CS-565 | INTELLIGENT SYSTEMS AND INTERFACES
ASSIGNMENT-1
VAKUL GUPTA | 170101076

Link to Google Colab Notebook - **Click Here**

NOTE - Entire processing has been performed on full corpus (Both English and Hindi) unless specifically mentioned (only in stanza while processing for Hindi, partial data is taken due to RAM issues)

TASK 1.3.1 - Analysis using existing NLP tools

1. Sentence Segmentation and Word Tokenization

Sentence Segmentation (English)

- **First Method** - nltk Punkt sentence tokenizer
  Total number of sentence tokens - **761582**
  Output -

```
['The word "atom" was coined by ancient Greek philosophers.', 'However, these ideas were founded in philosophical and theological reasoning rather than evidence and experimentation.', 'As a result, their views on what atoms look like and how they behave were incorrect.']
```

- **Second Method** - nltk Regular expression tokenizer
  Pattern used - '(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s'
  Total number of sentence tokens - **750330**
  Output -

```
['The word "atom" was coined by ancient Greek philosophers.', 'However, these ideas were founded in philosophical and theological reasoning rather than evidence and experimentation.', 'As a result, their views on what atoms look like and how they behave were incorrect.']
```

Sentence Segmentation (Hindi)

- **First Method** - Sentence tokenizer using indicnlp module
  Total number of sentence tokens - **348593**
  Output -

```
['मास्टर ऑफ़ हेल्थ एडमिनिस्ट्रेशन या मास्टर ऑफ हेल्थ केयर एडमिनिस्ट्रेशन (एमएचए या एम. एच. ए) स्नातकोत्तर (पोस्ट ग्रेजुएशन) की एक पेशेवर डिग्री है जो स्वास्थ्य प्रशासन के क्षेत्र में दी जाती हैं।', 'यह उन छात्रों को प्रदान की जाती हैं जिन्होंने स्वास्थ्य प्रशासन, अस्पताल प्रबंधन एवं अन्य स्वास्थ्य सेवा संगठनों के क्षेत्र में जरूरी ज्ञान और दक्षता हासिल की हैं।']
```

- **Second Method** - Sentence tokenizer using stanza module.
  Total number of sentence tokens on 5 percent data (Due to RAM issues) - **17928**

Total number of sentence tokens are approximately - 20*17928 = **358560**

Output -

```
['मास्टर ऑफ़ हेल्थ एडमिनिस्ट्रेशन या मास्टर ऑफ हेल्थ केयर एडमिनिस्ट्रेशन (एमएचए या एम. एच. ए) स्नातकोत्तर (पोस्ट ग्रेजुएशन) की एक पेशेवर डिग्री है जो स्वास्थ्य प्रशासन के क्षेत्र में दी जाती हैं।', 'यह उन छात्रों को प्रदान की जाती हैं जिन्होंने स्वास्थ्य प्रशासन, अस्पताल प्रबंधन एवं अन्य स्वास्थ्य सेवा संगठनों के क्षेत्र में जरूरी ज्ञान और दक्षता हासिल की हैं।']
```

Word Tokenization (English)

- **First Method** - Treebank Word Tokenizer

  Total number of word tokens - **18915300**

  Output -

```
['The', 'word', '``', 'atom', "''", 'was', 'coined', 'by', 'ancient', 'Greek', 'philosophers.', 'However', ',', 'these', 'ideas', 'were', 'founded', 'in']
```

- **Second Method** - Punct Word Tokenizer

  Total number of word tokens - **20372526**

  Output -

```
['The', 'word', '"', 'atom', '"', 'was', 'coined', 'by', 'ancient', 'Greek', 'philosophers', '.', 'However', ',', 'these', 'ideas', 'were', 'founded']
```

Word Tokenization (Hindi)

- **First Method** - Trivial word Tokenizer from indicnlp module

  Total number of word tokens - **8640033**

  Output -

```
['मास्टर', 'ऑफ़', 'हेल्थ', 'एडमिनिस्ट्रेशन', 'या', 'मास्टर', 'ऑफ', 'हेल्थकेयर', 'एडमिनिस्ट्रेशन', '(', 'एमएचए', 'या', 'एम', '.', 'एच', '.', 'ए', ')', 'स्नातकोत्तर']
```

- **Second Method** - Word tokenizer from stanza module

  Total number of word tokens on 5 percent data (Due to RAM issues) - **420958**

  Total number of word tokens are approximately - 20*420958 = **8419160**

  Output -

```
['मास्टर', 'ऑफ़', 'हेल्थ', 'एडमिनिस्ट्रेशन', 'या', 'मास्टर', 'ऑफ', 'हेल्थकेयर', 'एडमिनिस्ट्रेशन', '(']
```

