EEE 486/586 Statistical Foundations of Natural Language Processing
Midterm Project Report
Enes Koşar – 21901224

ABSTRACT

In this report, I will investigate some basic laws and methods used in Natural Language Processing (NLP). I will start choosing some corpora. Then I will apply necessary preprocessing steps. So that, the corpora will be ready to extract meaningful statistics and information. I will explore frequencies of word types, type frequency vs rank plots and type size vs token size plots for different variations of my corpora. I will investigate if the results are consistent with the Zipf's Law and Heaps' Law. I will explore the effect of removing stop words from the corpora. Then, I will try to find a simple clustering method to separate one particular corpus from another. At the end I will explore if a randomly generated text obeys Zipf's Law and Heaps' Law.

INTRODUCTION

NLP is a subfield of computer science and linguistics. It's concern is about how to program computers to process and analyze natural language data. The challenges in NLP are mainly speech recognition, natural language understanding and natural language generation. For this purposes, scientist explored some statistical laws. These laws are not as strong as physics laws. But still, they provide a good understanding about the text or the language for us. Two important laws are Zipf's Law and Heaps' Law.

Zipf's Law explains the relation between rank and frequency of a type. It states that there is a constant K such that:

K = rank * frequency

So, Zipf's Law states that frequency of a type is inversely proportional to rank of that type. When we take logarithm of both sides:

Log(K) = Log(rank) + Log(frequency)

A result of Zipf's Law is that, inverse proportionality is in linear format when we evaluate frequency-rank relation in Log-Log format.

For a given corpus, type-token relation is explained via **Heaps' Law.** (1) It is an empirical law which describes the number of distinct words(types) in a document as a function of the document length (total number of token). It is formulated as:

$$V_R(n) = kn^{\beta}$$

In this formula, $V_R(n)$ is number of types (vocabulary size), n is number of tokens, k and β are free parameters determined empirically. In an English text corpora, k is typically between 10-100 and β is between 0.4 and 0.6.

I will investigate if various corpora are consistent with Zipf's Law and Heaps' Law. I will also explore the effect of removing stop words to these laws. I will develop a simple clustering method based on these laws. Then I will check if a randomly generated corpus obeys Zipf's Law and Heaps' Law.

CORPUS CONSTRUCTION AND IMPLEMENTATION

I will investigate some fundamental laws of statistical natural language processing. I will explore these laws by using 18 different books. Each book is approximately 1 MB. There will be two categories: Author based grouped books and genre based grouped books. In each category, I will choose 3 different author/genre. For each author/genre, I will pick 3 different books. My books are as follows:

(I) Author based books:
- Charles Dickens
  - Bleak House
  - Dombey and Son
  - Our Mutual Friend
- Dostoyevsky
  - Idiot
  - Karamazov Brothers
  - Possessed
- Tolstoy
  - Anna Karenina
  - Resurrection
  - War and Peace

(II) Genre based books:
- Biography
  - Life and Letters of Robert Browning
  - Life of Froude
  - Life of William Ewart
- Detective
  - Moonstone
  - Sherlock
  - Twenty Years Detective
- Myths
  - Assyria
  - China
  - Rome

After choosing my books, I followed these pre-processing steps:

- Tokenization
- Getting rid of punctuation marks
- Casting every word to lowercase

I used following basic Python functions for this purpose: split(), str.maketrans(), translate(), lower(). So that, I obtained a corpus for each book. Then, I determined a common English stop word list. Then I removed those words from my corpus. Stop word list(2) is attached in appendix. So that, I obtained my corpora which are ready to apply statistical methods.

RESULTS

(e) Type frequencies are calculated by using Counter() function from the Counter library. This is done for all the books before and after stop word removal. Results are written in text files. Two examples can be seen partly in the figures below:

1	the 10432
2	and 7030
3	to 6669
4	of 5549
5	a 4845
6	i 4627
7	you 4257
8	he 4233
9	that 3627
10	in 3510
11	was 3093
12	it 2774
13	his 2482
14	at 2164
15	not 2157
16	had 2086
17	with 1950
18	for 1924
19	is 1900
20	she 1793
21	as 1787
22	but 1783
23	her 1760
24	all 1647
25	prince 1643
26	have 1560

Fig1: Word type vs Frequencies of Idiot

(Before stop word removal)

1	prince 1643
2	“i 596
3	don’t 454
4	time 408
5	aglaya 407
6	nastasia 365
7	little 356
8	cried 334
9	rogojin 323
10	gania 311
11	lebedeff 305
12	moment 299
13	course 283
14	house 273
15	eyes 256
16	tell 254
17	suddenly 251
18	“oh 238
19	look 229
20	“you 227
21	looked 214
22	evgenie 213
23	told 205
24	colia 204
25	word 199
26	people 190

Fig2: Word type vs Frequencies of Idiot

(After stop word removal)

(f) I composed larger author corpora by combining books of the same authors. Frequency/Rank plots are as follows:

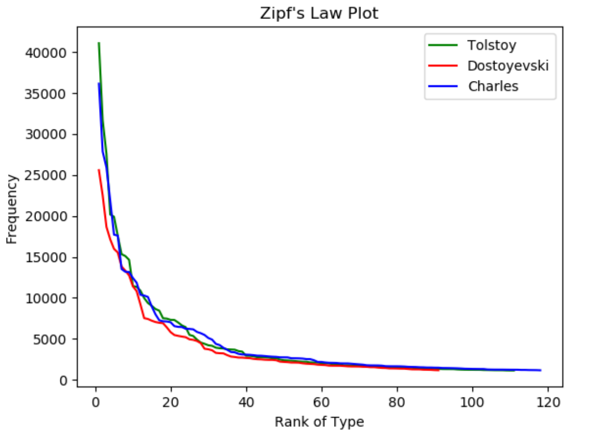

Fig3: Normal Frequency/Type

Plot of Author Corpora

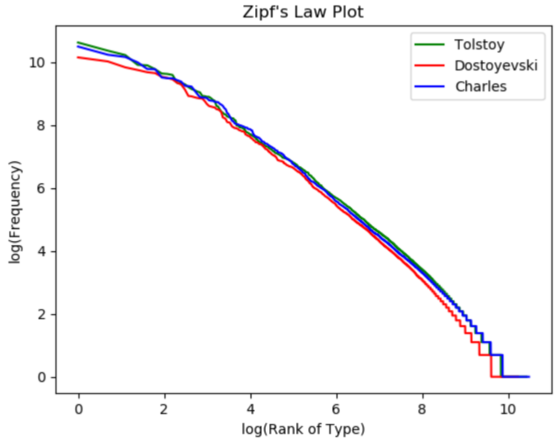

Fig4: Log-Log Frequency/Type

Plot of Author Corpora

- Figure 3 shows that rank and frequency of a type are inversely proportional.

- Figure 4 shows that when we plot frequency/rank plot in log-log form, it becomes approximately a linear line.

Zipf's Law says that there is a constant k such that k = rank*frequency. Figure 3 is obviously consistent with this formula.

When we take logarithm of both sides, we see that log(k) = log(rank)+log(frequency). This formula shows that frequency/rank relationship in log-log form is inversely linear. So, figure 4 is also consistent with Zipf's Law.

