Report for Assignment 1

Student ID: 2018HT13143

Student Name: Wilfred Roshan Lobo

Text corpus file used is "**wiki_05**"

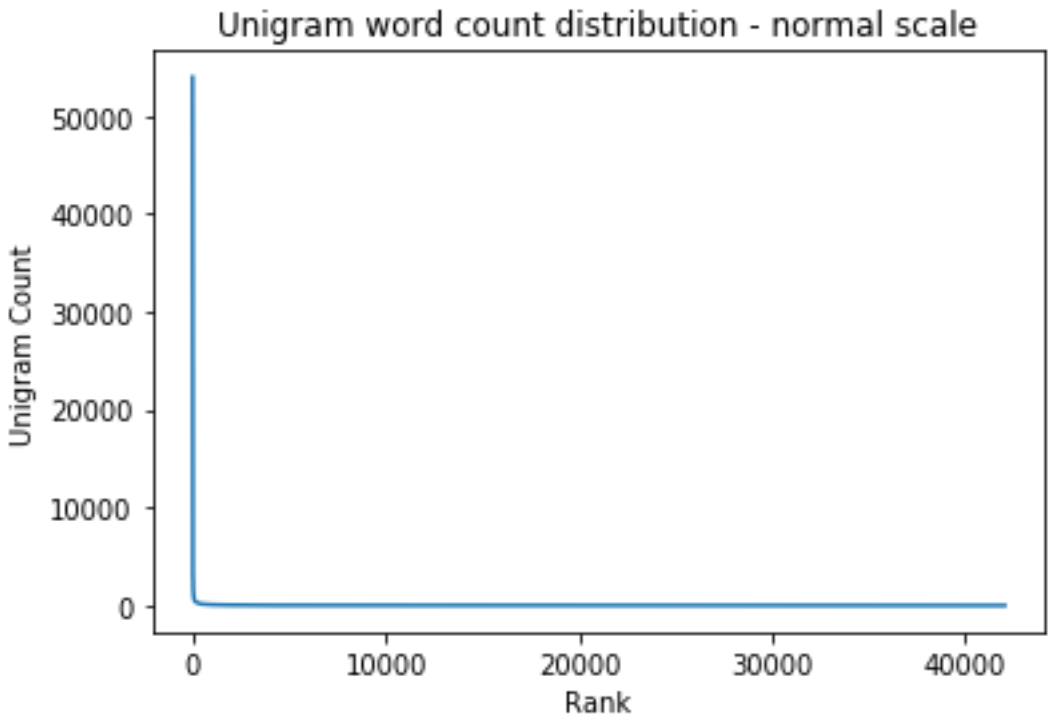
Implementation details:

- Code is developed using python on jupyter notebook.
- Python Code file name for Part-1: 2018HT13143_Part-1.ipynb
- Python Code file name for Part-2A (index construction): 2018HT13143_Part-2-IndexingCreation-SM.ipynb
- Python Code file name for Part-2B (query): 2018HT13143_Part-2-Querying-SM.ipynb
- All code files should be placed in same folder.
- Data files must be placed in a folder named "**data**" in current folder where notebook files are placed.
- Program stores the index file, vocabulary and document ids in separate file under "**data**" folder.

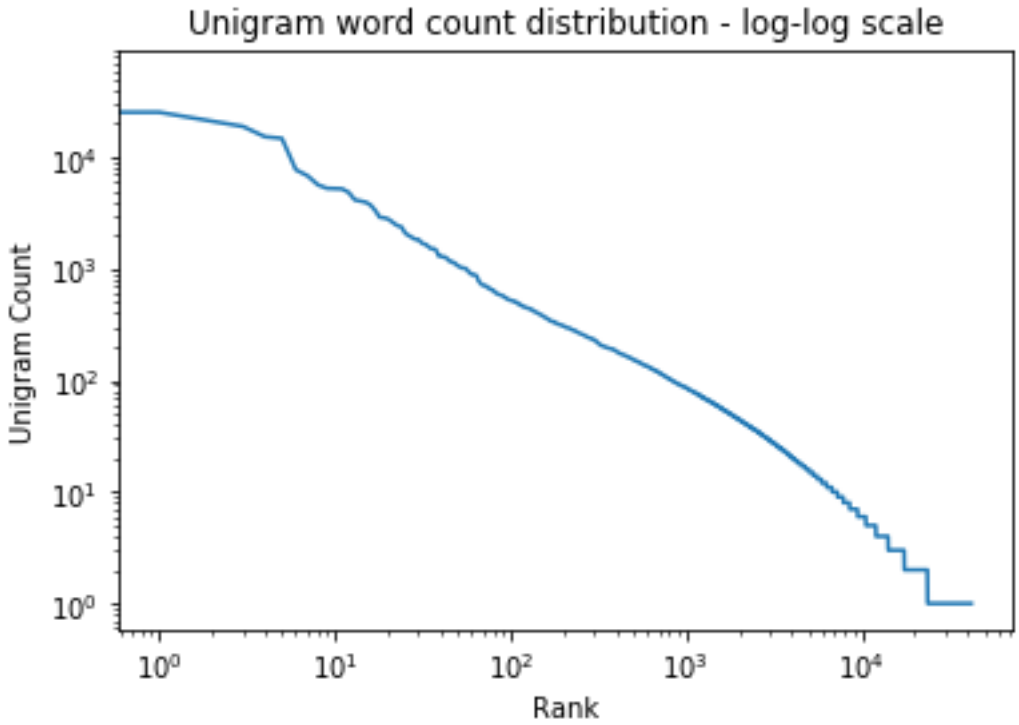
Part 1: Corpus Analysis

Q1. Unigram analysis

- Total unique unigrams present in the corpus: 42040
- Unigram frequency distribution graph – normal scale



