CS565: Assignment 1

Name: Arpit Gupta

Roll No.: 170101012

Notebook: https://colab.research.google.com/drive/1goupGZ3xqmJa27nNgnpgLxdTZvXurKDL?usp=sharing

All the results are obtained after using **50%** of the given corpus in both English and Hindi language.

Task 1.3.1

- **Tools Available**: Various options are available for sentence segmentation and word tokenization, basically all the NLP libraries can do this.
    - For English corpus I used NLTK and spaCy libraries.
    - For Hindi corpus I used indicNLP and spaCy libraries.

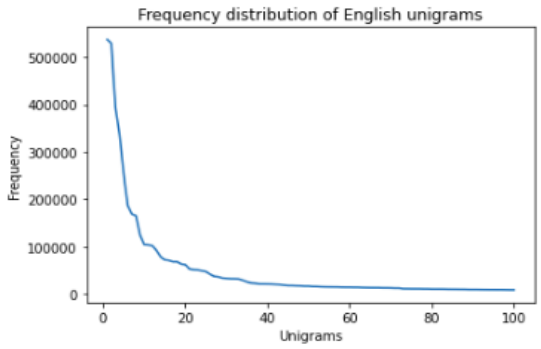
- **Results (1.3.1.1):** The results obtained and some differences observed are:
    - For English corpus:
        - Using NLTK total number of sentences are 397072 and number of tokens are 9936739.
        - Using spaCy total number of sentences are 394417 and number of tokens are 10255199.
        - **NLTK has more sentences as it is sometimes also using comma (,) as a delimiter**. **NLTK has more tokens because it tokenizes "word," into 2 tokens (I.e. word and comma) whereas spaCy into 1.**

    - For Hindi corpus:
        - Using indicNLP total number of sentences are 175117 and total number of words are 4315181.
        - Using spaCy total number of sentences are 178911 and total number of words are 4356434.
        - **SpaCy has more sentences because indicNLP is more intelligent in terms of handling abbreviations, for e.g. is a sentence ends with बी.टेक. then spacy will consider this to be end of sentence whereas indicNLP not. Spacy also has more tokens because it breaks abbreviations on the basis of period(.) and also breaks negative numbers into sign and number.**

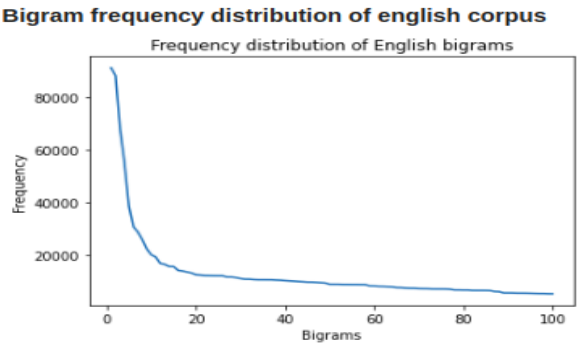
- The graphs for frequency distribution **(Frequency vs rank)** of all n-grams for both English and Hindi are given below. All the graphs show a similar distribution that some words are present in high frequency then and the rest of them are in low frequency, so the graph decreases sharply and looks like a straight line after some words.
    - The **most frequent unigrams** in both languages are **determiners for e.g. 'the' in English and 'के' in Hindi.** Other most frequent unigrams are **punctuations** like ",", "." in English and "|" in Hindi.

- **Least frequent unigrams** are tokens like "**every", "most"** in English corpus and **"क्षेत्र"** in Hindi corpus.
- **Most frequent bigrams** in both languages are again delimiters and punctuations often used together like **"of the"** in English and **"है ।"** in Hindi.
- **Least frequent bigrams** in both languages generally consist of least unigrams in respective languages like **"into time"** in English and **"क्षेत्र में"** in Hindi.
- For the **trigrams** the analysis is same as that of bigrams but one interesting thing to note here is that most of the trigrams (most frequent) consist of token **"age"** which is because of the type of corpus given.