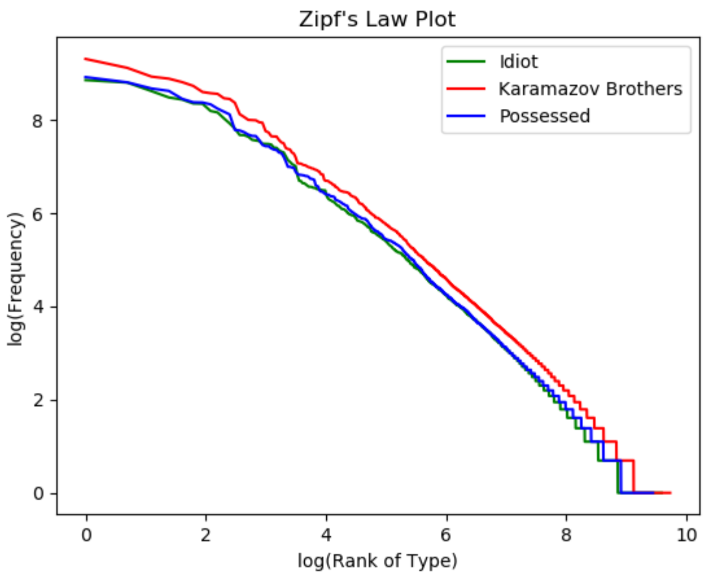

Fig5: Books of Dostoyevsky

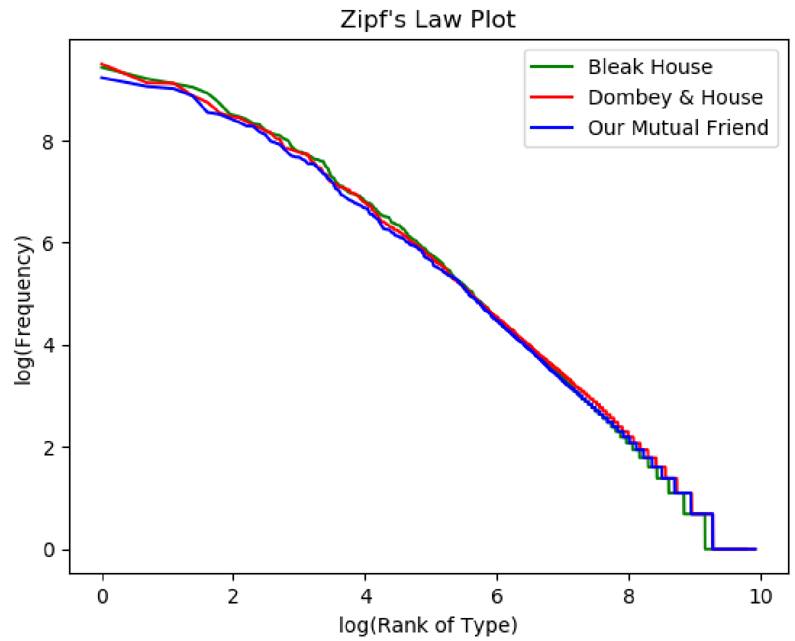

Fig6: Books of Charles Dickens

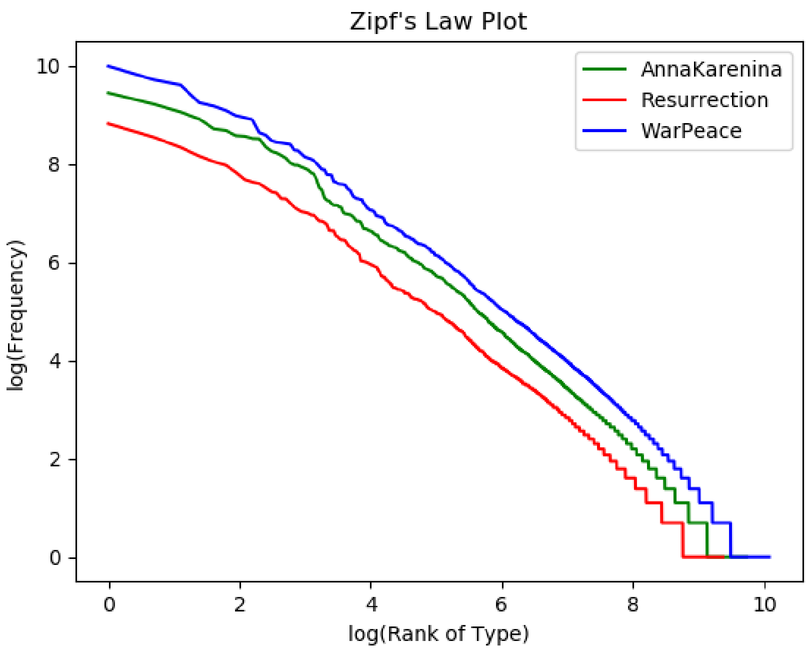

Fig7: Books of Tolstoy

In the figures 5,6 and 7 we see the frequency/rank plot of different books of different authors. These plots are in log-log form and they are linear and inversely proportional as in the larger author corpora case. Since Zipf's Law says that log(k) = log(rank)+log(frequency), these three plots are consistent with Zipf's Law.

(g) Token is number of words in a corpus. Type is number of unique words. For the larger author corpora, I plotted type/token graph below in normal and Log-Log formats. In normal format, as we traverse the corpus, number of types decreasingly increases. This makes sense intuitively. Because as we traverse the corpus, some types will be already seen. So, they won't be added to total number of types.

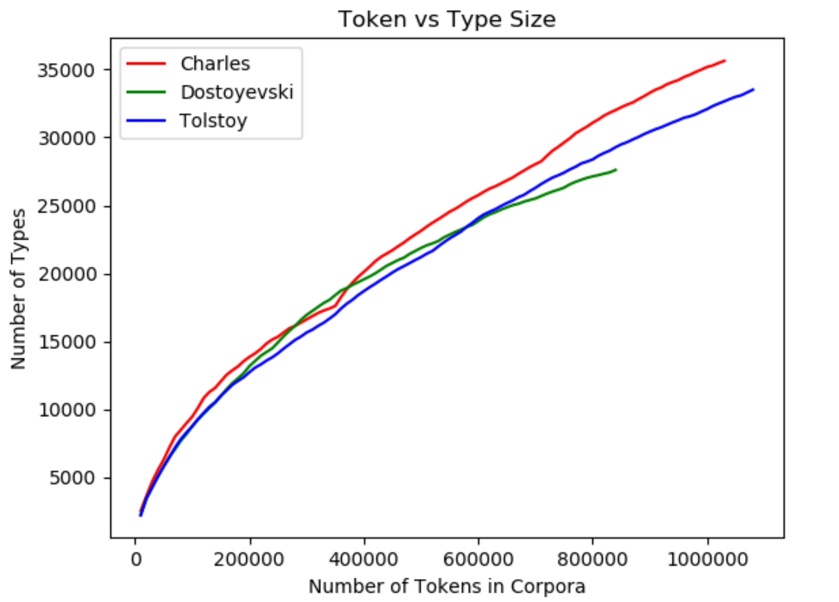

Fig8: Type Token Relation for Larger Author Corpora

When we plot this graph in Log-Log format we see an almost linear relation.

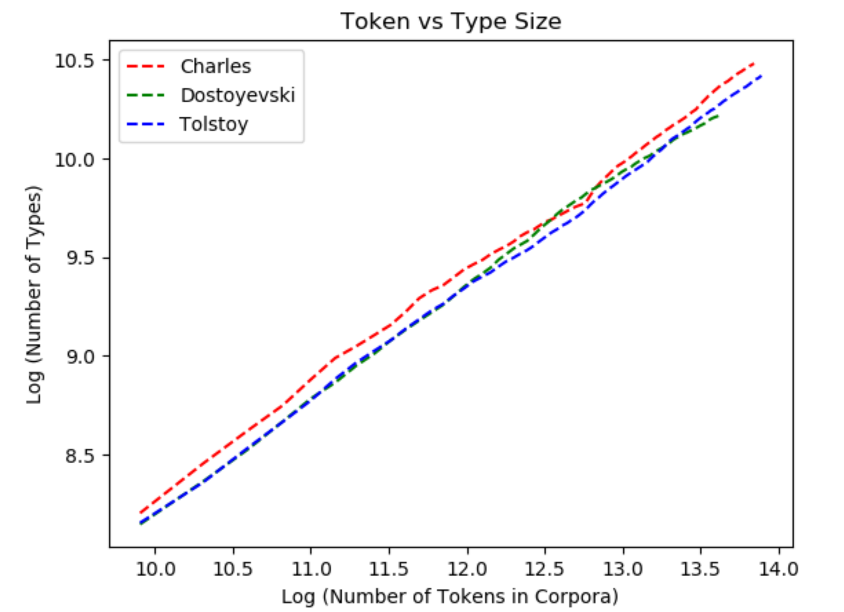

Fig9: Type Token Relation for Larger Author Corpora in Log-Log Format

For a given corpus, type-token relation is explained via **<u>Heaps' Law.</u>** (1) It is an empirical law which describes the number of distinct words(types) in a document as a function of the document length (total number of token). It is formulated as:

$$V_R(n) = kn^{\beta}$$

In this formula, $V_R(n)$ is number of types (vocabulary size), n is number of tokens, k and β are free parameters determined empirically. In an English text corpora, k is typically between 10-100 and β is between 0.4 and 0.6.

The type-token plots we obtained are obviously consistent with Heaps' Law.

(h) When I explore the relation between number of type and token for each book separately, I obtained the following figures.