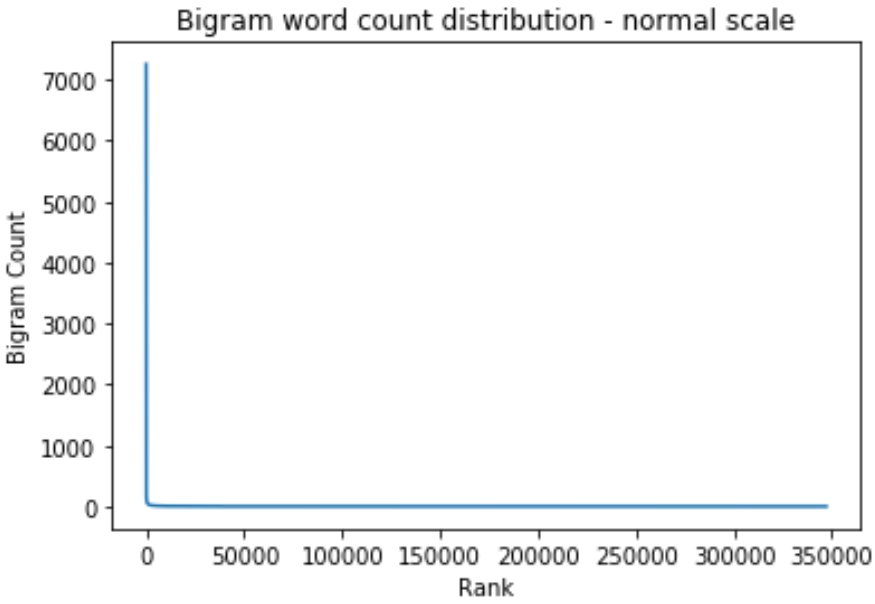
- Unigram frequency distribution graph – log-log scale



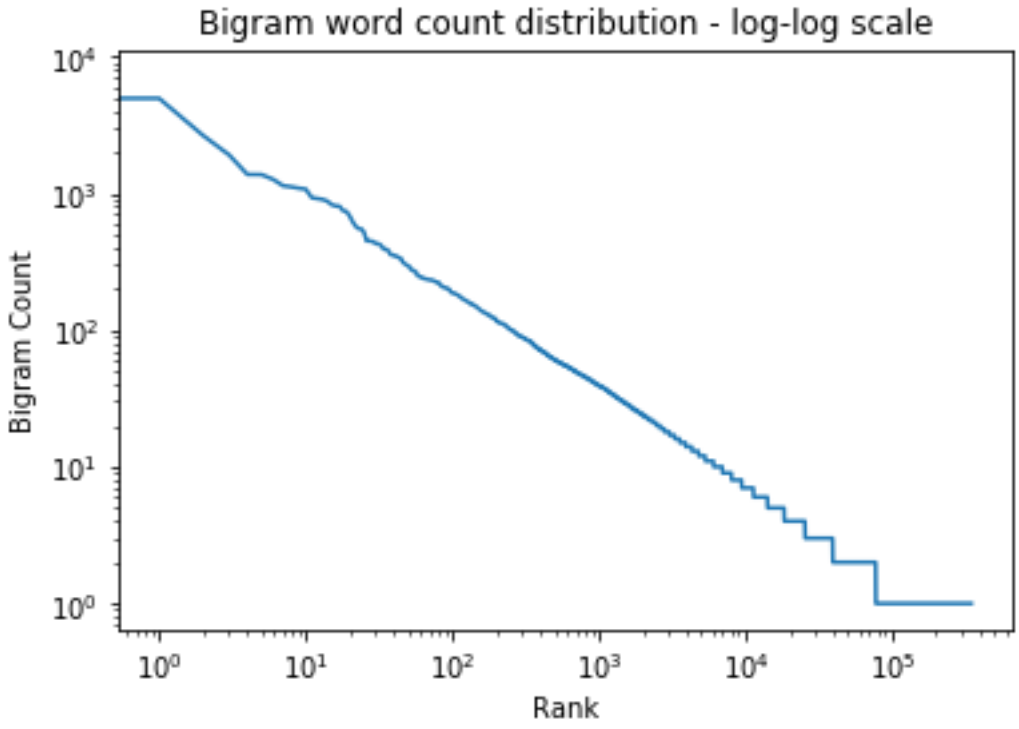
- Number of most frequent unigrams required to cover 90% of corpus are: 8368

Q2. Bigram analysis

- Total number of unique Bigrams present in the corpus: 708044
- Frequency distribution of bigrams on normal scale



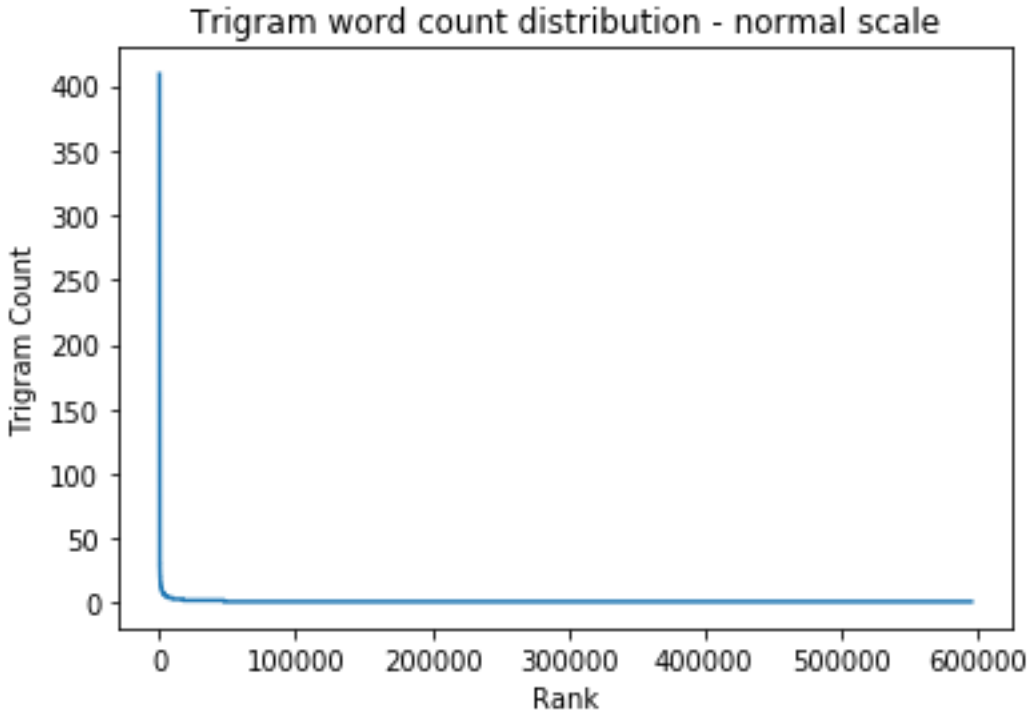
- Frequency distribution of bigrams on log-log scale