2. Statistical Analysis for Unigrams

- **English**

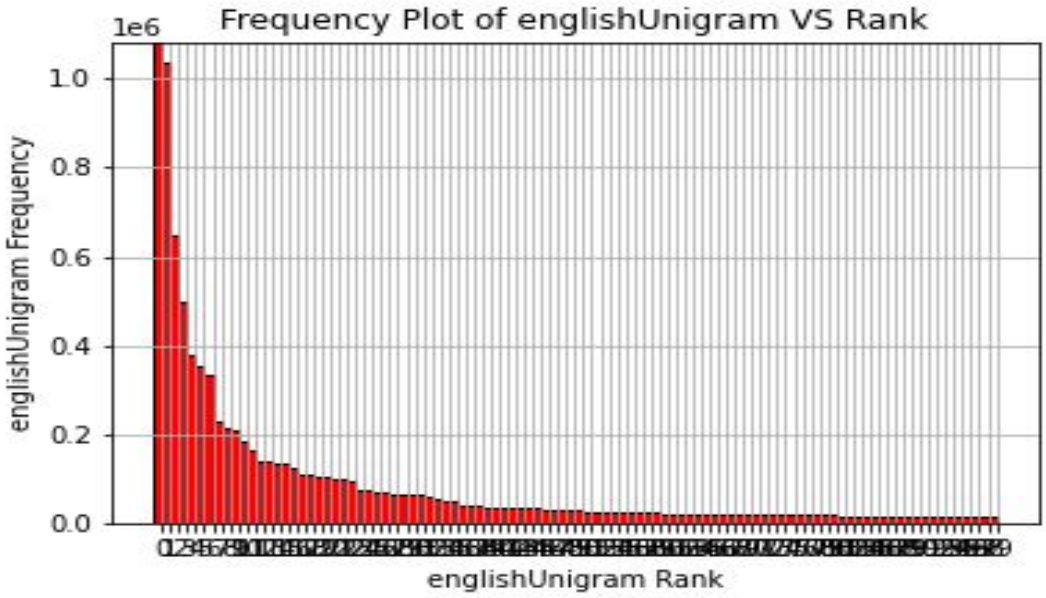
  Total number of input token words - **20372526**

  Total number of unigrams - **20372526**

Total number of distinct unigrams - **286686**

Five most frequent unigrams and frequency distribution curve -

```
[('the', 1084635), (',', 994858), ('.', 862133), ('of', 646923), ('and', 499999)]
```



- **Hindi**

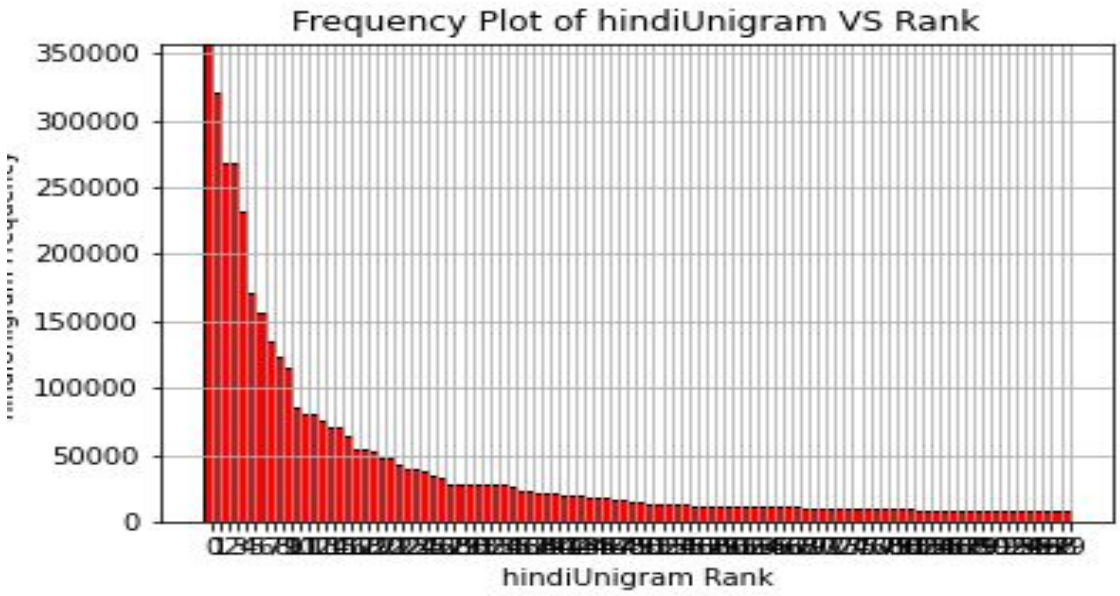
  Total number of input token words - **8640033**

  Total number of unigrams - **8640033**

  Total number of distinct unigrams - **322350**

  Five most frequent unigrams and frequency distribution curve -

```
[('के', 357833), ('।', 321526), ('में', 268249), (',', 267743), ('है', 232156)]
```



3. Statistical Analysis for Bigrams

- **English**

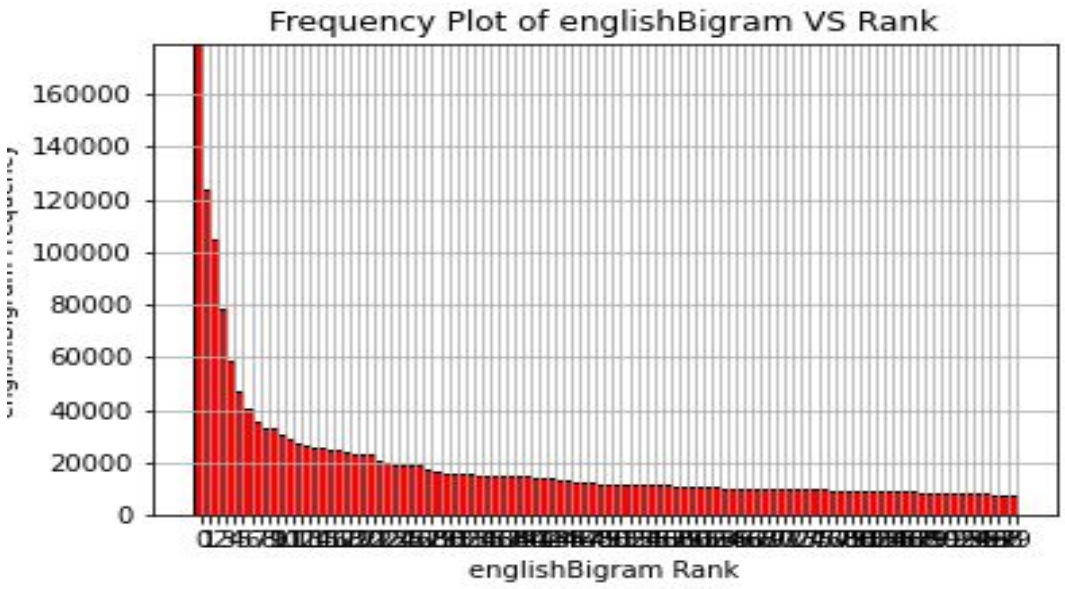
    Total number of input token words - **20372526**

