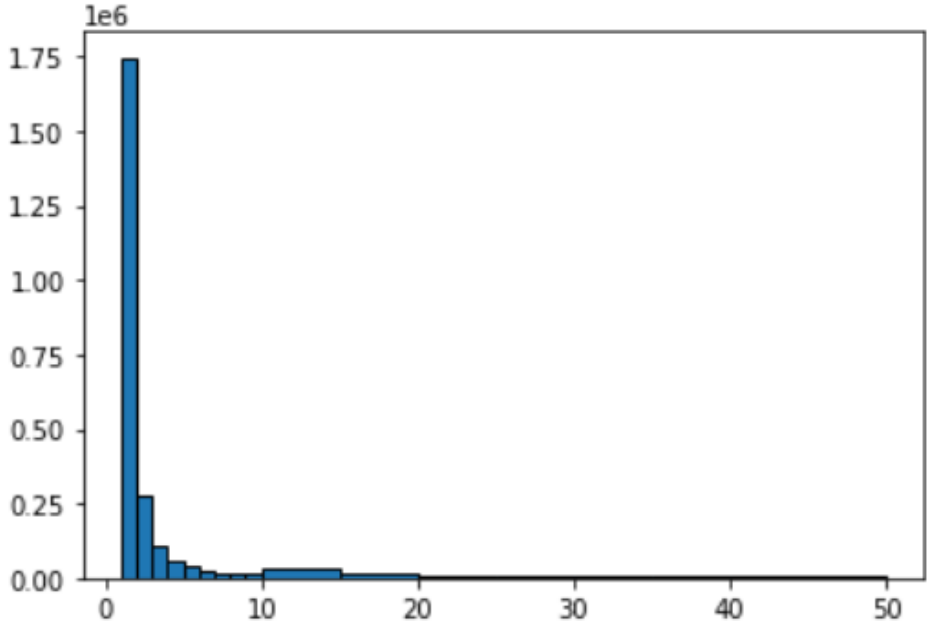
        - ➢
        - ➢ For Hindi: -



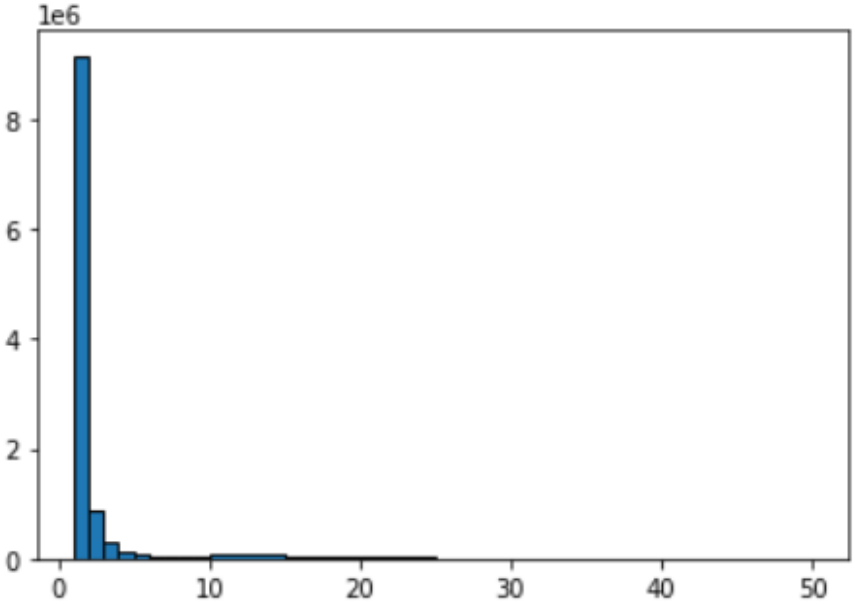
- Bigrams:
  - Total no. of bigrams (Distinct) was 4,095,732 in English and 2,392,979 in Hindi. Approx. 3/4th of the total has frequency 1 in both the cases.
  - Most frequent bigrams were pair of articles, preposition and conjunctions. “of the” has highest frequency i.e. 179411 in English corpus and ('है', '।') has highest frequency 125,369.
  - Least frequent bigrams were pairs of quantity and complex scientific words. “theological reasoning” has frequency 1 in English corpus and ('एमएचए', 'या') has frequency in Hindi corpus and any more has frequency 1 as mentioned in 1st point.
  - We can characterize most frequent and least frequent. Most frequent are prepositions, articles, conjunctions. Least frequent are quantities and complex scientific words.
  - **x-axis**- frequency, **y-axis**- no. of Bigrams (10^6)
  - For English: -