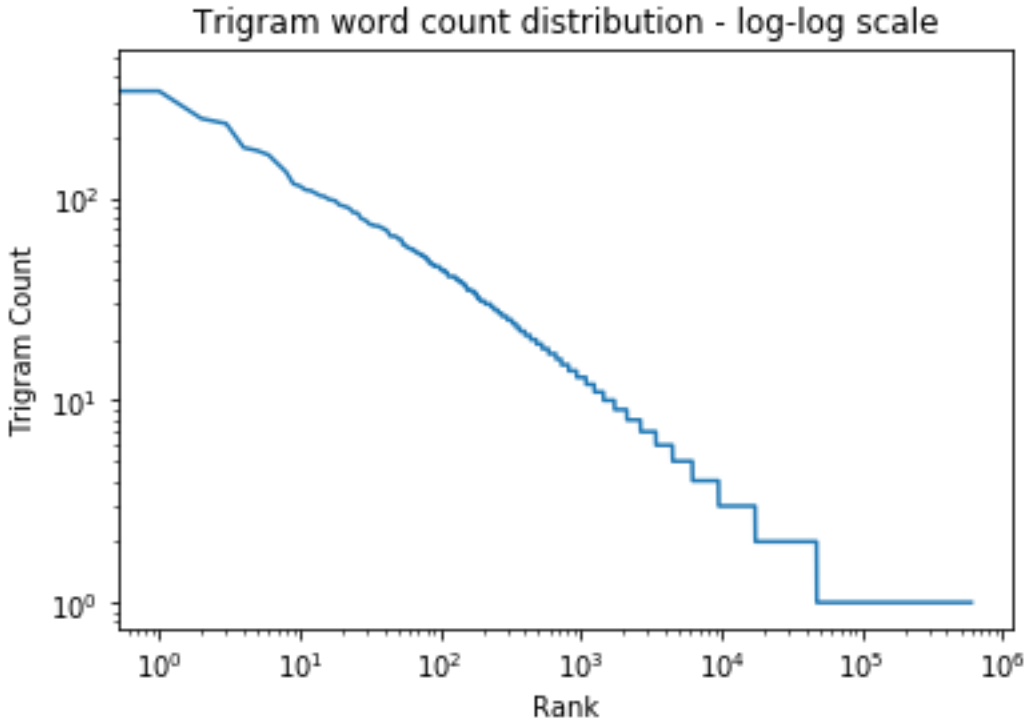
- Number of most frequent bigrams required to cover 90% of corpus are: 276325

Q3. Trigram analysis

- Total number of unique Trigrams present in the corpus: 708043
- Frequency distribution of trigrams on normal scale



- Frequency distribution of trigrams on log-log scale



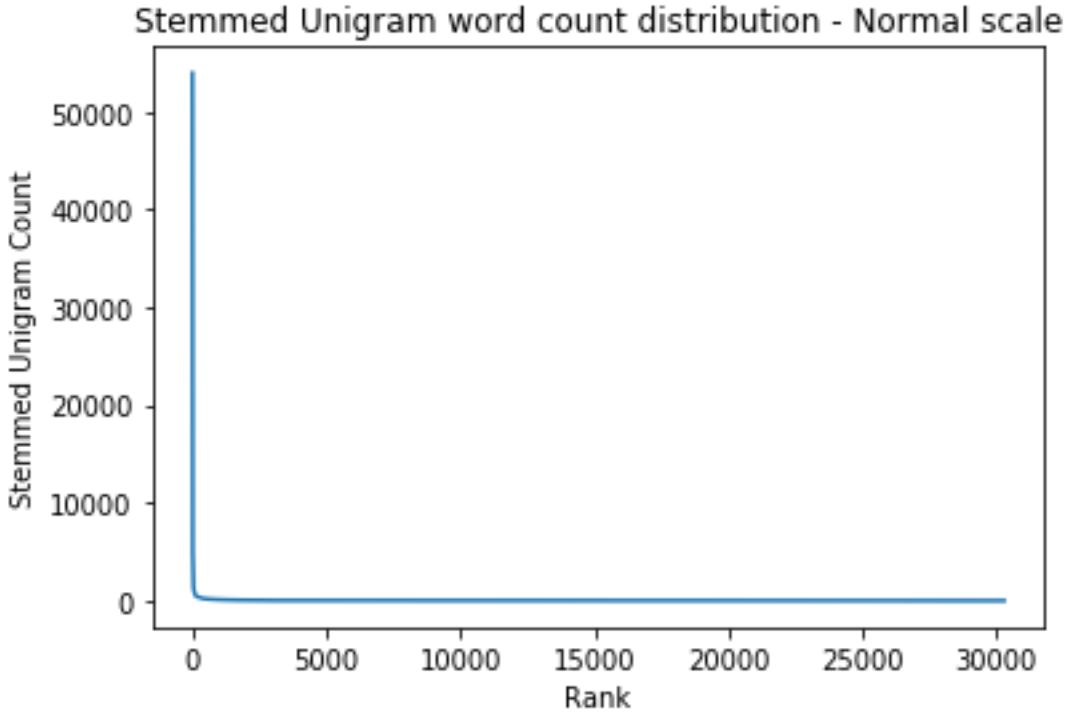
- Number of most frequent trigrams required to cover 90% of corpus are: 525123

Q4. Repeat Q1, Q2 and Q3 after performing stemming process on the tokens.