    Total number of bigrams - **20372525**

    Total number of distinct bigrams - **3977129**

    Five most frequent bigrams and frequency distribution curve -

```
[('of the', 179431), ('. The', 140543), (', and', 114438), ("' s", 105543), ('in the', 104836)]
```



- **Hindi**

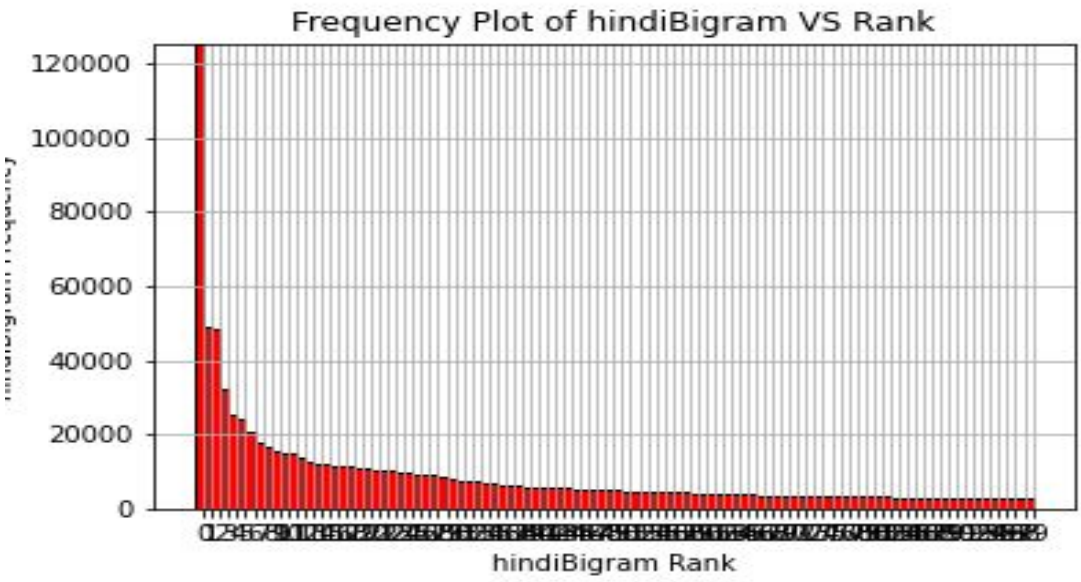
    Total number of input token words - **8640033**

    Total number of bigrams - **8640032**

    Total number of distinct bigrams - **2392979**

    Five most frequent bigrams and frequency distribution curve -

```
[('है ।', 125369), ('के लिए', 49250), ('हैं ।', 48556), ('है ,', 32259), ('जाता है', 25250)]
```



4. Statistical Analysis for Trigrams

- **English**

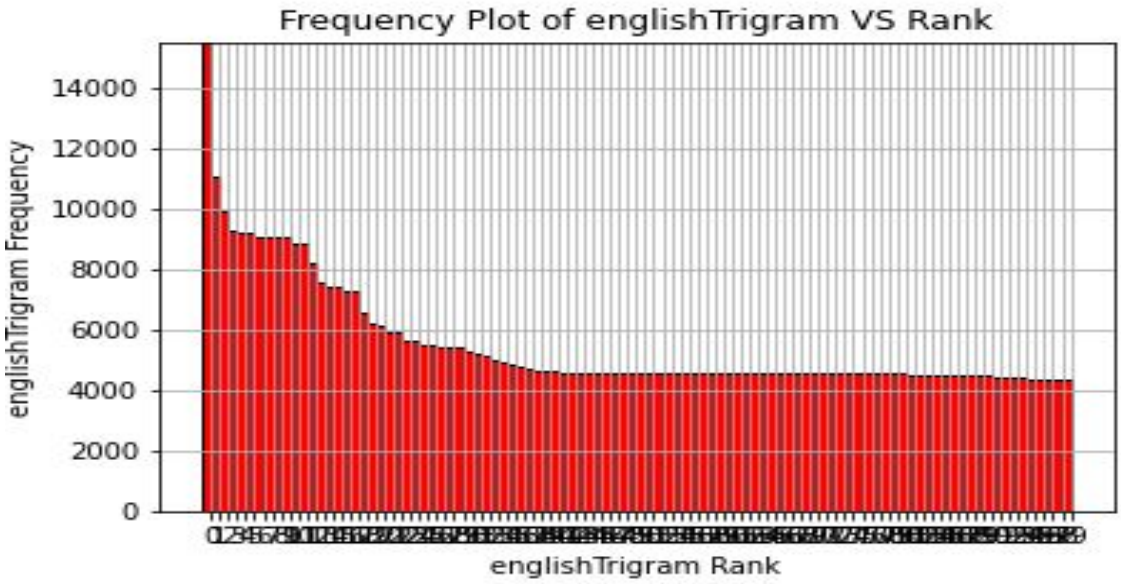
  Total number of input token words - **20372526**

  Total number of trigrams - **20372524**

  Total number of distinct trigrams - **10878682**

  Five most frequent trigrams and frequency distribution curve -

```
[(', and the', 14397), ('. In the', 13244), ('the age of', 11064), ('under the age', 9928), ('the United States', 9467)]
```