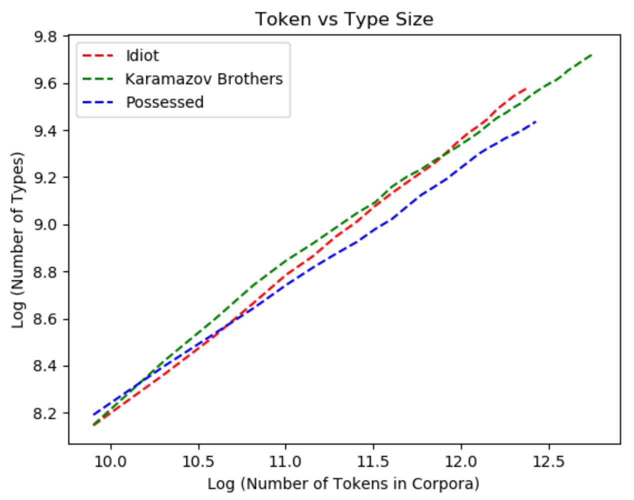

Fig10: Type Token Relation for Books of Dostoyevsky

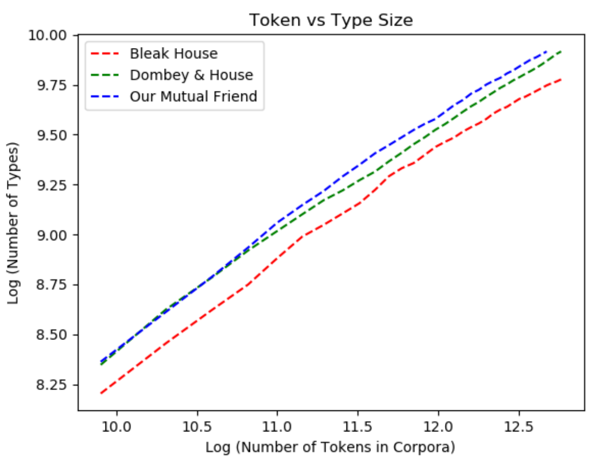

Fig11: Type Token Relation for Books of Charles Dickens

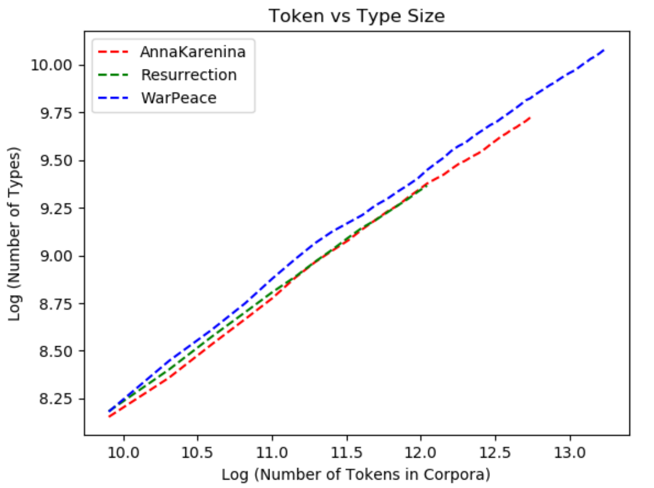

Fig12: Type Token Relation for Books of Tolstoy

There are small differences between slopes of the lines. Larger slope means that the author uses new types in the new pages of the book. Small slope means that the author doesn't use so many new words in upcoming pages. So, type-token ratio gives us idea about richness of vocabulary usage.

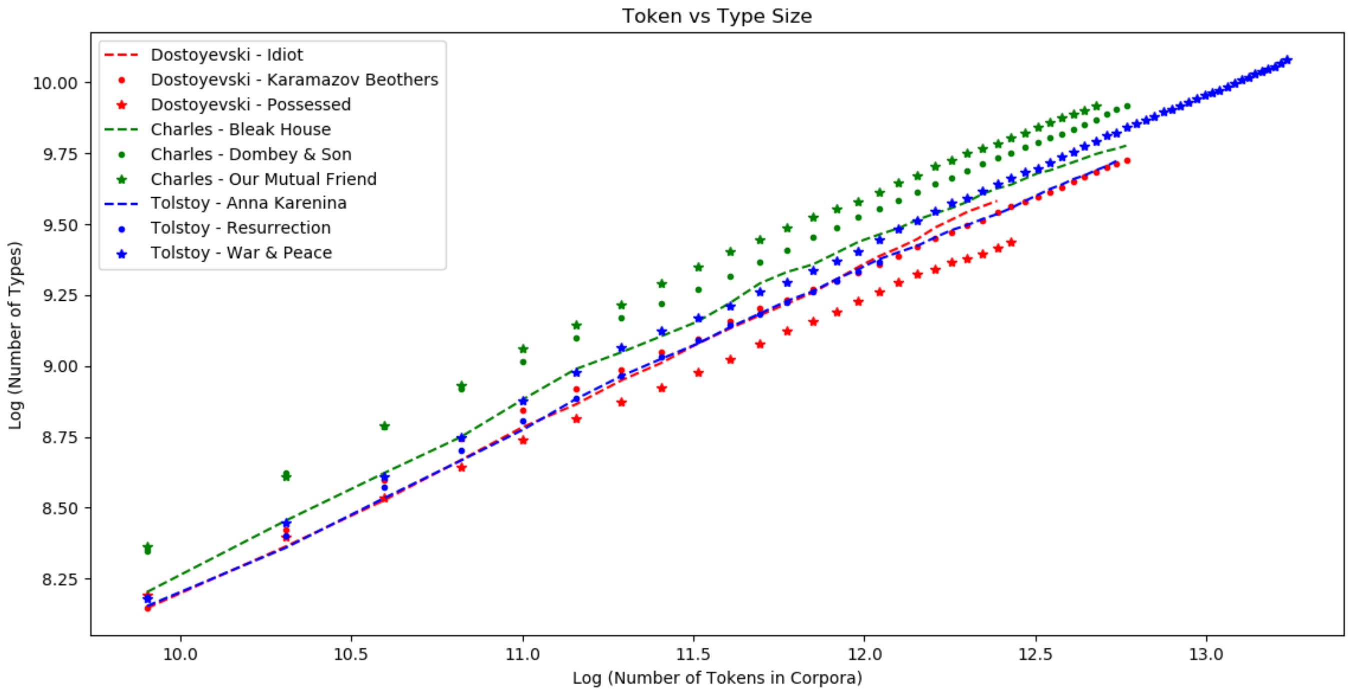

Fig13: Type Token Relation for All Books

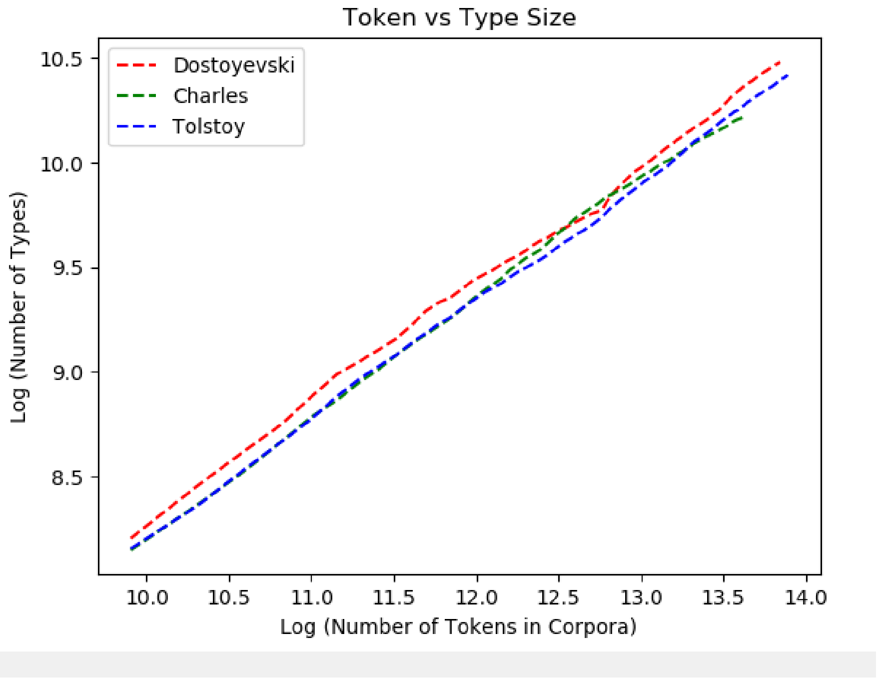

Fig14: Type Token Relation for Larger Author Corpora

By looking at figure 14, one can say that Dostoyevsky's usage of vocabulary is his books are richer than Tolstoy and Charles. However, the corpora include stop words in this plot. So, it may give wrong idea about the real content of the books.

(i) I find the best fitting lines in terms of mean square error (MSE).