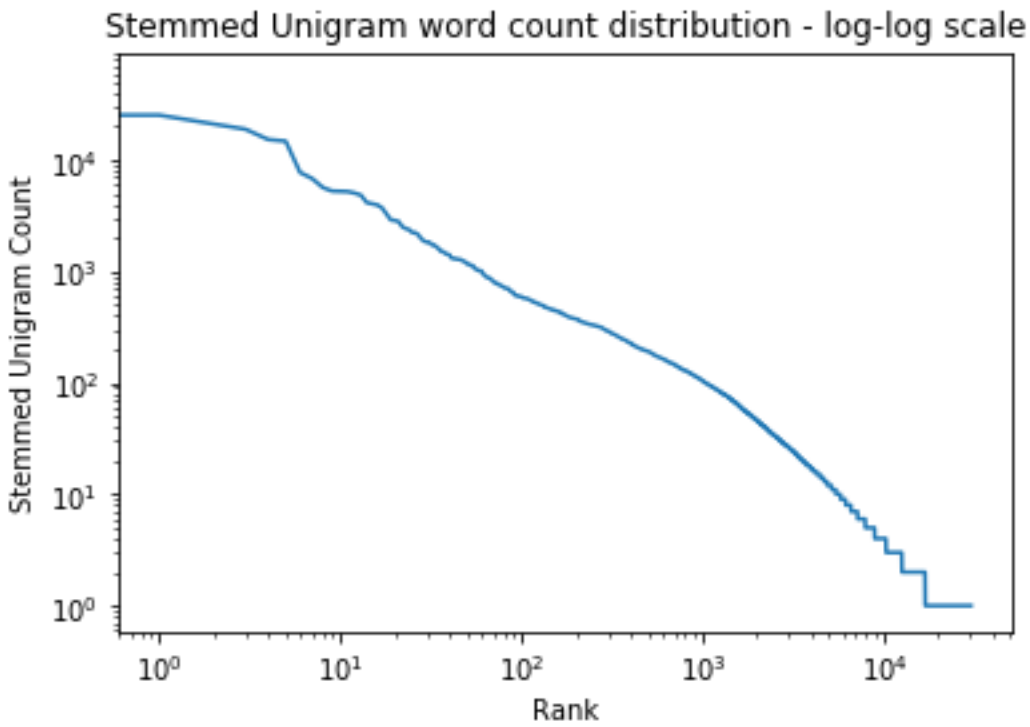
- Stemming is performed using nltk.stem.porter.PorterStemmer.

Unigram analysis after stemming

- Total unique unigrams present in the corpus: 30287
- Unigram frequency distribution graph for stemmed tokens – normal scale



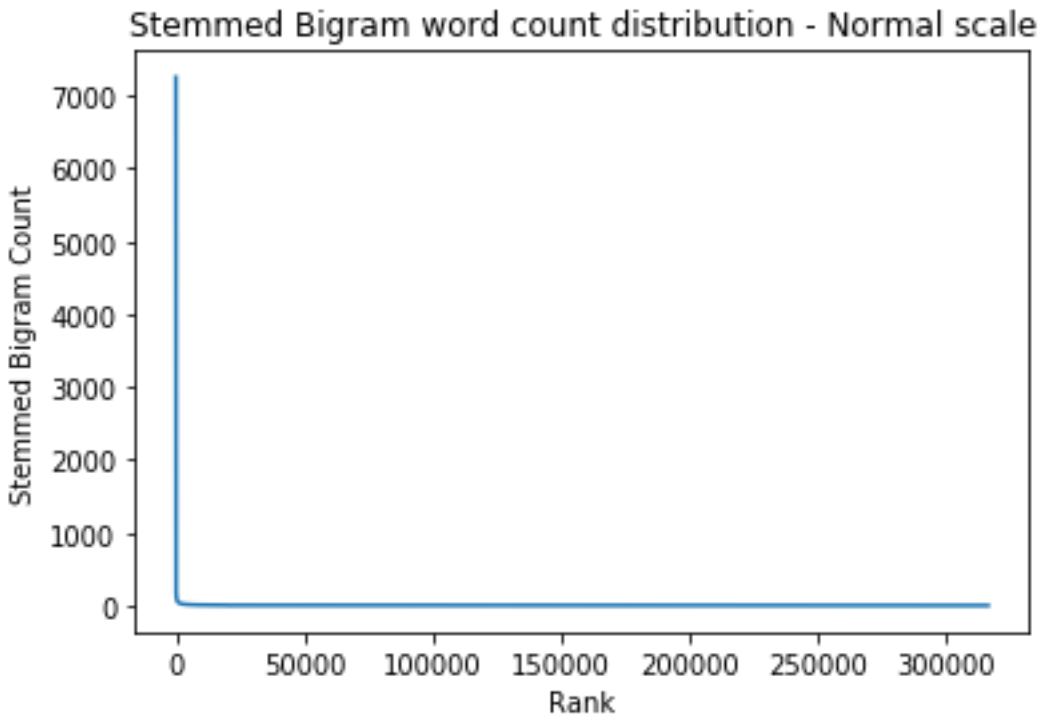
- Unigram frequency distribution graph for stemmed tokens – log-log scale