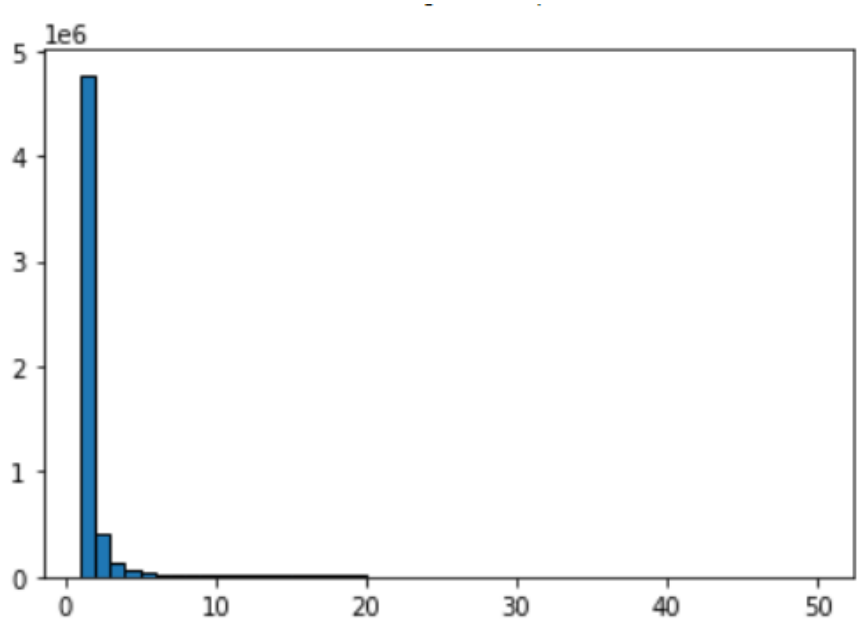
- For Hindi: -



- Trigrams:
  - Total no. of trigrams (Distinct) was 10,827,502 in English corpus and 5,508,737. Approx. 5/6th of the total has frequency 1 in both the corpus.
  - Most frequent trigrams were from of articles, preposition, punctuations and conjunctions. “, and the” has highest frequency i.e. 15519 in English corpus and 'के', 'रूप', 'में' has highest frequency 16,475 in Hindi corpus.
  - Least frequent trigrams were from of quantity and complex scientific words. “coined by ancient” has frequency 1 and 'हेल्थ', 'एडमिनिस्ट्रेशन', 'या' has frequency 1 and many more has frequency 1 as mentioned in 1st point.
  - We can characterize most frequent and least frequent. Most frequent are prepositions, articles, punctuations and conjunctions. Least frequent are quantities and complex scientific words.
  - **x-axis**- frequency, **y-axis**- no. of Trigrams (10^6)
  - For English: -