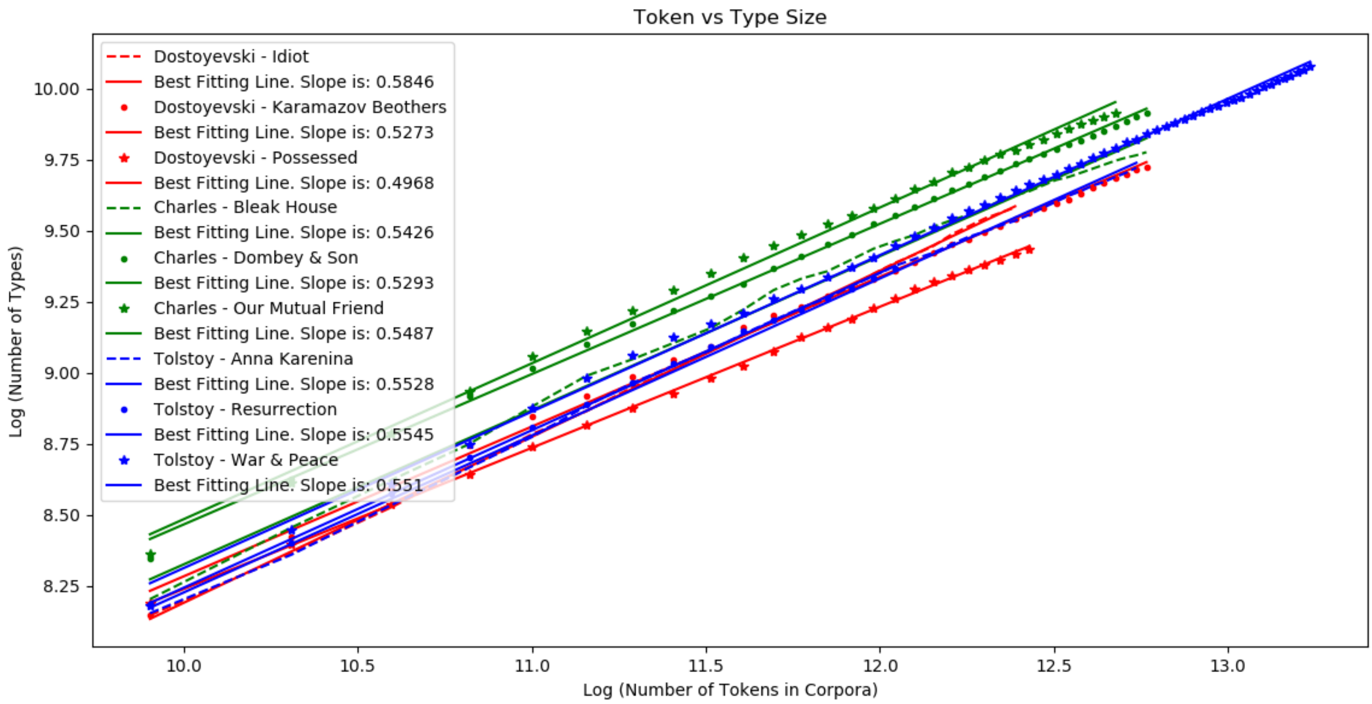

Fig15: Type Token Relation and best Fitting Lines for Each Book

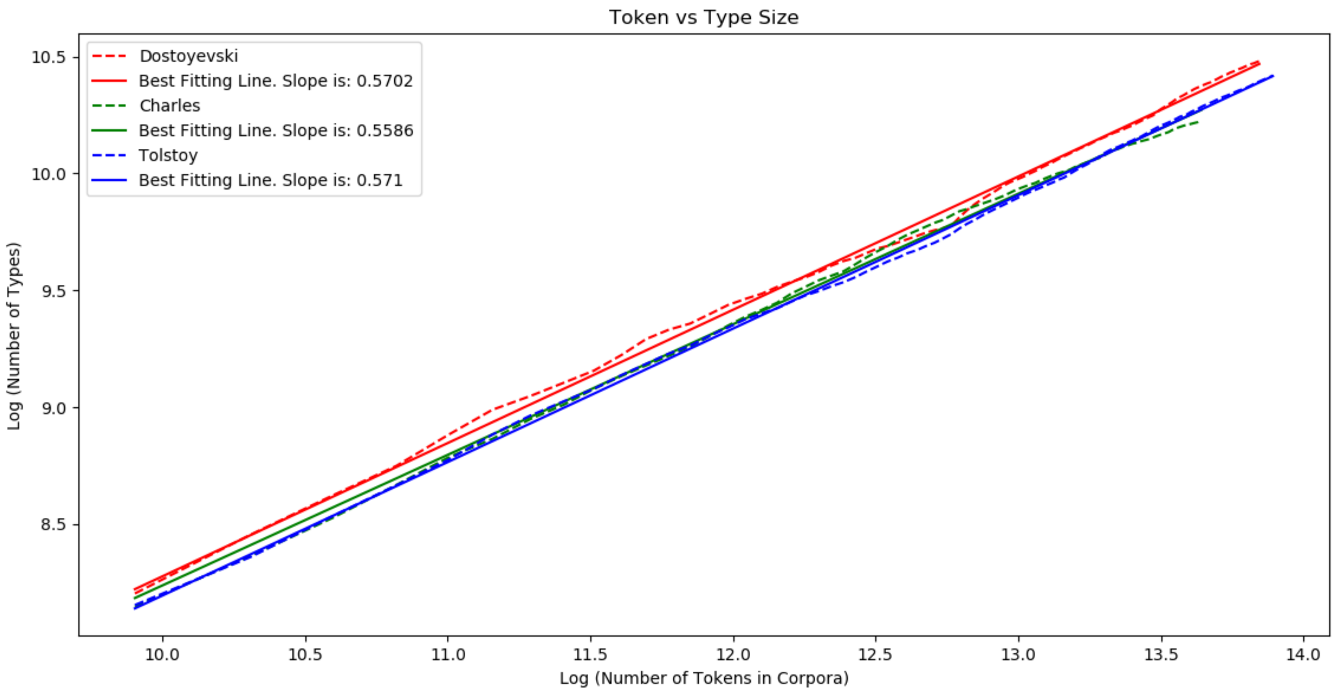

Fig16: Type Token Relation and best Fitting Lines for Larger Author Corpora

Book Name	Best Fitting Line Slope
Dostoyevsky - Idiot	0.5846
Dostoyevsky – Karamazov Brothers	0.5273
Dostoyevsky - Possessed	0.4968
Charles Dickens – Bleak House	0.5426
Charles Dickens – Dombey & Son	0.5293
Charles Dickens – Our Mutual Friend	0.5487
Tolstoy- Anne Karenina	0.5528
Tolstoy- Resurrection	0.5545
Tolstoy- War & Peace	0.5510

Table1: Best Fitting Line Slopes of Each Book

Larger Author Corpora	Best Fitting Line Slope
Dostoyevsky	0.5702
Charles Dickens	0.5586
Tolstoy	0.5710

Table2: Best Fitting Line Slopes of Larger Author Corpora

Slopes are pretty close to each other. The reason is probably stop words. Because they are major part of the corpora and affect the slopes a lot. So, dominance of stop words don't allow us to get an idea about the content of the books. Another observation is that: Books of Tolstoy have a richer vocabulary than others. Because slopes of the books of Tolstoy are higher. This is a sign of rich vocabulary usage of the author. On the other hand, Merged corpora slope of Dostoyevsky is very high compared to single slope of his books. This is a sign that, Dostoyevsky uses different vocabulary in his different books. Also, his vocabulary richness change from book to book. However, Tolstoy uses more standard vocabulary richness in his books. Because all his books have similar slopes.

(j) I repeat parts (h) and (i) for literary types corpora.