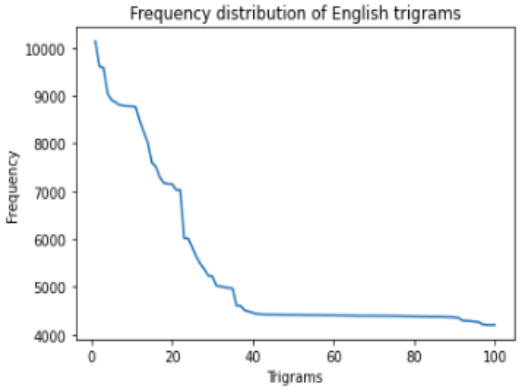
Unigram frequency distribution of english corpus



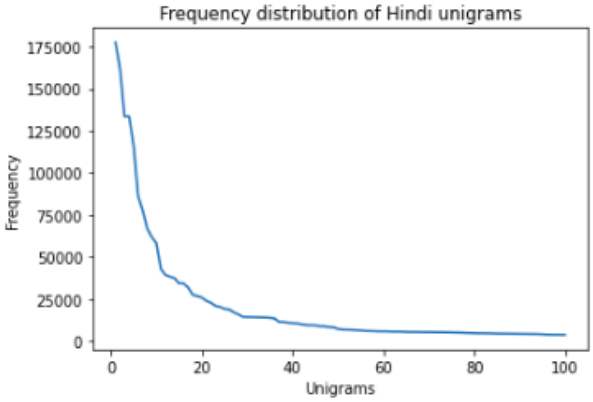
Bigram frequency distribution of english corpus



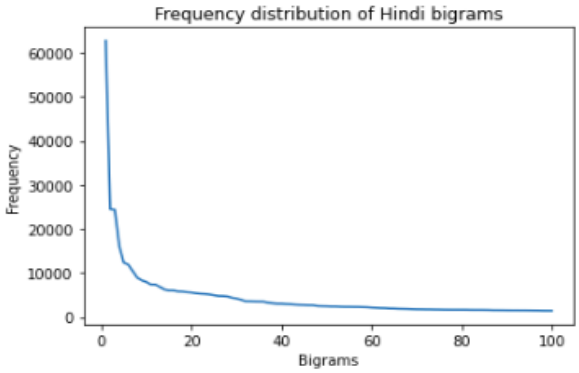
Trigram frequency distribution of english corpus



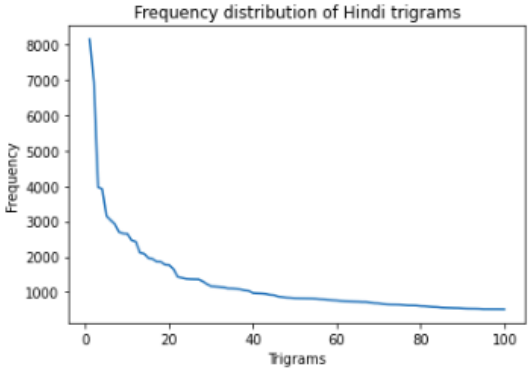
Unigram frequency distribution of hindi corpus



Bigram frequency distribution of hindi corpus



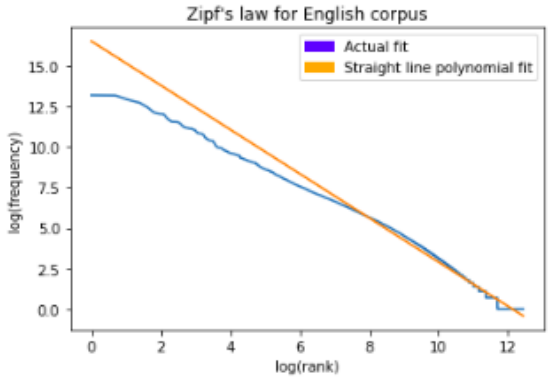
Trigram frequency distribution of hindi corpus