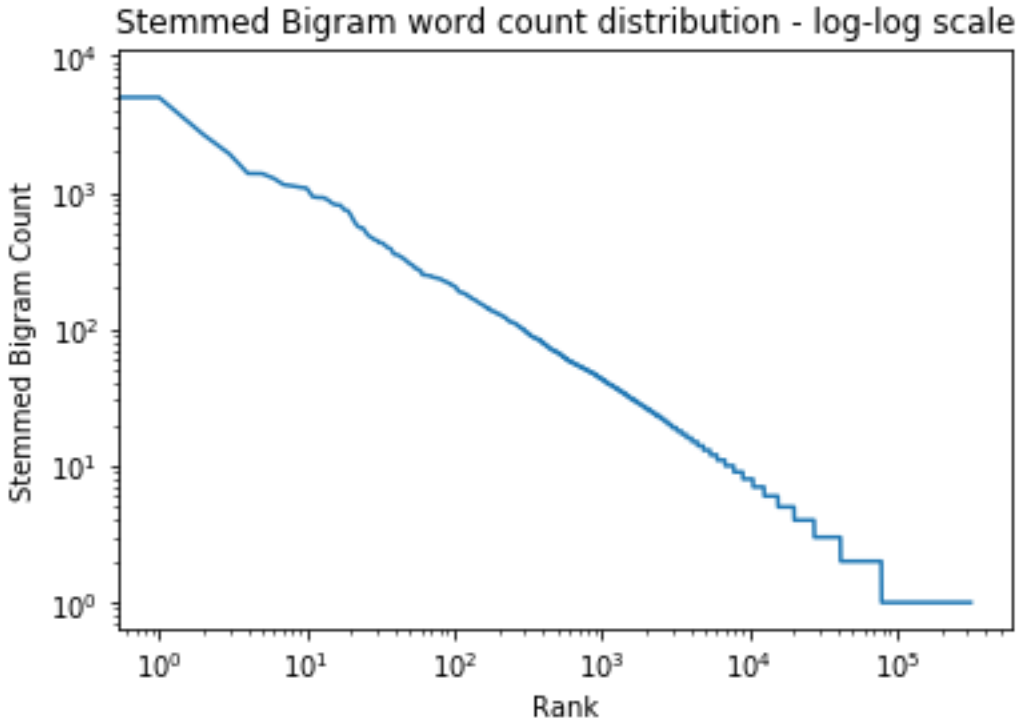
- Number of most frequent unigrams required to cover 90% of corpus are: 4414

Bigram analysis after stemming

- Total number of unique Bigrams present in the corpus: 708044
- Frequency distribution of stemmed token bigrams on normal scale



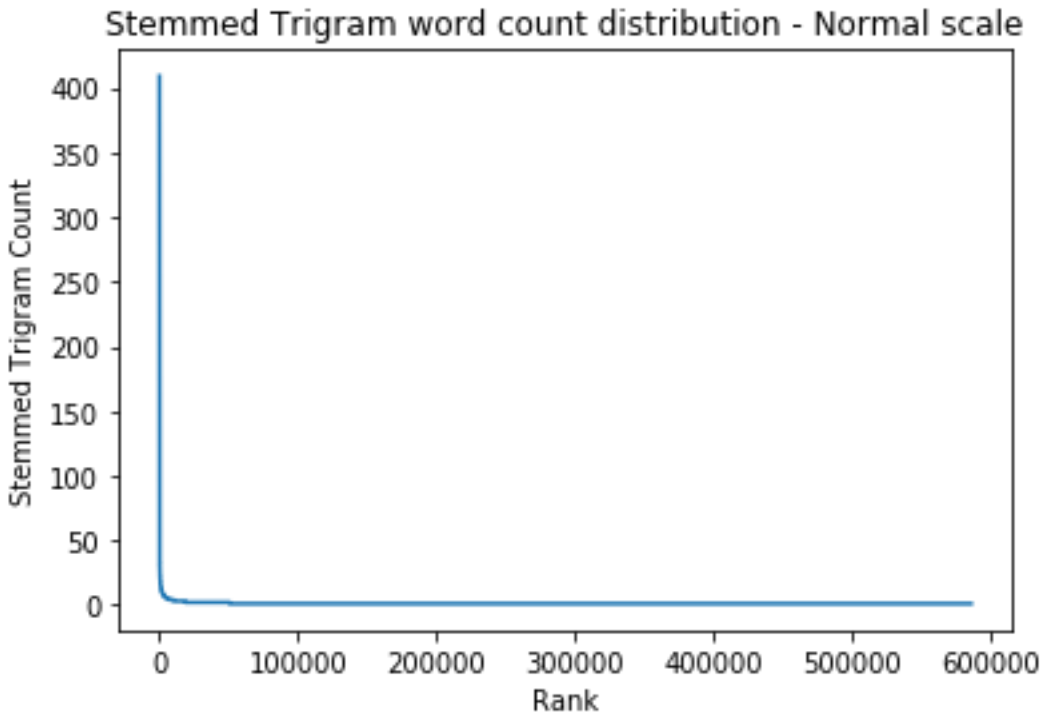
- Frequency distribution of stemmed token bigrams on log-log scale