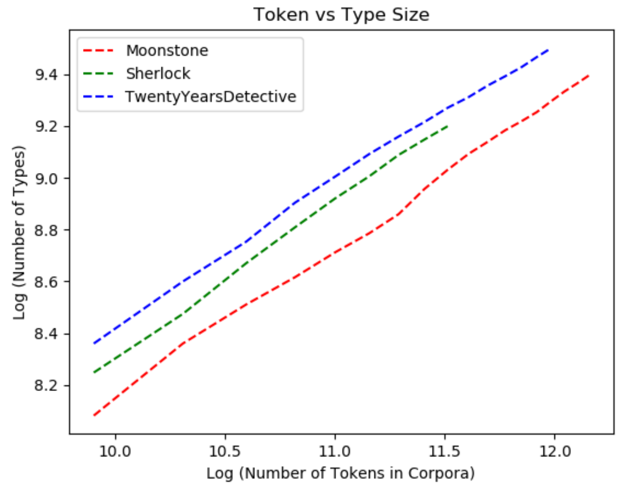

Fig17: Type Token Relation for the Books in Detective Genre

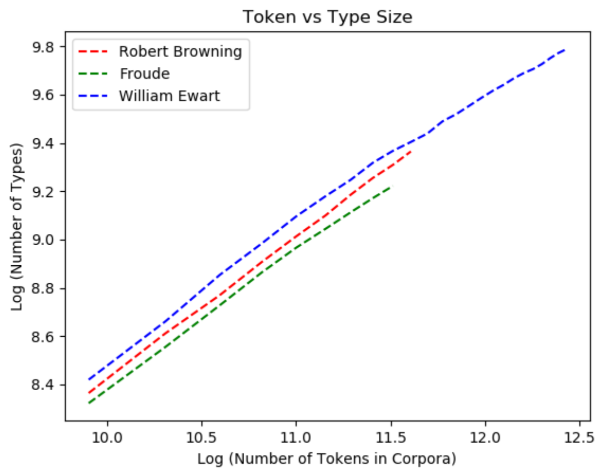

Fig18: Type Token Relation for the Books in Biography Genre

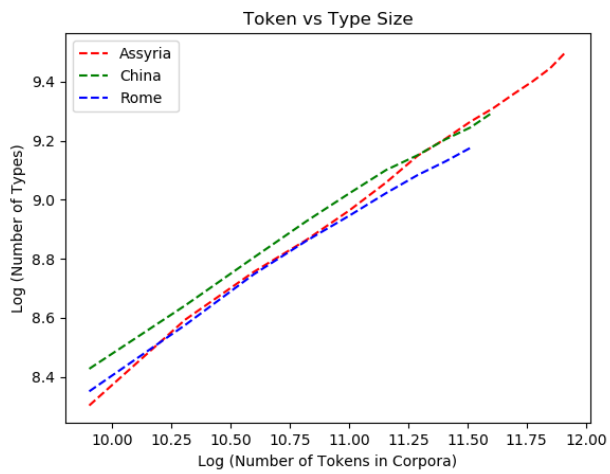

Fig19: Type Token Relation for the Books in Myth Genre

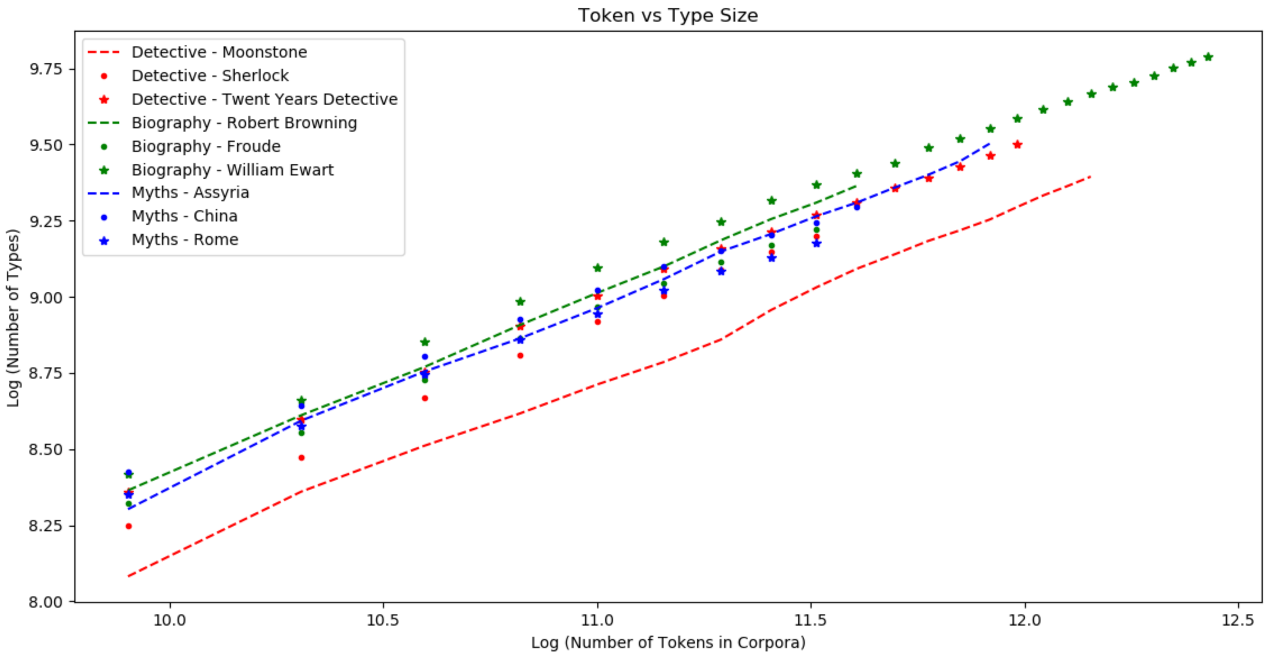

Fig20: Type Token Relation for Each Book in Genre Group

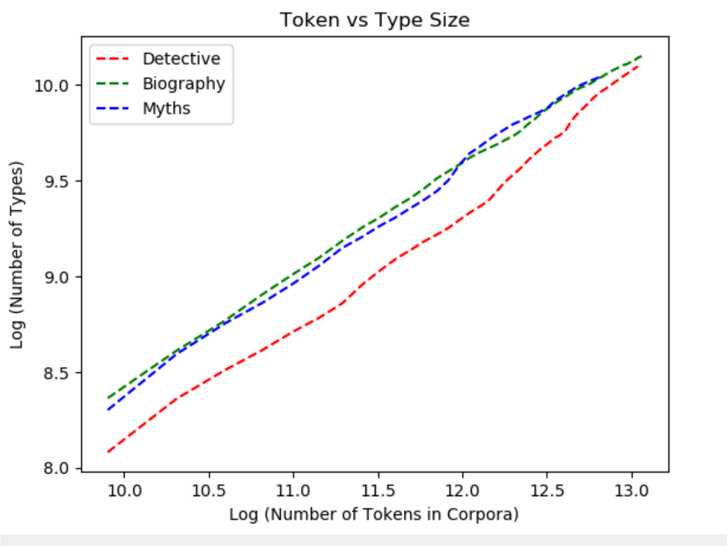

Fig21: Type Token Relation for Larger Genre Corpora

When we look at the best fitting lines, we see that they are more separable than author based books. This is probably because of that author based books are all literary books and their contents are more similar to each other. However, for different genres richness of vocabulary varies more. For example vocabulary richness of biography type of books are smaller than myths and detectives. This is as expected. Because describing detective stories or myths requires richer vocabulary than describing someone’s life.

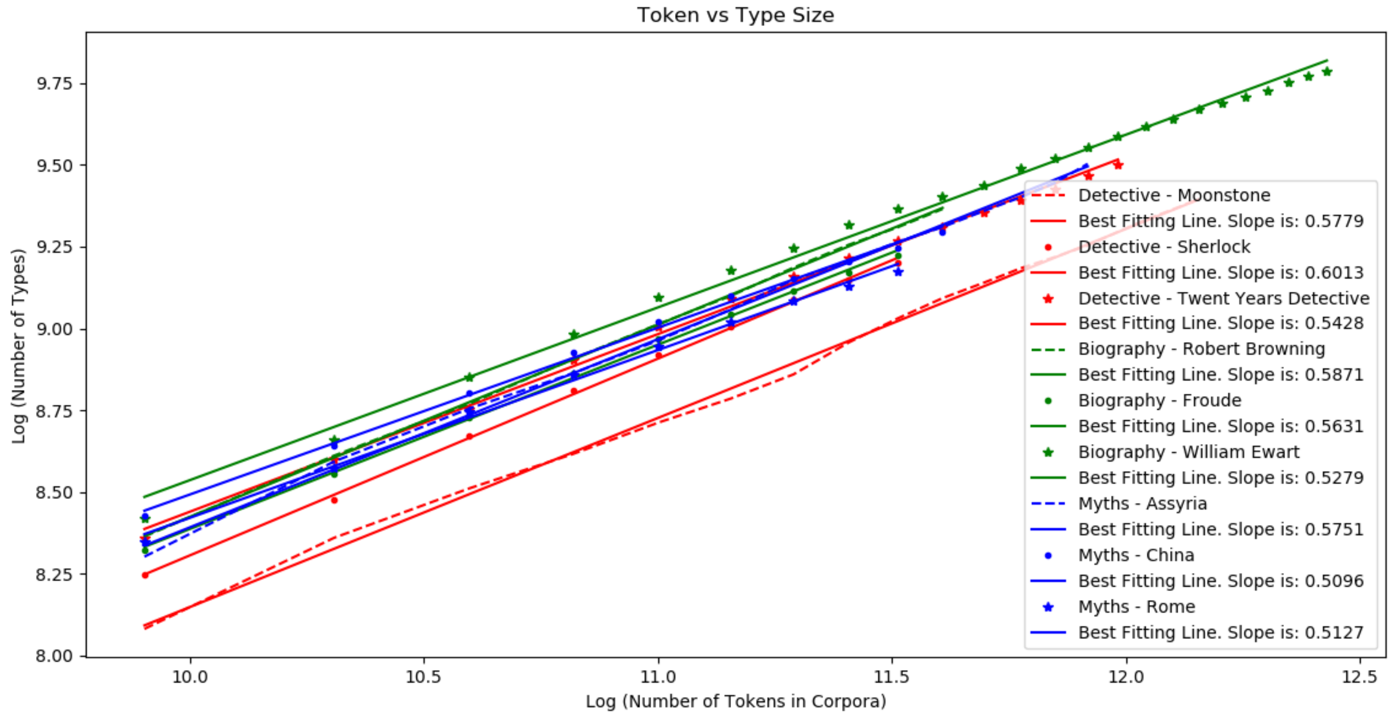

Fig22: Best Fitting Lines for Each Book in Genre Class

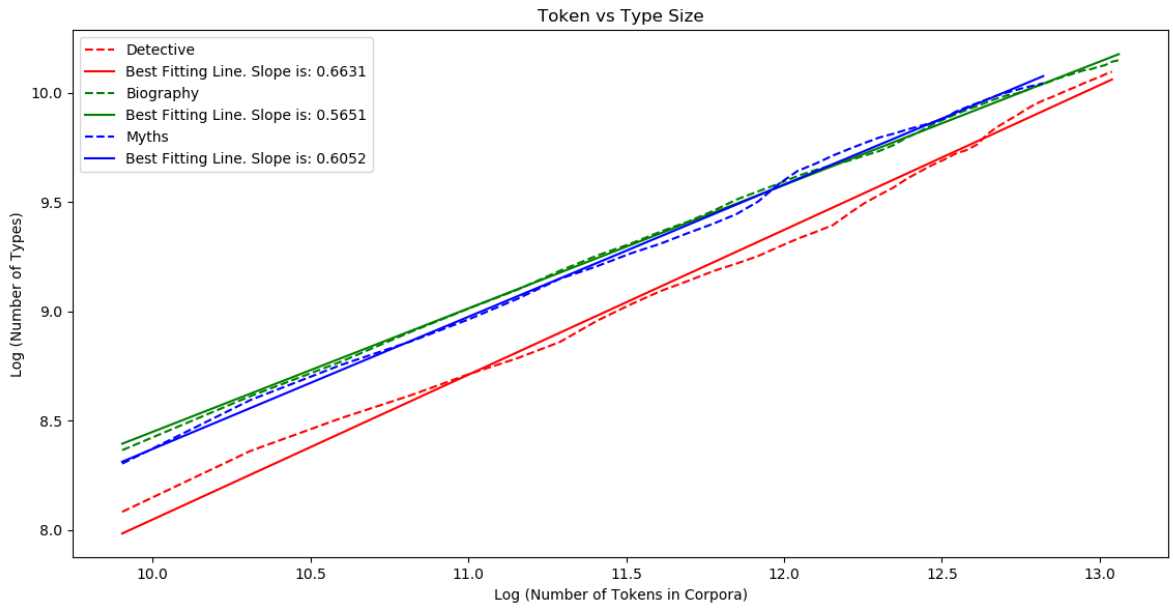

Fig23: Best Fitting Lines for Larger Genre Corpora

Another observation is that: Biography books individually have higher slopes than Myths books. However merged corpus of biography books have smaller slope than merged corpus of myths books. This means that different biography books have similar vocabularies. So, when we merge the books, total slope isn't so high. On the other hand, different myths books have different vocabularies. Hence, slope of merged books of myths have higher slope than biography books.