- According to Zipf's law equation the frequency of token and its rank (when all tokens are sorted in decreasing order) are inversely proportional. So basically **frequency*rank** should approximately be constant I.e. **log(frequency) vs log(rank) should produce a straight line** which is basically what we obtain in our graphs.
    - The graphs show the actual logarithmic fit and the corresponding straight line (produced by polynomial fit). We can also see that the slopes of straight lines are close to 1.
    - Also, as we know that Zipf's law is not precisely correct at the end points which can also be seen from out graph. **The parameters of curve fitting I.e. the slope and intercept** are given in the graph:

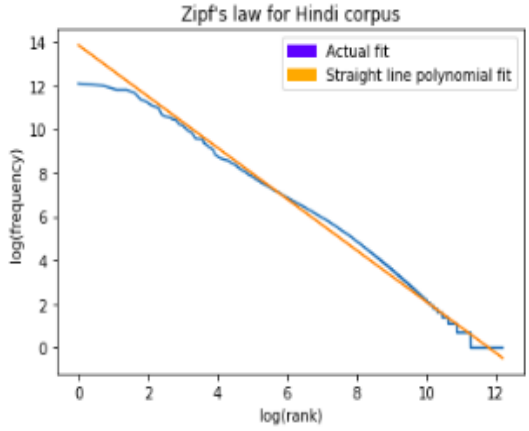
Zipf's Law(English Corpus)

Approximate straight line has slope = -1.3632912841002662 and y-intercept = 16.52330873933697



Zipf's Law(Hindi Corpus)

Approximate straight line has slope = -1.1737064184389572 and y-intercept = 13.846670757909154



Task 1.3.2

- For stemming I used Porter Stemmer from NLTK and for Hindi I referred to the suffix stripping algorithm given in paper (reference 1 and reference 2).
- The total number of n-grams before and after stemming will not change because stemming in English only converts the token into its root word and in Hindi it only removes the matras ("ो", "े", "ू", "ु", "ी", "ि).
- Although the type of tokens will certainly will decrease as all words are converted into their root forms like “philosophers” and “philosopher” have been converted to “philosopher” and in Hindi like “संगठनों” and “संगठन” will be changed to “संगठन”.
- This fact can be seen from the required number of n-grams for a particular coverage decreases in all three cases after stemming in both the corpora.
- We can also see that we require more bigrams and trigrams for less coverage than unigrams, this is because trigrams and bigrams have more chance to be unique than unigrams as they are formed from multiple tokens.
- **Results**:

English	Unigrams (90% coverage)	Bigrams (80% coverage)	Trigrams (70% coverage)
Before stemming	12339	490199	2509699
After stemming	5301	300497	2208333
Hindi	Unigrams (90% coverage)	Bigrams (80% coverage)	Trigrams (70% coverage)
Before stemming	12641	523386	1654558
After stemming	7656	371517	1530240

Task 1.3.3

1. **Heuristics:**
    - **English:**
        - For **sentence segmentation** I used delimiters like “.”, “?” and “!” as the marker for end of sentence. If the marker is a period then we have to check if the preceding text is an abbreviation or not. If not, then this marks the end of sentence. For “?” and “!” we check the next word, if it has lower case as first letter then we continue otherwise this marks the end of sentence.
        - For **tokenization** any space (\n, \t, ‘ ‘) marks the end of token. In that token we check for last character which can be a punctuation mark, if a period is present and preceding text is an abbreviation then it is counted as 1 token otherwise in every case it is counted as 2 tokens. Since we are only checking last character so single/double quotes or hyphens which come in the middle of word are not counted as a token.
        - **Results:**
            - Total number of sentences: 397020
            - Total number of tokens: 9802615

- We can clearly see that there is not difference between the heuristics I used and the one's that tools (like NLTK or spaCy) provide; thus, our heuristics are working fine.

English (Heuristics)	Unigrams (90% coverage)	Bigrams (80% coverage)	Trigrams (70% coverage)
Before stemming	15060	440654	2557858
After stemming	6619	345965	2282512

- **Hindi:**
  - For **sentence segmentation** method similar to the one used in English is followed. Three punctuation marks ("।", "?", "!") are used to mark the end of sentence.
  - For **tokenization** tokens are separated by spaces and all the rules same as that in English heuristic are followed.
  - **Results:**
    - Total number of sentences: 159512
    - Total number of tokens: 6068517
    - Just as in English corpora here also the results are nearby the ones, we get in libraries of industrial level strength like spaCy.