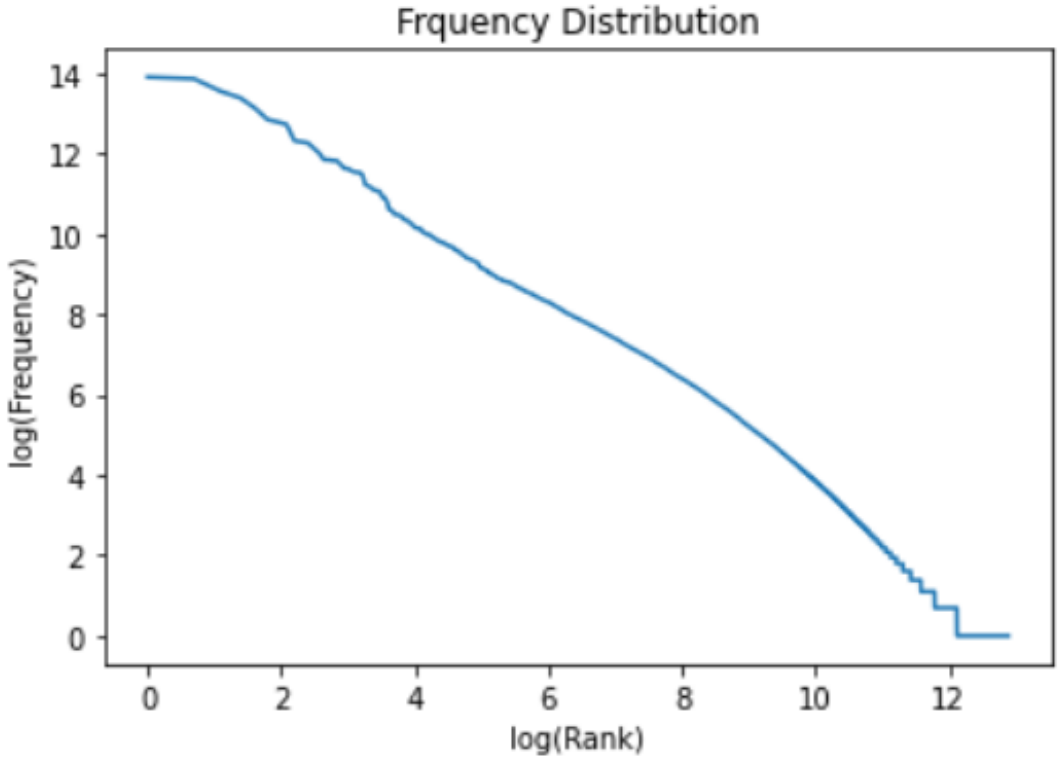
- For Hindi: -



4. Fitting Zips law in frequency distribution: -

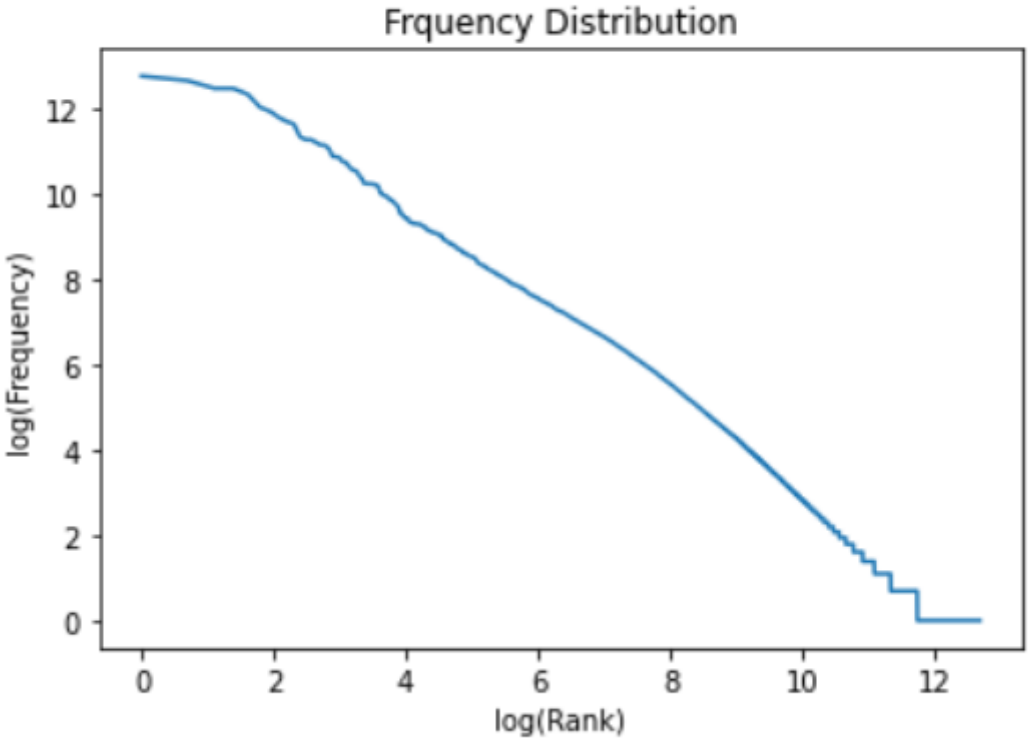
- For English: -

```
Intercept : [17.50200632]
Slope : [[-1.39427482]]
```



- For Hindi: -

```
Intercept : [14.42984786]
Slope : [[-1.17584404]]
```



- ♦ Observation: You can see that slop close to -1 indicates the validation that frequency and rank is inversely related and also you can observe that it deviate at very low frequency and at very high frequency. Hindi is more following Ziph's Law. as compare to English.