- **Hindi**

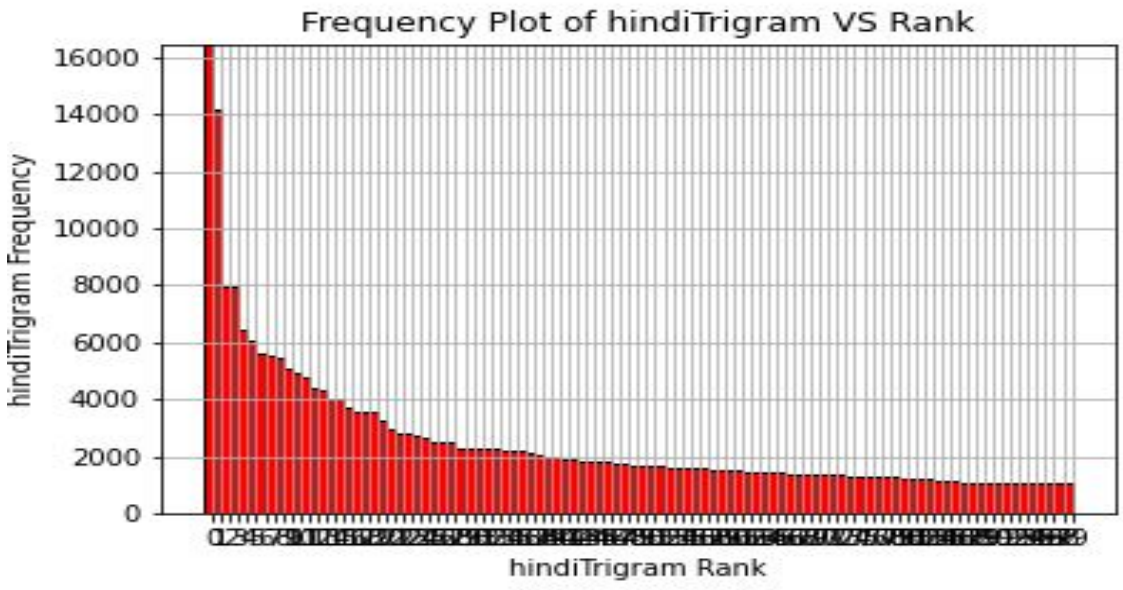
  Total number of input token words - **8640033**

  Total number of trigrams - **8640031**

Total number of distinct trigrams - **5508737**

Five most frequent trigrams and frequency distribution curve -

```
[('के रूप में', 16475), ('जाता है ।', 14139), ('करने के लिए', 7987), ('होता है ।', 7927),
('किया जाता है', 6457)]
```



5. Overall Analysis -

Sentence Segmentation

- **NLTK Sentence Tokenizer:** This tokenizer divides a text into a list of sentences, by using an unsupervised algorithm to build a model for abbreviation words, collocations and words that start sentences. It must be trained on a large collection of plaintext in the target language before it can be used.
- **Regex Sentence Tokenizer:** Here we perform the sentence tokenization based on the regex expression as defined in the Google colab notebook. Here the regex is constructed in a way such that sentence does not break at Dr. and Mr. and hence provides for better tokenization.
- **Stanza Sentence Tokenizer:** Tokenization and sentence segmentation in Stanza are jointly performed by the TokenizeProcessor. This processor splits the raw input text into tokens and sentences, so that downstream annotation can happen at the sentence level.

Word Tokenization

- **Treebank Word Tokenizer:** The Treebank tokenizer uses regular expressions to tokenize text. It assumes that text has already been split into sentences. It splits common English contractions eg. don't is tokenized into do n't. It handles punctuation characters as separate tokens and splits commas and single quotes off from words, when they are followed by whitespace. It also splits off periods that occur at the end of the sentence.
- **Word Punct Tokenizer:** With this method we are able to extract the tokens from a string of words or sentences in the form of Alphabetic or Non-Alphabetic character. It is based on a simple regex tokenization. It used the regular expression **\w+|[^\w\s]+** to split the input.

- **Indic Trivial Word Tokenizer:** This is a NLP library for common text processing in Indian Languages. As Indian languages share a lot of similarity in terms of script, phonology, language syntax and all, this library is an attempt to provide a general solution to very commonly required tool sets for Indian language text. This just tokenizes the punctuation boundaries which also includes purna virama and deergha virama.

Difference in results while using different methods for both tasks -

- **English**: The Punkt Sentence tokenizer uses a **trained unsupervised model** as opposed to the regular based tokenizer which uses a simple heuristic to split the given text into sentences. Hence, we obtain **more** number of sentences splitted while using **Punkt Sentence tokenizer**.
- **Hindi:** Here also, the stanza sentence tokenizer uses a **neural based model** to split sentences whereas the indicnlp uses a rule-based model. Hence we get more sentence splits with the former one.

Zipf's Law

- According to **Zipf's Law**, the relationship between frequency f of word type and its rank r based on frequency is given as:

$$f \propto \frac{1}{r}$$

- This corresponds to the equation of a hyperbola in geometry (where k is a constant) -

$$f.r = k$$

- The frequency vs rank graphs obtained by the frequency distribution of the ngrams also resemble a hyperbola closely. The resemblance is **more strict in case of unigrams**, and **skewed in case of trigrams**. This is because trigrams enhance the effect of uniqueness because several words are joined together (e.g. three words such as 'this', 'of' and 'and' can form six types of trigrams).
- Thus, the frequency remains constant for specific ranges of rank forming groups, as shown in the graph of English trigrams.
- Also, **Zipf's law** is not an exact law, but **a statistical law** and therefore does not hold exactly but only on average (for most words).