Hindi	Unigrams (90% coverage)	Bigrams (80% coverage)	Trigrams (70% coverage)
Before stemming	5556	106844	507463
After stemming	4224	91892	461750

2. **Likelihood Ratio test** is used for ranking collocations or bigrams. It is simply a number which tells us how much more likely one hypothesis is than the other where the two hypotheses are:

- Hypothesis 1: Occurrence of second token in bigram is independent of the previous occurrence of first token. ($P(w^2|w^1) = p = P(w^2|\neg w^1)$)
- Hypothesis 2: Formalization of dependence of first and second tokens thus providing a good evidence for an interesting collocation. ($P(w^2|w^1) = p \neq P(w^2|\neg w^1)$)

So, we basically the ratio of the log of two hypotheses and then rank the bigrams on its basis. Log Likelihood ratio is given by $\mathbf{log\lambda = log(L(H^1)/L(H^2))}$

Some of the top collocations obtained are "United States" and "median income" in English corpus which proves the working of this method and some collocations in Hindi are "माध्यम से".

Since this method is purely mathematical so we also bigrams which cannot be termed as collocations in top ten for e.g. "The ." in English and "है ।" in Hindi which are actually punctuation marks and determiners.

Task 1.3.4

The library used for morphological analysis is **polyglot**. Polyglot offers trained morfessor models to generate morphemes from words, its goal is to develop unsupervised data-driven methods that discover the regularities behind word forming in natural languages.

Some common terms before taking a deep dive into model are:

- Atoms: The smallest pieces of text that the algorithm processes.
- Compounds: The set of sequences of atoms.
- Construction: Lexical units between atoms and compounds.
- Tokenization: The operation of replacing a compound with its construction.
- Detokenization: The reverse process of tokenization.

The model is defined as follows:

- It consists of two parts: a lexicon and a grammar.
- The lexicon stores the properties of constructions and the grammar determines how the constructions can be combined to form the compounds.
- The grammar has two basic assumptions: 1) compound consists of one or more constructions. 2) construction of a compound occurs independently.
- The cost function is derived from the maximum a posteriori (MAP) estimate of the model.
- The cost function has two parts model likelihood and prior (determines the probability of the model lexicon).
- Likelihood derives from assumptions above and prior from the minimum description length principle.

For the morphological analysis we give the morphemes and the POS-TAG (provided by NLTK) as the output for each token.

For e.g.:

- English: **including ---> [‘include’, ‘ing’] as the morphemes and VBG (verb in present tense)** as the POS-TAG
- Hindi: **राज्वी ---> ['राज', 'ृव', 'ी'] as the morphemes and JJ (adjective)** as POS-TAG.

Task 1.3.5

Used 10,000 as the vocab size and 5000 as the number of merges for both English and Hindi corpora.

And ‘#’ marks the end of a token.

Byte Pair Encoding:

- Some of the most frequent tokens found after training the vocabulary for English are (‘the#’, ‘and#’, ‘was#’, ‘The#’, ‘were#’)

- Some of the least frequent tokens found after training the vocabulary for English are (‘’vi#’, ‘riscus#’, ‘cerefoli’, ‘thor’)
- Some of the most frequent tokens found after training the vocabulary for Hindi are (‘में#’,'हैं#','किया#’,'लिए#’)
- Some of the least frequent tokens found after training the vocabulary for Hindi are ('आत्मानु','राहु', 'बन', 'वारि')
- Exhaustive list is given in the notebook.

Observations while comparing the morphemes:

- English: Some of the tokens give same results like “no” and “out”. And since the vocabulary is small on test data some of the words are not converted to morphemes completely or broken more than necessary (like landing --> land + in + g).
- Hindi: It is harder to train Hindi vocabulary as we need more data and more merges because of all the matras present in words. But still it does give good results for some tokens like ‘पहले’ which get converted to 'पहल' + 'े' and some of the words don’t give any morphemes because they could find any match in the trained tokens like ‘(‘