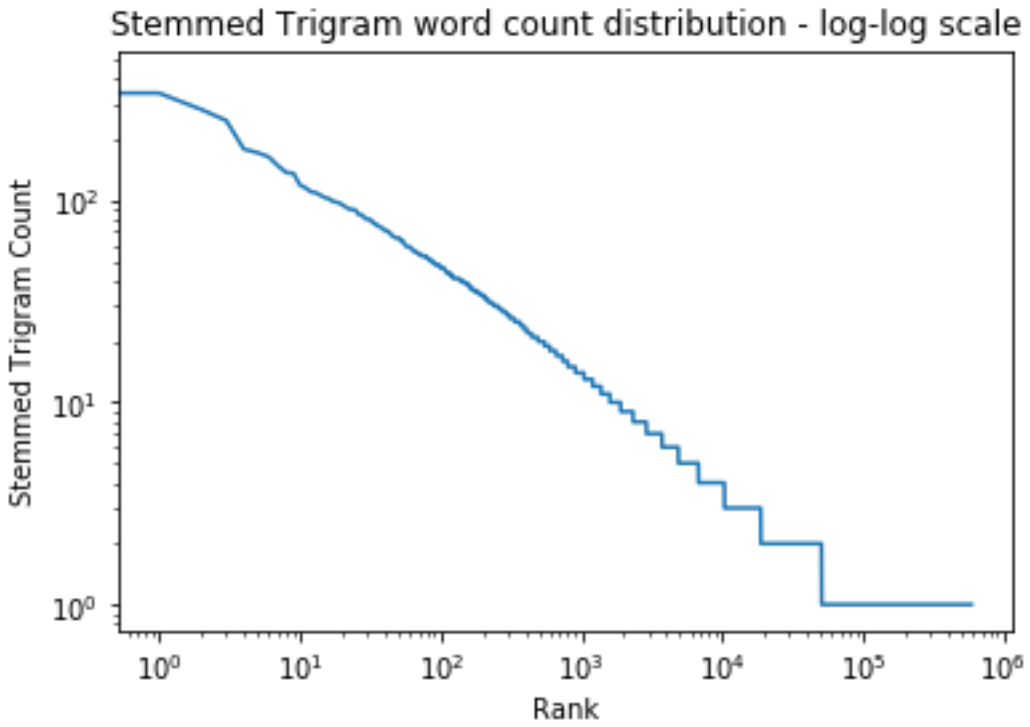
- Number of most frequent bigrams required to cover 90% of corpus are: 246031

Trigram analysis after stemming

- Total number of unique Trigrams present in the corpus: 708043
- Frequency distribution of stemmed word trigrams on normal scale



- Frequency distribution of stemmed word trigrams on log-log scale

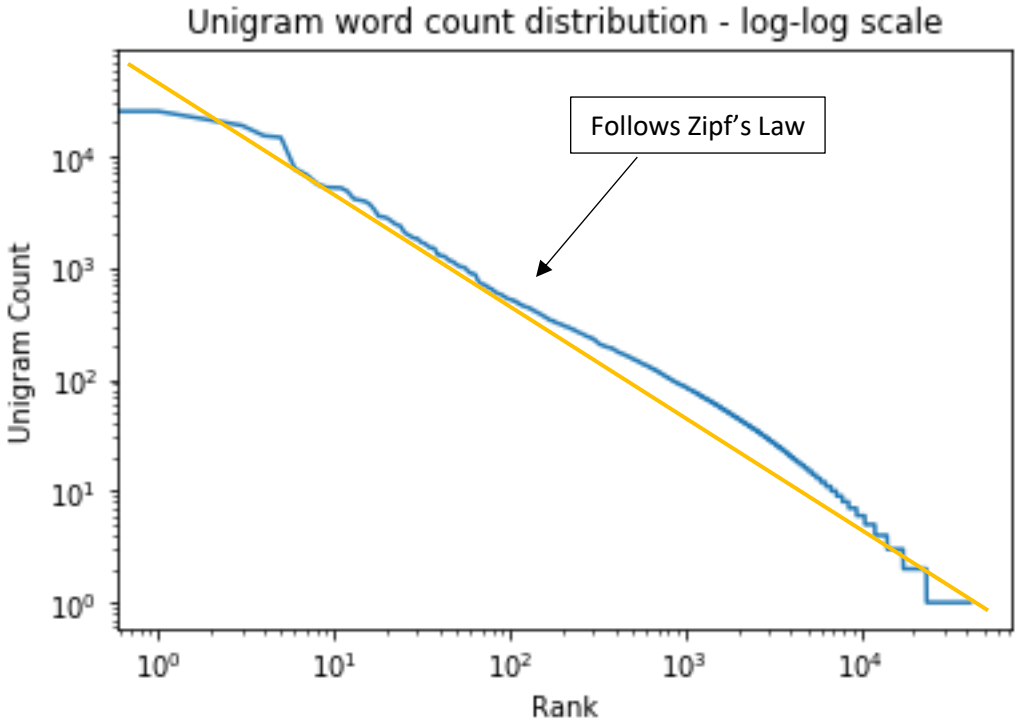


- Number of most frequent trigrams required to cover 90% of corpus are: 515797

Q5. Briefly summarize and discuss the frequency distributed obtained in Q1 to Q4. Do these distributions approximately follow Zipf's law?

The frequency distribution graph of any text corpus will show that there would be some words/tokens that are frequently used and many other(remaining) words/tokens that are infrequently used. The frequency distribution graph is generally forms the L shape when we plot it from most frequent to least frequent. This gives rise to concepts of frequent-words and rare-words for improving search techniques.

When we plot the frequency graph in log-log scale, from most frequent to least frequent, we see that the graph's slope is almost -1. An Example for unigrams is show below.



This follows Zipf's law which state that, if t_1 is the most common term in the collection, t_2 is next most common, and so on, then the collection frequency cf_i of the i^{th} most common term is proportional to 1/i

Q6. What library you used for tokenization and stemming? What were the underlying algorithms used by the library for these tasks?

- For tokenization, **nltk.tokenize** library has a method **word_tokenize(s)**. This is used for tokenization of the text corpus. This method creates tokens out of text corpus by splitting the

text on punctuations (except period). The algorithm is based in regular expression-based selection of group of characters.

- For stemming, I have used **nltk.stem.porter.PorterStemmer**. The **PorterStemmer** has a "stem(s)" method, which will perform stemming on the input term 's'. This algorithm is based to suffix stripping method. The default mode with "NLTK_EXTENSIONS" simply checks if the input term has any one of the knowns suffixes, such as "ied", "ing", "ed", etc., and truncates that suffix. The resulting term may not be a actual dictionary word, but will represent the base form.