TASK 1.3.2 - Few Basic Questions

Analysis Before performing Stemming -

Total 90% Coverage	English	Hindi
Count of unique unigrams	286686	322350
Type of unigrams required for coverage	11224	13339

Total 80% Coverage	English	Hindi
Count of unique bigrams	3977129	2392979
Type of bigrams required for coverage	643572	664972

Total 70% Coverage	English	Hindi
Count of unique trigrams	10878682	5508737
Type of trigrams required for coverage	4766924	2916727

Analysis After performing Stemming -

- **Stemming** is basically the process of producing morphological variants of a root/base word.
- For **English**, stemming is done with the help of Porter Stemmer which is a submodule of the nltk module.
- For **Hindi** we use heuristics based stemmer. The full implementation is given in the Google Colab notebook. Here we define some common suffixes and try to remove that suffix from the words present in the Hindi corpora. As Hindi language follows more of a general rule in terms of suffix as compared to English, hence this way of stemming gives us desired results.

English -

- Total number of stemmed word tokens - **20372526**
- Total number of distinct unigrams after stemming - **200158**

```
[('the', 1270134), (',', 994858), ('.', 862133), ('of', 646927), ('and', 501314)]
```

- Total number of distinct bigrams after stemming - **3184100**

```
[('of the', 179997), ('. the', 140918), (', and', 114466), ("' s", 105543), ('in the', 105047)]
```

- Total number of distinct trigrams after stemming - **10145300**

```
[(', and the', 14485), ('. In the', 13255), ('the age of', 12013), ('age of 18', 10081), ('under the age', 9929)]
```

Hindi -

- Total number of stemmed word tokens - **8640033**
- Total number of distinct unigrams after stemming - **279722**

```
[('के', 357866), ('।', 321526), ('में', 268249), (',', 267743), ('है', 232300)]
```

- Total number of distinct bigrams after stemming - **2117887**

```
[('है ।', 125371), ('के लिए', 49252), ('हैं ।', 48556), ('जा है', 32300), ('है ,', 32263)]
```

- Total number of distinct trigrams after stemming - **5242642**

```
[('जा है ।', 18479), ('के रूप में', 16488), ('हो है ।', 12505), ('कर के लिए', 8091), ('है । यह', 7876)]
```

Summary -

Total 90% Coverage	English	Hindi
Count of unique unigrams	200158	279722
Type of unigrams required for coverage	4499	8033

Total 80% Coverage	English	Hindi
Count of unique bigrams	3184100	2117887
Type of bigrams required for coverage	365889	493541

Total 70% Coverage	English	Hindi
Count of unique trigrams	10145300	5242642
Type of trigrams required for coverage	4033542	2650632

Comparison and Analysis of Results -

English -

	Coverage before stemming		Coverage after stemming	
Ngram (Coverage Percentage)	Reqd. / Total	% of required ngrams	Reqd. / Total	% of required ngrams
unigram (90%)	11224 / 286686	3.91 %	4499 / 200158	2.24 %
bigram (80%)	643572 / 3977129	16.18 %	365889 / 3184100	11.49 %
trigram (70%)	4766924 / 10878682	43.81 %	4033542/ 10145300	39.75 %

Hindi -

	Coverage before stemming	Coverage after stemming

Ngram (Coverage Percentage)	Reqd. / Total	% of required ngrams	Reqd. / Total	% of required ngrams
unigram (90%)	13339 / 322350	4.13 %	8033 / 279722	2.87 %
bigram (80%)	664972 / 2392979	27.78 %	493541 / 2117887	23.30 %
trigram (70%)	2916727 / 5508737	52.94 %	2650632 / 5242642	50.55 %

TASK 1.3.3 - Writing some of your basic codes and comparing with results obtained using tools

1. Heuristics for Sentence Segmentation and Word Tokenization

English

A complex regular expression is used here handling the following cases as described below. After applying this regular expression for sentence segmentation, all the words are converted into lower case for more uniform segmentation.

Example of that is -

The word U.S.A breaks into 'U.S.A' but before heuristics it used to break into 'U' , '.' , 'S' , '.' , 'A'. Thus the total number of tokens have decreased.

```
pattern = r'''(?x)          # set flag to allow verbose regexps
        (?:[A-Z]\.)+        # abbreviations, e.g. U.S.A.
      | (?:\s\w\w\.)+       # Dr. Mr. Ms.
      | \w+(?:-\w+)*        # words with optional internal hyphens
      | \$?\d+(?:\.\d+)?%?  # currency and percentages, e.g. $12.40, 82%
      | \.\.\.              # ellipsis
      | [][.,;"'?():_`-]    # these are separate tokens; includes ], [
    '''
```

- Comparison of Tokens before and after heuristics -
  Count of tokens before heuristics: **20372526**
  Count of tokens after heuristics: **20065018**

Analysis -

	Without Stemming		With Stemming	
Ngram (Coverage Percentage)	**Coverage before Heuristics**	**Coverage after Heuristics**	**Coverage before Heuristics**	**Coverage after Heuristics**
Unigram (90%)	11224	9297	4499	4705
Bigram (80%)	643572	521365	365889	349518

Trigram (70%)	4766924	4463426	4033542	3865865

- As shown above, less number of n grams are required to reach the desired coverage after heuristics. This can be explained as we used a complex regular expression for segmentation.
- Also, after stemming the requirement of n grams for the coverage decreases significantly.
- Also, Coverage before heuristics with stemming achieves better results as compared to coverage after heuristics before stemming. This implies that stemming plays a greater role in segmentation, as all the words are converted to their root words thus increasing the frequencies of words.