1.3.2 Few Basic Questions [5 marks]

- For English: -
  1. Only 13,970 (most frequent) out of 395,757 distinct unigrams are required for 90% coverage.
  2. 723,390 (most frequent) out of 4,095,732 distinct bigrams are required for 80% coverage.
  2. 4,946,832 (most frequent) out of 10,827,502 distinct trigrams are required for 70% coverage.
  3. After Stemming: -
     - ♦ Only 5704 (most frequent) out of 305,198 distinct unigrams are required for 90% coverage.
     - ♦ 422393 (most frequent) out of 3,334,382 distinct bigrams are required for 80% coverage.
     - ♦ 4,228,490 (most frequent) out of 10,109,160 distinct trigrams are required for 70% coverage.
- For Hindi: -
  3. Only 13,340 (most frequent) out of 322,350 distinct unigrams are required for 90% coverage.
  4. 664,973 (most frequent) out of 2,392,979 distinct bigrams are required for 80% coverage.
  4. 2,916,728 (most frequent) out of 5,508,737 distinct trigrams are required for 70% coverage.
  5. After Stemming: -
     - ♦ Only 7710 (most frequent) out of 276,548 distinct unigrams are required for 90% coverage.
     - ♦ 420,861 (most frequent) out of 2,008,144 distinct bigrams are required for 80% coverage.
     - ♦ 2,431,385 (most frequent) out of 5,023,394 distinct trigrams are required for 70% coverage.

6. Comparison: -
    - Without Stemming: - n-grams having different length or affixes but same stems were considered different. So, more frequent n-grams are required for given coverage. Also, you can see in histogram that frequency of n-grams having frequency 1 decreased after stemming because that n-gram had same stem n-gram available but without stemming those n-grams were considered different.
    - With Stemming: - n-grams having different length or affixes but same stems were considered same. So, frequency of n-grams increased, this result in less no. of frequent n-grams required for given coverage.

7. Result: -
    - Total no. of unigrams (repeated) = Total no. of bigrams (repeated) = Total no. of trigrams (repeated). This should be the case also because given array of size n, total no. of subarray of size two is n-1 and total no. of subarray of size three is n-2. (when n is large then all three are equal).
    - Total no. of distinct unigrams is very less than total no. of distinct bigrams/trigrams because in bigrams/trigrams multiple permutations are there and they will be considered as different n-grams and also happening two/three events simultaneously is less probable that happening one event (collocation may be there but it will be for very few words, so it will not affect much in distinct no. of bigrams/trigrams) that's why bigram/trigram have less frequency as compared to unigrams.
    - With Stemming frequency of each n-gram increases that results in decrease in no. of n-grams with less frequency.
    - Less no. of n-grams is needed for given coverage with stemming because frequency of each n-grams increases with stemming.

1.3.3 Writing some of your basic codes and comparing with results obtained using tools [15 marks]

1. Implementing Heuristic:

- For English:-
    - Sentence Segmentation: - "." which are not full stops are replaced by "<use>" and "." which are full stops are replaced by ".<stop>" and also "?" and "!" are replaced by "?<stop>" and "!<stop>". At the end, split is performed on "<stop>" this divides corpus in different sentences. I had considered some frequent cases which have "." but not full stop. They are:
        - Word Starting with capital letter and have "." at last. I assumed this word as abbreviation. Example: - "Wash.", "J.".
        - Some Prefixes like "Mr." "Mrs. "Dr." are also considered as non-full stops.
        - Websites having ".edu" ".com" ".org" and some more.
        - Starters ("The" "Their" "Whenever") are generally not used after abbreviation. So if we find starter after abbreviation then we considered starter as the starting of new sentence.
        - Multiple capital letter with "." between them are considered abbreviation.
        - Digits with "." between them are considered decimal no.

- Two small letters with “.’ between them and “.” at least is considered abbreviation. Eg. “i.e.”, “e.g.”.
- “…” is not considered full stop.
- Rest all (“.” ”?” and ”!”) are considered as end of the sentence.

♦ Tokenization: - All spaces” “ as are considered division of tokens. All “.” , “?” , ”’”, brackets, apostrophe, ”-“ , ”!” are also considered as division except some: -

- Word Starting with capital letter and have “.” at last. I assumed this word as abbreviation. Example: - “Wash.”, “J.”.
- Some Prefixes like “Mr.” “Mrs. “Dr.” are also considered as non-full stops.
- Websites having “.edu” “.com” “.org” and some more.
- Multiple capital letter with “.” between them are considered abbreviation.
- Digits with “.” between them are considered decimal no.
- Two small letters with “.’ between them and “.” at least is considered abbreviation. Eg. “i.e.”, “e.g.”.
- “…” is not considered as division.