Book Name	Best Fitting Line Slope
Detective - Moonstone	0.5779
Detective - Sherlock	0.6013
Detective - Twenty Years Detective	0.5428
Biography – Robert Browning	0.5871
Biography – Froude	0.5631
Biography – Ewart	0.5279
Myths - Assyria	0.5751
Myths - China	0.5096
Myths - Rome	0.5127

Table3: Best Fitting Line Slopes of Each Book in Genre Class

Larger Author Corpora	Best Fitting Line Slope
Detective	0.6631
Biography	0.5651
Myths	0.6052

Table4: Best Fitting Line Slopes of Larger Genre Corpora

(k) When we look at the table1 we see that books of the same author have very similar slopes. This intuitively makes sense. Because type token slope represents the vocabulary of the author. So, it is expected that different books of the same author have similar vocabularies and similar slope values. Hence for Charles Dickens and Tolstoy we can use this as a clue and determine a clustering method based on average slopes of the books:

- We can classify the books which has slope values 0.55 with error margin 0.005 as Tolstoy.

- We can classify the books which has slope values 0.535 with error margin 0.015 as Charles Dickens.

- However, we can’t classify books of Dostoyevsky. Because Dostoyevsky has a very large variance of type/token slopes for the given three books. This is a possible case, because the author may prefer different vocabularies while writing different books. Types of these books may be different. So, this may change the language.

(I) When I remove stop words, I some observed small changes.

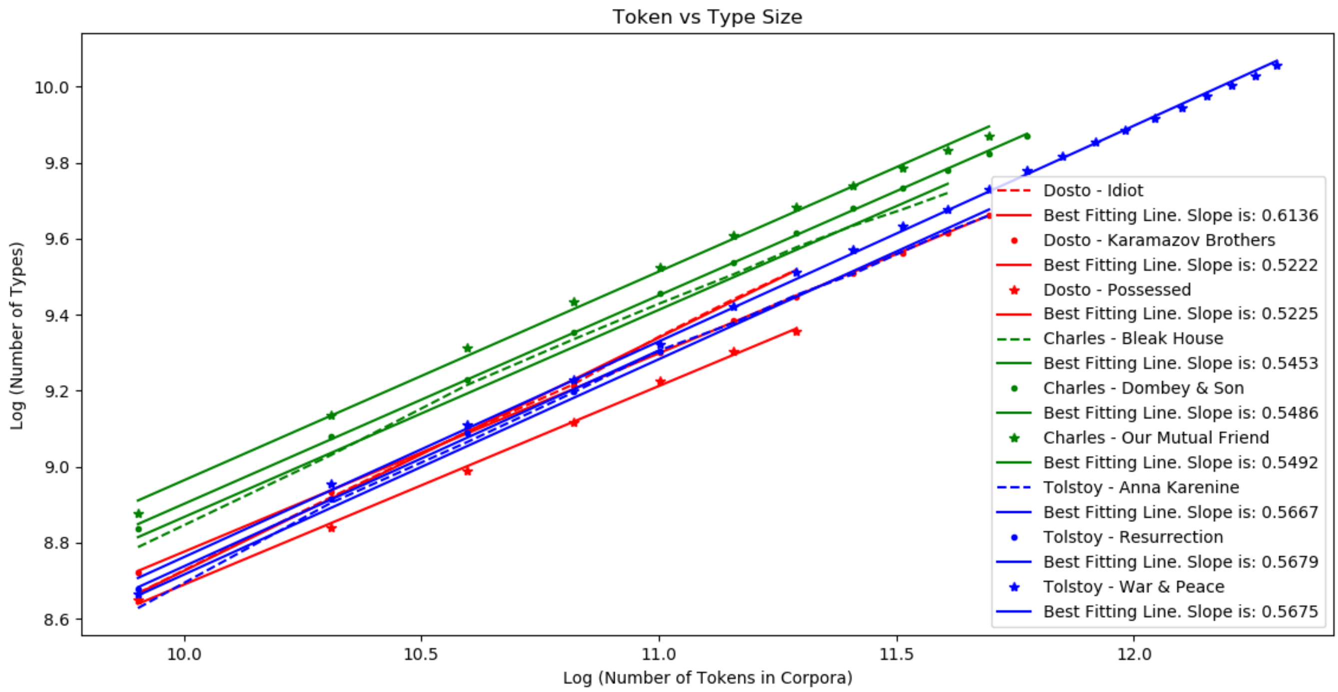

Fig24: Best Fitting Lines for each Author Based Book After Removal of Stop Words

My first observation is that: Variance of slopes for the books of Dostoyevsky decreased. Before removing stop words, slope of each book was very different from each other. Now, slope of two books are almost the same.

My second observation is that: Variance of slopes of merged books increased. This is as intiutivily expected. Because common part of the vocabulary is removed from the corpora. So, differences will be more. Other than that, individual slopes didn't change much as we can see in the figures and tables.

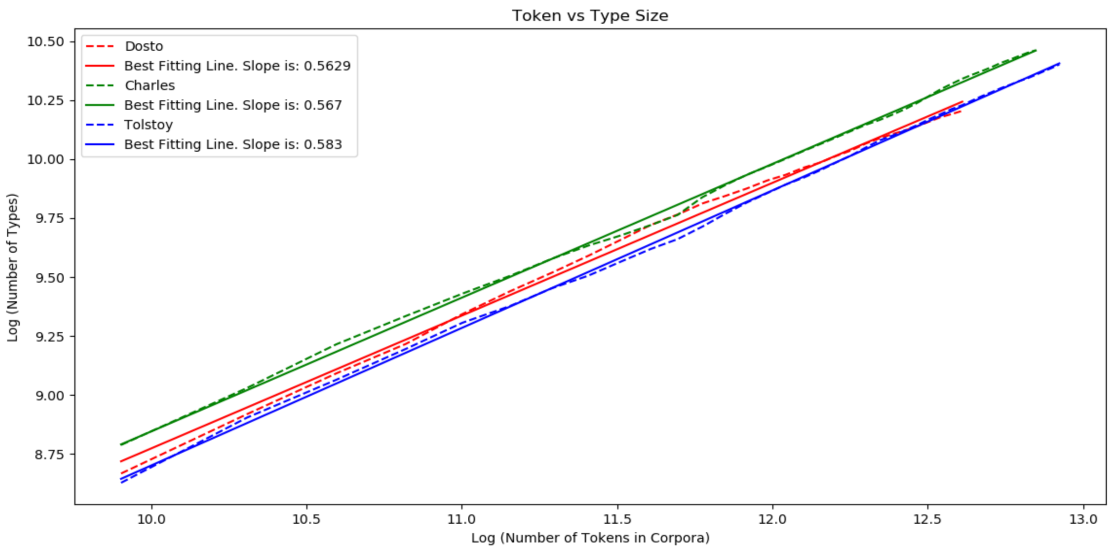

Fig25: Best Fitting Lines for Larger Author Corpora After Stop Word Removal

Book Name	Best Fitting Line Slope
Dostoyevsky - Idiot	0.6136
Dostoyevsky – Karamazov Brothers	0.5222
Dostoyevsky - Possessed	0.5225
Charles Dickens – Bleak House	0.5453
Charles Dickens – Dombey & Son	0.5486
Charles Dickens – Our Mutual Friend	0.5492
Tolstoy- Anne Karenina	0.5667
Tolstoy- Resurrection	0.5679
Tolstoy- War & Peace	0.5675

Table5: Best Fitting Line Slopes of Each Book Without Stop Words

Larger Author Corpora	Best Fitting Line Slope
Dostoyevsky	0.5629
Charles Dickens	0.5670
Tolstoy	0.5830