Hindi

Here the heuristics is performed by ending a sentence on a purnviram -

```
text.split(u"।")
```

- Comparison of Tokens before and after heuristics -
  Count of tokens before heuristics: **8640033**
  Count of tokens after heuristics: **8158694**

Analysis -

	Without Stemming		**With Stemming**	
Ngram (Coverage Percentage)	**Coverage before Heuristics**	**Coverage after Heuristics**	**Coverage before Heuristics**	**Coverage after Heuristics**
Unigram (90%)	13339	12245	8033	9984
Bigram (80%)	664972	675731	493541	501253
Trigram (70%)	2916727	2909875	2650632	2748961

- As shown above, the results slightly differ with the analysis obtained from the English language. More number of n grams are required to achieve the desired coverage in some cases after heuristics. This is due to the fact, that some complex tokens are formed such as ('है?') which creates a sense of false differentiation between other forms (such as 'है'), thereby increasing the number of n grams required for coverage.
- Here also stemming achieves better results as words are converted to their root form thereby increasing the frequency.

2. Likelihood Ratio Test Implementation and comparison

- **NOTE - The entire implementation of the Likelihood Ratio Test is done in the Google Colab Notebook.**

  English

  - **Comparison of Bigrams using Likelihood Test and Existing Tools -**
    Count of bigrams using inbuilt tools: **3977129**
    Count of bigrams using likelihood test: **491686**
  - Sample Bigram Output achieved using Likelihood Test -

```
['The word', 'word "', '" was', 'was coined', 'coined by', 'ancient Greek', 'Greek philosophers', '. However', 'However ,', ', these', 'these ideas', 'ideas were', 'were founded', 'founded in']
```

  Hindi

  - **Comparison of Bigrams using Likelihood Test and Existing Tools -**
    Count of bigrams using inbuilt tools: **2392979**
    Count of bigrams using likelihood test: **262625**
  - Sample Bigram Output achieved using Likelihood Test -

```
['मास्टर ऑफ़', 'ऑफ़ हेल्थ', 'हेल्थ एडमिनिस्ट्रेशन', 'मास्टर ऑफ', 'हेल्थकेयर एडमिनिस्ट्रेशन', 'एडमिनिस्ट्रेशन (', 'एम .', '. एच', 'एच .', '. ए', 'स्नातकोतर (', 'पोस्ट ग्रेजुएशन', ') की', 'की एक', 'एक पेशेवर', 'है जो', 'स्वास्थ्य प्रशासन', 'प्रशासन के']
```

  Analysis -

- Around (262625/2392979 = **10.97%**) bigrams passed the likelihood test in case of Hindi language while (491686/3977129 = **12.36%**) bigrams passed in case of English. Likelihood ratios are one of the approaches to hypothesis testing. This test is generally more appropriate for **sparse** data. Some of the **passed bigrams** are actually so common in general English or Hindi text like -
- **English** - ['was coined', 'coined by', 'ancient Greek', 'Greek philosophers']
- **Hindi** - ['मास्टर ऑफ़', 'ऑफ़ हेल्थ', 'हेल्थ एडमिनिस्ट्रेशन', 'स्वास्थ्य प्रशासन']

TASK 1.3.4 - Morphological parsing

- Morphological parsing, in natural language processing, is the process of determining the **morphemes** from which a given word is constructed. It must be able to distinguish between orthographic rules and morphological rules. For example, the word **'foxes'** can be decomposed into 'fox' (the stem), and 'es' (a suffix indicating plurality).

English (polyglot)

- **Polyglot** offers trained morfessor models to generate morphemes from words.
- In the Morfessor Baseline method we use raw text as training data in order to learn a model, i.e., a vocabulary of morphs. We construct a morph vocabulary, or a lexicon of morphs, so that it is

possible to form any word in the data by the concatenation of some morphs. This model is inspired by the Minimum Description Length (MDL) principle.

- **Most Frequent Words**

  5 Random words chosen from the 100 most frequent are -

```
['is', 'their', 'States', 'its', 'females']
```

Word	Morphemes
is	['is']
their	['t', 'heir']
States	['State', 's']
its	['it', 's']
females	['female', 's]'

- **Least Frequent Words**

  5 Random words chosen from the 100 least frequent are -

```
['STAT', 'COPY', 'Sportstar', 'lication', 'featural']
```

Word	Morphemes
STAT	['S', 'TA', 'T']
COPY	['CO', 'P', 'Y']
Sportstar	['S', 'port', 'star']
lication	['lic', 'ation']
featural	['feat', 'ural']

Hindi (indicnlp)

- **Indicnlp**: Unsupervised morphological analysers for various Indian languages. Given a word, the analyzer returns the component morphemes. The analyzer can recognize inflectional and derivational morphemes. This also uses a Morfessor Baseline approach.
- **Most Frequent Words**

  5 Random words chosen from the 100 most frequent are -

```
['उन्हें', 'उसके', 'दो', 'की', 'तो']
```

Word	Morphemes
उन्हें	['उन्हें']
उसके	['उसके']
दो	['दो']
की	['की']
तो	['तो']

- **Least Frequent Words**
  5 Random words chosen from the 100 least frequent are -

```
['ऑटोमिक्टिक', 'ऐरिनॉटोकस', 'लिंगनिश्चयन', 'मालकोविच', 'प्लेबाय']
```

Word	Morphemes
ऑटोमिक्टिक	['ऑटो', 'मिक', 'ृटिक']
ऐरिनॉटोकस	['ऐरि', 'नॉट', 'ो', 'कस']
लिंगनिश्चयन	['लिंग', 'निश्चय', 'न']
मालकोविच	['माल', 'कोविच']
प्लेबाय	['प्लेबाय']

TASK 1.3.5 - Sub-word tokenization (Byte Pair Encoding)

- **NOTE - The entire implementation of the Byte Pair Encoding (BPE) is done in the Google Colab Notebook.**

English

- Type of words taken as input: **20372526**
- Number of merges/Vocabulary size = **500**
- Output after byte encoding (only showing 10 entries) -

```
[('e', '</w>'), ('s', '</w>'), ('t', 'h'), ('d', '</w>'), ('n', '</w>'), ('e', 'r'),
('t', '</w>'), ('th', 'e</w>'), (',', '</w>'), ('i', 'n')]
```