Q7. Report three examples based on your observation, where the tool used for tokenization did not tokenize the character sequence properly.

Some of the words that will not tokenize properly are:

- Zipf's -> this will tokenize to 'Zipf', "'s". Notice that, the punctuation-based splitting will cause splitting at apostrophe character.
- on-the-fly -> this will tokenize to 'on-the-fly' without separating the words.
- 90% -> will tokenize to '90' and '%' where are 90% was a single word.

Some of the words that will not stem properly using PorterStemmer are:

- singles -> will stem to 'singl' instead of single
- mingle -> will stem to 'mingl' instead of remaining as is 'mingle'
- this -> will stem to 'thi' instead of remaining as 'this'

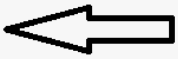
Part2: Vector-space based IR System

- Python Code file name for Part-2A (index construction): 2018HT13143_Part-2-IndexingCreation-SM.ipynb
- Python Code file name for Part-2B (query): 2018HT13143_Part-2-Querying-SM.ipynb
- Value of K = 10 (number of document to output based on most relevance)

Part 2A (index construction):

```
# decalre names of all files
data_file_path = ".\\data\\wiki_05"

# these are the output files
vocabulary_index_file_path = ".\\data\\vocabulary.pkl"
document_index_file_path = ".\\data\\file_index.pkl"
inverted_index_file_path = ".\\data\inverted_index.pkl"
```



How it works (algorithm):

1. Read the wiki file "wiki_05"
2. Extract text content using BeautifulSoup4 library

3. Convert all data to lower case.
4. Count unigrams, bigrams and trigrams using nltk library.
5. Draw graph using matplotlib library.
6. For creating inverted index, construct 3 data structures as follows:
    a. A dict for vocabulary
    b. A dict for document ids
    c. A sparce matrix (scipy.sparce.coo_matrix) for storing term frequencies.
7. Add any {new term, vocabulary_index} to vocabulary dict as they occur. Index is the current length of the vocabulary dict items.
8. Add any {new doc_id, document_index} to document dict as they occur. Index is the current length of the document dict items.
9. Add term count from step 3 to sparce matrix SM at index [vocabulary_index][document_index]
10. Save these 3 data structures in separate files.

Part 2B (Querying):

```
# decalre names of all files generated in part 2A
vocabulary_index_file_path = ".\\data\\vocabulary.pkl"
document_index_file_path = ".\\data\\file_index.pkl"
inverted_index_file_path = ".\\data\inverted_index.pkl"
```

How is works (algorithm):

1. Read the 3 data structure files listed above and get contents into the program.
2. Convert all text to lower case.
3. Calculate ltc score for the query (consider query as a document).
4. Calculate lnc score for each of the documents (required data is in sparce matrix SM)
5. Compute relevancy score by summing the product of ltc score and lnc score.
6. Arrange by descending order of relevancy score.
7. List top 10 documents.

System evaluation (with common nouns):

Query	Top K Documents		Score	Is the document relevant to the query?
	doc_id	Document Title		
Irish language	5813	C. S. Lewis	0.999949072	Yes. Doc contains 'irish' 12 times and 'language' 4 times.
	5830	County Mayo	0.999501739	Yes. Doc contains 'irish' 27 times and 'language' 3 times.
	6501	Columbanus	0.998533453	Yes. Doc contains 'irish' 7 times and 'language' 2 times.
	6514	County Dublin	0.997112175	Yes. Doc contains 'irish' 5 times and 'language' 2 times.

	6561	Columba	0.993525206	Yes. Doc contains 'irish' 10 times and 'language' 1 time.
	6532	Charlotte Brontë	0.978468009	Yes. Doc contains 'irish' 1 times and 'language' 1 time.
	6643	Croquet	0.978468009	Yes. Doc contains 'irish' 1 times and 'language' 1 time.
	6677	Classical liberalism	0.978468009	Yes. Doc contains 'irish' 1 times and 'language' 1 time.
	6546	Celts	0.964628002	Yes. Doc contains 'irish' 14 times and 'language' 48 times.
	6678	Cat	0.921770635	Yes. Doc contains 'irish' 1 times and 'language' 5 times.

System evaluation (with proper nouns):

Query	Top K Documents		Score	Is the document relevant to the query?
	doc_id	Document Title		
Joy and Lewis	5813	C. S. Lewis	0.725262	Yes. Doc contains Joy 11 times, and 243 times Lewis 176 times
	6652	Cleveland Indians	0.390068	Yes. Doc contains Joy 1 time, and 363 times Lewis 1 time
	6533	Charles Williams (British writer)	0.376577	Partial Yes. Doc contains Joy 0 time, and 51 times, Lewis 8 times
	6612	Cincinnati Bengals	0.313785	Partial Yes. Doc contains Joy 0 time, and 107 times, Lewis 4 times
	6435	Canes Venatici	0.304627	Partial Yes. Doc contains Joy 1 time, and 22 times, Lewis 0 times
	6710	Coyote	0.291599	Partial Yes. Doc contains Joy 0 time, and 295 times, Lewis 5 times
	6548	Claude Monet	0.239645	Partial Yes. Doc contains Joy 1 time, and 158 times, Lewis 0 times
	6611	Chicago Bears	0.233413	Partial Yes. Doc contains Joy 1 time,

				and 176 times, Lewis 0 times
	6503	Concord, New Hampshire	0.21648	Partial Yes. Doc contains Joy 0 time, and 87 times, Lewis 1 times
	6719	Columbia, Missouri	0.190961	Partial Yes. Doc contains Joy 0 time, and 231 times, Lewis 1 times

System evaluation (with rare terms):