Table6: Best Fitting Line Slopes of Larger Author Corpora Without Stop Words

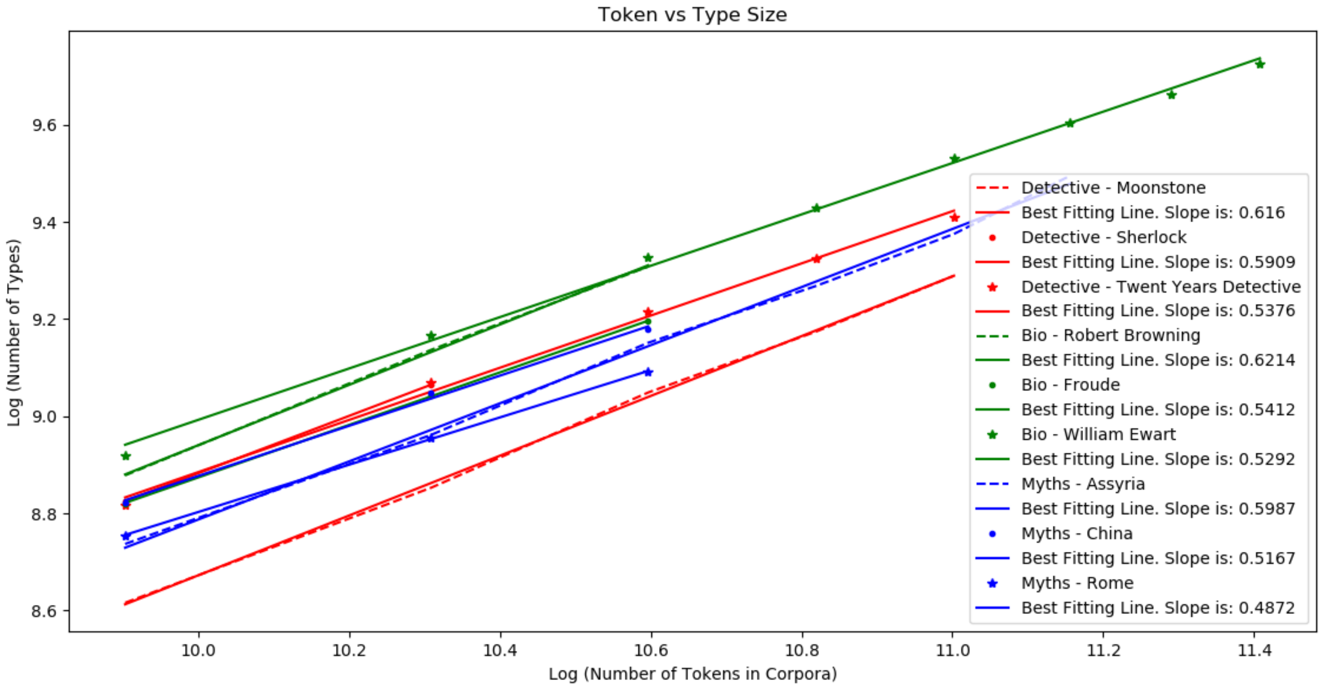

Fig26: Best Fitting Lines for Each Book in Genre Corpora After Stop Word Removal

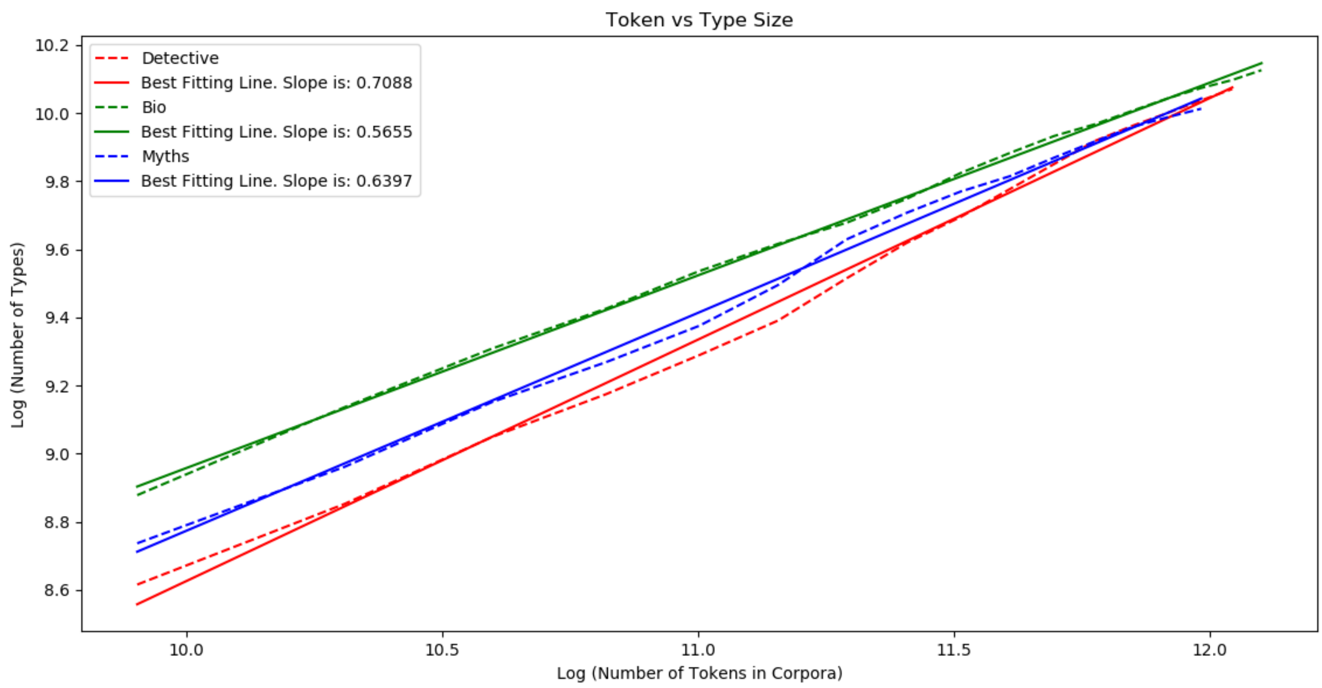

Fig27: Best Fitting Lines for Larger Genre Corpora After Stop Word Removal

Book Name	Best Fitting Line Slope
Detective - Moonstone	0.6160
Detective - Sherlock	0.5909
Detective - Twenty Years Detective	0.5376
Biography – Robert Browning	0.6214
Biography – Froude	0.5412
Biography – Ewart	0.5292
Myths - Assyria	0.5987
Myths - China	0.5167
Myths - Rome	0.4872

Table7: Best Fitting Line Slopes of Each Book in Genre Class Without Stop Words

Larger Author Corpora	Best Fitting Line Slope
Detective	0.7088
Biography	0.5655
Myths	0.6397

Table8: Best Fitting Line Slopes of Larger Genre Corpora Without Stop Words

(m) I created a random corpus which has 1.500.000 words. Length of a word is between 1 and 4. Possible letters are English alphabet letters and numbers. For each word, firstly I choose a random number between 1 and 4 to determine word length. Then I generate a random string with the randomly generated length. All random generations have uniform pdf's. For this randomly generated corpus, I obtained following results. These results show that, this corpus obeys Zipf's Law and Heaps' Law.