- For Hindi:-
  - ♦ Sentence Segmentation: - Sentences are segmented on the basis of purna viram(‘|’).
  - ♦ Word Tokenization: - Words are separated based on space and hyphen(“ “ and “-“).

- Statistics: -

- For English: -
  - 10,186 (most frequent) out of 404,373 distinct unigrams are required for 90% coverage.
  - 805,484 (most frequent) out of 4,136,480 distinct bigrams are required for 80% coverage.
  - 5,099,073 (most frequent) out of 10,636,936 distinct trigrams are required for 70% coverage.
- For Hindi: -
  - 15,198 (most frequent) out of 371,475 distinct unigrams are required for 90% coverage.
  - 766,515 (most frequent) out of 2,293,853 distinct bigrams are required for 80% coverage.
  - 2,809,635 (most frequent) out of 5,100,641 distinct trigrams are required for 70% coverage.

- Comparison: -
  - More frequent ngrams are required for given coverage when tokenization is done by implementing heuristic as compared to the case when tokenization was done with the nltk tool.
  - Frequency of each ngram is reduced in heuristic approach as compared to tool based approach.
  - The main reason behind the above observation is we only considered some parameters for division there can be more characters which can cause division such as “/” and many more special characters and operators. And if there is some error/noice in the tokenization, then after stemming also some sub-words having same stem will be considered different which result in decrease in frequency.

2. Likelihood Ratio Test

SortedStemUni is the dictionary containing unigrams and their frequency.
SortedStemBi is the dictionary containing bigrams and their frequency.
N is calculated by adding all the frequencies of bigrams/unigrams

- Algorithm

```
N = 19602235   #calculated by adding frequencies of each bigram
likelihoodRatioBi = {}
for x, y in sortedStemBi.items():
  #print(x, y)
  w1 = x[0]
  w2 = x[1]
  c12 = y
  c1 = sortedStemUni[(w1,)]     #frequency of w1 from unigram
  c2 = sortedStemUni[(w2,)]       #frequency of w2 from unigram
  #print(c1,c2,c12)
  p= c2/N
  p1 = c12/c1
  p2 = (c2-c12)/(N-c1)
  #print(p,p1,p2)
  getcontext().prec = 100
  b11 = binom.logpmf(c12, c1, p)
  b12 = binom.logpmf(c2-c12, N-c1, p)
  b21 = binom.logpmf(c12, c1, p1)
  b22 = binom.logpmf(c2-c12, N-c1, p2)
  #print(b11,b12,b21,b22)
  lgL = (b11 + b12)-(b21 + b22)
  ans = 2*(-1)*lgL
  #print(ans)
  likelihoodRatio[x] = ans
```

- Observation: Not necessary that most frequent bigrams have more likelihood ratio for collocation because bigram frequency can be less condition frequency of w1, w2 is less separately i.e. when they occur, they occur together, this increases the likelihood of collocation. As a result, in top100 bigrams based on likelihood ratio, contain bigrams consisting of words other from preposition, punctuation, article, and conjunction. E.g. `('unit', 'state')` – "United" and "State" occurred less no. of times but when occurred, mostly occurs together. Other examples are `('median', 'incom'), ('the', 'citi')` ('New' 'York').

1.3.4 Morphological parsing [5 marks]

- Analysis: - Since most frequent words are more simpler words that's why they are divided into less no. of morphemes on the other hand least frequent words are more complex words that's why they are divided into more no. of morphemes and they contain variety of morphemes.
- Model: - The implementation uses Morfessor which is a family of probabilistic machine learning methods that find morphological segmentations for words of a natural language, based solely on raw text data. Models of the Morfessor family are generative probabilistic

models that predict compounds and their analyses (segmentations) given the model parameters.

1.3.5 Sub-word tokenization [20 marks]

- For English: - Bit Pairs encoding didn’t give satisfactory result because I had trained it on small datasets, so it is able to do for the words in the trained datasets but not for the words outside the data sets. But when I used Morfessor the results are very well because that was pre-trained on large data sets.