- **Most frequent words** (showing 50 entries)

```
[('the</w>', 1084635), (',</w>', 994858), ('.</w>', 862133), ('of</w>', 646923),
('and</w>', 499999), ('in</w>', 379124), ('to</w>', 356109), ('a</w>', 334452),
('"</w>', 308618), ('was</w>', 210438), ('The</w>', 185456), ('-</w>', 167996),
('is</w>', 164277), ("'</w>", 137466), ('as</w>', 135740), ('for</w>', 135273),
('(</w>', 124767), ('with</w>', 122627), ('by</w>', 112181), ('s</w>', 111365),
('that</w>', 111239), ('were</w>', 102633), ('on</w>', 101450), ('%</w>', 97183),
('from</w>', 94091), ('or</w>', 75555), ('his</w>', 72419), ('at</w>', 70801),
('In</w>', 66810), ('are</w>', 65130), ('an</w>', 64192), ('which</w>', 63647),
('had</w>', 58794), ('it</w>', 53681), ('be</w>', 53605), (')</w>', 52856), ('he</w>',
48977), ('has</w>', 40496), ('also</w>', 39104), ('",</w>', 38576), ('not</w>', 38251),
('have</w>', 35905), ('who</w>', 35719), ('age</w>', 35458), ('their</w>', 34618),
('1</w>', 33976), (':</w>', 33077), ('but</w>', 32776), ('first</w>', 31967), ('2</w>',
31521)]
```

- **Least frequent words** (showing 50 entries)

```
[('V and en br in k</w>', 1), ('1 0 4 5 0</w>', 1), ('1 0 4 7 F </w>', 1), ('M ul ti l
ing u al</w>', 1), ('C on S c ri p t</w>', 1), ('el li p tic iti es</w>', 1), ('E
7</w>', 1), ('M 4 9</w>', 1), ('M 5 9</w>', 1), ('M 8 7</w>', 1), ('4 1 2 5</w>', 1),
('S b c </w>', 1), ('S B b </w>', 1), ('M 3 1</w>', 1), ('M 1 0 4</w>', 1), ('M 5 1
a</w>', 1), ('M 9 1</w>', 1), ('M 9 5</w>', 1), ('N G C 1 6 7 2</w>', 1), ('2 5 3
6</w>', 1), ('2 9 0 3</w>', 1), ('S and age</w>', 1), ('M 8 6</w>', 1), ('5 8 6 6</w>',
1), ('S m</w>', 1), ('1 4 2 7 A</w>', 1), ('S T A T </w>', 1), ('u ti l il ty</w>', 1),
('bu il ti n</w>', 1), ('1 K B </w>', 1), ('2 K B </w>', 1), ('3 2 K B </w>', 1), ('l
ic ation</w>', 1), ('D e bu g g ing</w>', 1), ('S mar t K ey</w>', 1), ('C O P Y </w>',
1), ('in iti iz ing</w>', 1), ('te ch i e</w>', 1), ('S h u g ar t</w>', 1), ('D E C sy
st em</w>', 1), ('K il d all s</w>', 1), ('S of t C ard</w>', 1), ('l u g g ab l
es</w>', 1), ('P C W </w>', 1), ('M ul ti pl an</w>', 1), ('T el en g ard</w>', 1), ('G
or ill as</w>', 1), ('H am ur ab i</w>', 1), ('X M O D E M </w>', 1), ('5 1/4 </w>',
1)]
```

- Tokenization on **10 unknown words** and **COMPARISON ANALYSIS**

Word	Byte-Pair-Encoding Output	Morphological Analysis using Libr.
dehydrofreezing	['de', 'h', 'y', 'd', 'ro', 'f', 're', 'e', 'z', 'ing</w>']	['de', 'hydro', 'free', 'z', 'ing']
baronetised	['b', 'ar', 'on', 'e', 'ti', 's', 'ed</w>']	['baron', 'et', 'ised']

negroising	['n', 'eg', 'ro', 'is', 'ing</w>']	['negro', 'ising']
nonconsequence	['n', 'on', 'con', 'se', 'qu', 'ence</w>']	['non', 'con', 'sequence']
autarkically	['a', 'ut', 'ar', 'k', 'ic', 'ally</w>']	['aut', 'ark', 'ical', 'ly']
serendipity	['ser', 'en', 'di', 'p', 'ity</w>']	['s', 'er', 'en', 'dip', 'ity']
gobbledygook	['g', 'ob', 'b', 'le', 'd', 'y', 'g', 'oo', 'k</w>']	['go', 'b', 'ble', 'dy', 'go', 'o', 'k']
scrumptious	['sc', 'r', 'um', 'p', 'ti', 'ous</w>']	['s', 'c', 'rump', 't', 'ious']
agastopia	['ag', 'ast', 'op', 'ia</w>']	['a', 'ga', 'stop', 'ia']
delightful	'd', 'el', 'igh', 't', 'ful','</w>'	'de', 'light', 'ful'

Hindi

- Type of words taken as input: **8640033**
- Number of merges/Vocabulary size = **500**
- Output after byte encoding (only showing 10 entries) -

```
[('े', '</w>'), ('ा', '</w>'), ('ी', '</w>'), ('ं', '</w>'), ('र', '</w>'), ('क', 'े</w>'), ('ृ', 'र'), ('न', '</w>'), ('ह', 'ै'), ('।', '</w>')]
```