Query	Top K Documents		Score	Is the document relevant to the query?
	doc_id	Document Title		
tuomas Cosmos holopainen sojourner lakeview Testament	6824	Carl Sagan	0.883940681	Yes. Doc contains tuomas 1 time, Cosmos 18 times, Holopainen 1 time, Sojourner 1 time, Lakeview 1 time, Testament 1 time.
	5813	C. S. Lewis	0.409035633	Partial Yes. Doc contains tuomas 0 time, Cosmos 1 times, Holopainen 0 time, Sojourner 0 time, Lakeview 0 time, Testament 2 time.
	5820	Confucianism	0.319493175	Partial Yes. Doc contains tuomas 0 time, Cosmos 1 times, Holopainen 0 time, Sojourner 0 time, Lakeview 0 time, Testament 0 time.
	6459	Wu Xing	0.319493175	Partial Yes. Doc contains tuomas 0 time, Cosmos 1 times, Holopainen 0 time, Sojourner 0 time, Lakeview 0 time, Testament 0 time.
	6516	Cosmological argument	0.319493175	Partial Yes. Doc contains tuomas 0 time, Cosmos 1 times, Holopainen 0 time,

				Sojourner 0 time, Lakeview 0 time, Testament 0 time.
	6424	Corona Australis	0.270331299	Partial Yes. Doc contains tuomas 0 time, Cosmos 0 times, Holopainen 0 time, Sojourner 0 time, Lakeview 0 time, Testament 1 time.
	6542	Czesław Miłosz	0.270331299	Partial Yes. Doc contains tuomas 0 time, Cosmos 0 times, Holopainen 0 time, Sojourner 0 time, Lakeview 0 time, Testament 1 time.
	6599	Chaldea	0.270331299	Partial Yes. Doc contains tuomas 0 time, Cosmos 0 times, Holopainen 0 time, Sojourner 0 time, Lakeview 0 time, Testament 1 time.
	6704	Christendom	0.270331299	Partial Yes. Doc contains tuomas 0 time, Cosmos 0 times, Holopainen 0 time, Sojourner 0 time, Lakeview 0 time, Testament 3 times.
	6728	Christianity and antisemitism	0.270331299	Partial Yes. Doc contains tuomas 0 time, Cosmos 0 times, Holopainen 0 time, Sojourner 0 time, Lakeview 0 time, Testament 9 times.

System evaluation (with ambiguous terms):

Query	Top K Documents		Score	Is the document relevant to the query?
	doc_id	Document Title		
Chinese England Testament	6728	Christianity and antisemitism	0.89173792	Partial Yes. Doc contains Chinese 0 times, England 1 time, Testament 9 times

	6424	Corona Australis	0.848071944	Partial Yes. Doc contains Chinese 2 times, England 0 time, Testament 1 times
	6599	Chaldea	0.842055224	Partial Yes. Doc contains Chinese 0 times, England 1 time, Testament 1 times
	6704	Christendom	0.829433068	Partial Yes. Doc contains Chinese 0 times, England 4 time, Testament 3 times
	6469	Canon law	0.821754715	Partial Yes. Doc contains Chinese 0 times, England 3 time, Testament 2 times
	6542	Czesław Miłosz	0.803495762	Partial Yes. Doc contains Chinese 0 times, England 0 time, Testament 1 times
	6824	Carl Sagan	0.803495762	Partial Yes. Doc contains Chinese 0 times, England 0 time, Testament 1 times
	6586	Claus Sluter	0.803495762	Partial Yes. Doc contains Chinese 0 times, England 0 time, Testament 2 times
	5813	C. S. Lewis	0.76283448	Partial Yes. Doc contains Chinese 0 times, England 10 time, Testament 2 times
	6449	Clock	0.595255152	Partial Yes. Doc contains Chinese 3 times, England 2 time, Testament 0 times

System evaluation (with complex query terms):

Query	**Top K Documents**		**Score**	**Is the document relevant to the query?**
	doc_id	**Document Title**		
Chinese writer sitting at Church of England	6675	Conservatism	0.435388424	Partial Yes. Doc contains Chinese 1 time, Writer 5 times, Sitting 0 time, At 688 times, Church 21 times,

				Of 323 times, England 3 times
	5820	Confucianism	0.420418014	Partial Yes. Doc contains Chinese 34 time, Writer 1 times, Sitting 0 time, At 327 times, Church 17 times, Of 331 times, England 0 times
	5813	C. S. Lewis	0.409459911	Partial Yes. Doc contains Chinese 0 time, Writer 7 times, Sitting 0 time, At 343 times, Church 14 times, Of 358 times, England 10 times
	6751	Cottingley Fairies	0.385829752	Partial Yes. Doc contains Chinese 0 time, Writer 1 times, Sitting 1 time, At 142 times, Church 0 times, Of 99 times, England 3 times
	6466	Connecticut	0.349940175	Partial Yes. Doc contains Chinese 0 time, Writer 1 times, Sitting 0 time, At 657 times, Church 6 times, Of 400 times, England 19 times
	6629	Candide	0.331681569	Partial Yes. Doc contains Chinese 0 time, Writer 4 times, Sitting 0 time, At 325 times, Church 3 times, Of 309 times, England 2 times
	6533	Charles Williams (British writer)	0.32884022	Partial Yes. Doc contains Chinese 0 time, Writer 4 times, Sitting 0 time, At 49 times,

				Church 2 times, Of 49 times, England 1 times
	6469	Canon law	0.326129385	Partial Yes. Doc contains Chinese 0 time, Writer 0 times, Sitting 0 time, At 114 times, Church 56 times, Of 96 times, England 3 times
	6814	Congregationalist polity	0.320756808	Partial Yes. Doc contains Chinese 0 time, Writer 0 times, Sitting 0 time, At 163 times, Church 46 times, Of 98 times, England 2 times
	6449	Clock	0.319514174	Partial Yes. Doc contains Chinese 3 time, Writer 0 times, Sitting 0 time, At 390 times, Church 3 times, Of 222 times, England 2 times