- **Most frequent words** (showing 50 entries)

```
[('के</w>', 357833), ('।</w>', 321526), ('में</w>', 268249), (',</w>', 267743), ('है</w>', 232156), ('की</w>', 171779), ('और</w>', 155743), ('से</w>', 134901), ('का</w>', 122948), ('को</w>', 115876), ('हैं</w>', 85787), ('"</w>', 80300), ('एक</w>', 80221), ('-</w>', 75215), (')</w>', 70811), ('(</w>', 70320), ('पर</w>', 64984), ('ने</w>', 54126), ('किया</w>', 53857), ('लिए</w>', 52226), ('भी</w>', 47609), ('.</w>', 47105), ('था</w>', 43005), ('कि</w>', 39996), ('गया</w>', 39123), ('यह</w>', 37774), ('इस</w>', 34164), ('रूप</w>', 32207), ('जाता</w>', 28738), ('ही</w>', 28698), ('जो</w>', 28627),
```

```
('करने</w>', 28440), ('साथ</w>', 28354), ('कर</w>', 28043), ('हो</w>', 27564),
('नहीं</w>', 26563), ('द्वारा</w>', 22877), ('या</w>', 22636), ("'</w>", 21649), ('थे</w>',
21405), ('तथा</w>', 21081), ('अपने</w>', 19993), ('बाद</w>', 19818), ('तक</w>', 19168),
('दिया</w>', 18771), (':</w>', 18034), ('होता</w>', 17543), ('थी</w>', 16868), ('वह</w>',
15920), ('कुछ</w>', 14228)]
```

- **Least frequent words** (showing 50 entries)

```
[('\n \n लि ंग नि श् च य</w>', 1), ('ऐ रि न ॉ ट ो की</w>', 1), ('स् त्र ी जन न</w>',
1), ('थ े लि ओ ट ो की</w>', 1), ('उ भ य जन न</w>', 1), ('डे ं ट रो ट ो की</w>', 1),
('ऐ ं फ़ ि ट ो की</w>', 1), ('\n \n को शि का त त्व</w>', 1), ('\n \n अ र् ध क</w>', 1),
('\n \n त न ू</w>', 1), ('के ं द्र क सू त्र ों</w>', 1), ('\n \n स्व त स् सं से च क</w>', 1),
('ऑ ट ो मि क् ट ि क</w>', 1), ('सि न ै प् सि स</w>', 1), ('अ र् ध के ं द्र कों</w>', 1),
('न ्यू क् लि आ ई</w>', 1), ('फ़ ्यू ज् ह न</w>', 1), ('रे स्ट ि ट ्यू टे ड</w>', 1), ('अं त
र् भा जन</w>', 1), ('ए ंड ो मा इ ट ो सि स</w>', 1), ('\n \n अ मै थ ु नी</w>', 1), ('ऐ
पो मि क् ट ि क</w>', 1), ('ऐ रि न ॉ ट ो क स</w>', 1), ('शे वे</w>', 1), ('ၜ ၀ ၆ </w>',
1), ('ရ ွ ှ ၹ </w>', 1), ('ဿ ိ ု ၮ ၆ </w>', 1), ('s a ? </w>', 1), ('ʃ w è </w>', 1),
('θ a i ? </w>', 1), ('पि छ ले S a o p h a </w>', 1), ('Y a w n g h w e</w>', 1), ('कं
बा स रह ता</w>', 1), ('थ ी री</w>', 1), ('प व ार महा व न् था</w>', 1), ('थ ु द म र ज़
ा</w>', 1), ('न ्य ंग</w>', 1), ('यो व घ वे</w>', 1), ('ह ॉ क में</w>', 1), ('स्था न अ
ब</w>', 1), ('\n \n ओ \n पर</w>', 1), ('\xa0 शे वे</w>', 1), ('T a u n g g y i </w>',
1), ('यो व घ ् वे</w>', 1), ('सो फ ा</w>', 1), ('ह क् कम</w>', 1), ('ह से न वी</w>', 1),
('स ौ फ ा</w>', 1), ('\n \n शे वे</w>', 1), ('स्व त ं</w>', 1)]
```

- Tokenization on **10 unknown words** and **COMPARISON ANALYSIS**

Word	Byte-Pair-Encoding Output	Morphological Analysis using Libr.
संपादन	['सं', 'पा', 'द', 'न</w>']	['संपादन']
दस्ता	['द', 'स्', 'ता</w>']	['दस्ता']
असहिष्णुता	['अ', 'स', 'हि', 'ष्', 'ण', 'ु', 'ता</w>']	['अ', 'सहिष्णु', 'ता']
प्रत्येक	['प्र', 'त्य', 'ेक</w>']	['प्रत्येक']
बंदरगाह	['बं', 'द', 'र', 'गा', 'ह</w>']	['बंदरगाह']
कुलाधिपति	['कु', 'ला', 'ध', 'ि', 'प', 'ति</w>']	['कुला', 'धिपति']
अधिनियम	['अधि', 'निय', 'म</w>']	['अधिनियम']

आवेग	['आ', 'वे', 'ग</w>']	['आवेग']
अभियांत्रिकी	['अ', 'भ', 'िया', 'ं', 'त्र', 'ि', 'की</w>']	['अभि', 'यांत्रिक', 'ी']
वैतरणी	['वै', 'तर', 'ण', 'ी</w>']	['वै', 'तरणी']

Comparison Analysis [BPE Algorithm VS Morphological Parsing using Inbuilt Tools] -

RESULT - In both English and Hindi, BPE algorithm provides us with **more number of morphemes** as compared to morphological parsing performed using inbuilt tools.

- One thing that we could do is to increase the training data set and number of merges to ensure better merging of the given words in the BPE algorithm.
- Also, The BPE is a more trivial algorithm than the Morfessor-based unsupervised model which is also trained on huge amounts of data, thus providing better results and a more complete morphological analysis than the BPE Algorithm.
- However one of the main **advantages of the BPE algorithm** is that this is **language independent**, and we can train almost any language using this algorithm. Also it handles the Out of the Vocabulary (OOV) words quite well compared to other sentence or word tokenization algorithms.