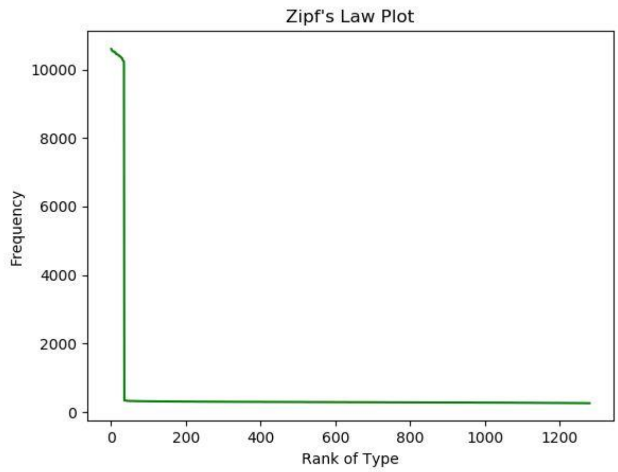

Fig28: Frequency rank relation of Randomly Generated Corpus

Fig29: Frequency rank relation of Randomly Generated Corpus in Log-Log Format

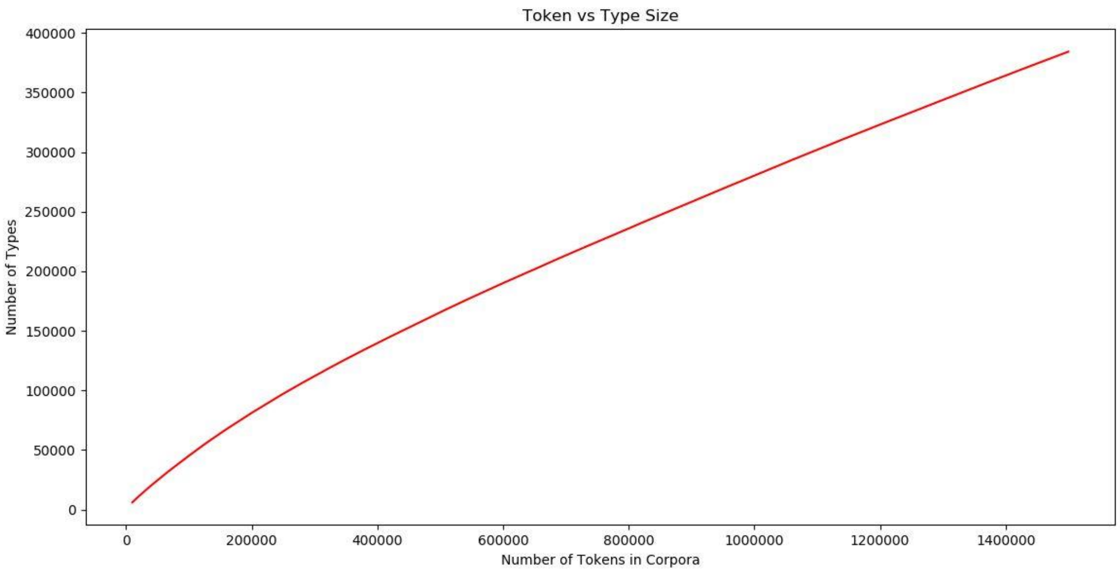

Fig30: Type Token Relation of Randomly Generated Corpus

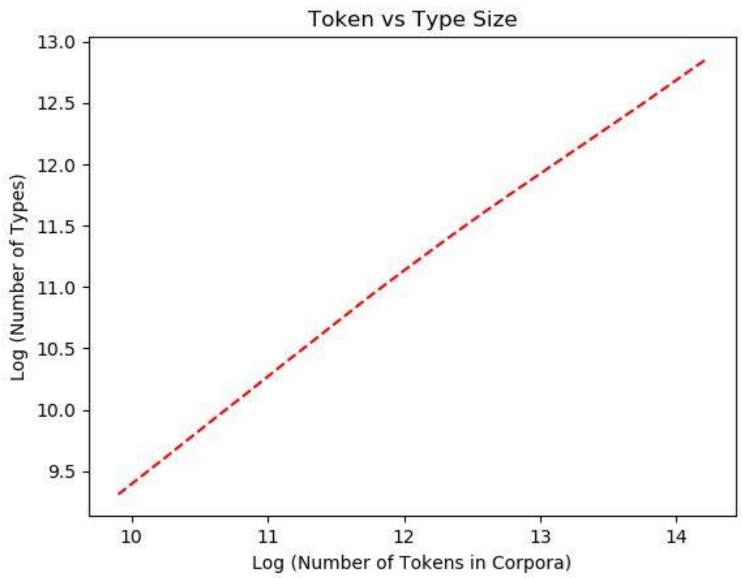

Fig31: Type Token Relation of Randomly Generated Corpus in Log-Log Format

DISCUSSION AND CONCLUSIONS

In this study used 18 different books. By preprocessing them, I created 18 basic corpora. By merging them with groups of 3 books, I created 6 more corpora. Also, I used stop word removed version of these corpora.

I investigated frequency rank relation by using these corpora. I have seen that there is an inverse proportional relation between frequency and rank of types in normal format. When I plot this in Log-Log format, I have seen that the relation becomes linear. This case was correct for any single book, their merged versions and even for randomly generated book. The law explains this relation is called Zipf's Law.

I also investigated number of type-token relation. I have seen that: number of type is a function of number of tokens. This is an increasing function. However, amount of increase decreases at every step. When I investigate this relation in Log-Log format, I have seen a linear relation. This observations were the same for all versions of the corpora. The law explaining this relation is called Heaps' Law. It is formulated as:

$$V_R(n) = kn^{\beta}$$

In this formula, $V_R(n)$ is number of types (vocabulary size), n is number of tokens, k and β are free parameters determined empirically. In an English text corpora, k is typically between 10-100 and β is between 0.4 and 0.6.

Also I investigated that type-token relation gives us idea about the vocabulary of the corpora. For example, detective books tend to have higher type-token ratio than biography books. This means that, vocabulary of detective books are richer than biography books. This information also gives us some hints about the authors of the books. When I checked the type-token ratio of the books of the authors, I have seen that books of the same author have very close type-token ratio slopes. This is because, this ratio slope represents the vocabulary of the authors. So, books of the same author tend to have similar type-token slopes. But there may be exception according to type of the book and target reader of the author. Two books of the same author written in different genres may have different type-token slopes.

At the end, I checked effect of removing stop words. It increased especially the variance of the slopes of merged books. Because stop words are common among different books. So, removing them makes vocabulary of books more different.

I have also investigated that even a randomly generated corpus obeys the Zipf's and Heaps' Laws.

REFERENCES

[1] Wikizeroo.org. 2020. *Heaps' Law - Wikipedia*. [online] Available at: <https://www.wikizeroo.org/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvSGVhcHMnX2xhdw> [Accessed 1 April 2020].

[2] Brownlee, J., 2020. *How To Clean Text For Machine Learning With Python*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/clean-text-machine-learning-python/> [Accessed 2 April 2020].

[3] Brownlee, J., 2020. *How To Clean Text For Machine Learning With Python*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/clean-text-machine-learning-python/> [Accessed 2 April 2020].

APPENDIX

The code for the implementation of the methods is here. This code has two classes. Methods are implemented in the class “Operations”. Main flow of the operations are in the class “Main”.

WARRNING: This code is written assuming the text files are in the directories given in the following hierarchy and names:

-NLPoroject

* Books
  - Author
  - Genre
  - MergedBooks
* Frequencies
  - AfterStopWord
  - BeforeStopWord
  - MergedFrequencies

CODE:

MAIN CLASS:

```
from Operations import Operations
authorBasedBooks = ["Author/Dostoyevski-Idiot.txt", "Author/Dostoyevski-KaramazovBrothers.txt" , "Author/Dostoyevski-Possessed.txt",
                    "Author/CharlesDickens-BleakHouse.txt", "Author/CharlesDickens-Dombey&Son.txt", "Author/CharlesDickens-OurMutualFriend.txt",
                    "Author/Tolstoy-AnnaKarenina.txt", "Author/Tolstoy-Resurrection.txt", "Author/Tolstoy-War&Peace.txt"]
genreBasedBooks = ["Genre/Detective-Moonstone.txt", "Genre/Detective-Sherlock.txt", "Genre/Detective-TwentyYearsDetective.txt",
                   "Genre/Biography-Life&LettersofRobertBrowning.txt", "Genre/Biography-LifeofFroude.txt", "Genre/Biography-LifeofWilliamEwart.txt",
                   "Genre/Myths-Assyria.txt", "Genre/Myths-China.txt", "Genre/Myths-Rome.txt" ]

authorBasedFrequencyTexts = ["Freq_Dostoyevski-Idiot.txt", "Freq_Dostoyevski-KaramazovBrothers.txt" , "Freq_Dostoyevski-Possessed.txt",
                    "Freq_CharlesDickens-BleakHouse.txt", "Freq_CharlesDickens-Dombey&Son.txt", "Freq_CharlesDickens-OurMutualFriend.txt",
                    "Freq_Tolstoy-AnnaKarenina.txt", "Freq_Tolstoy-Resurrection.txt", "Freq_Tolstoy-War&Peace.txt"]
genreBasedFrequencyTexts = ["Freq_Detective-Moonstone.txt", "Freq_Detective-Sherlock.txt", "Freq_Detective-TwentyYearsDetective.txt",
                   "Freq_Biography-Life&LettersofRobertBrowning.txt", "Freq_Biography-LifeofFroude.txt", "Freq_Biography-LifeofWilliamEwart.txt",
                   "Freq_Myths-Assyria.txt", "Freq_Myths-China.txt", "Freq_Myths-Rome.txt" ]

Corpora_authorBased = []
Corpora_genreBased = []
CorporaWOStopWords_authorBased = []
CorporaWOStopWords_genreBased = []

TypeFrequencies_authorBasedCorpora = []
TypeFrequencies_genreBasedCorpora = []
TypeFrequencies_authorBasedCorporaWOStopWords = []
TypeFrequencies_genreBasedCorporaWOStopWords = []

op = Operations()

########## PART C
for i in range(9):
    Corpora_authorBased.append(op.createCorpora(authorBasedBooks[i]))

for i in range(9):
    Corpora_genreBased.append(op.createCorpora(genreBasedBooks[i]))

########## PART D
for i in range(9):

CorporaWOStopWords_authorBased.append(op.removeStopWords(Corpora_authorBased[i]))

for i in range(9):
```

```
CorporaWOStopWords_genreBased.append(op.removeStopWords(Corpora_genreBased[i]))
########## PART E
for i in range(9):

TypeFrequencies_authorBasedCorpora.append(op.countTypeFrequencies(Corpora_authorBased[i]))

TypeFrequencies_genreBasedCorpora.append(op.countTypeFrequencies(Corpora_genreBased[i]))

TypeFrequencies_authorBasedCorporaWOStopWords.append(op.countTypeFrequencies(CorporaWOStopWords_authorBased[i]))

TypeFrequencies_genreBasedCorporaWOStopWords.append(op.countTypeFrequencies(CorporaWOStopWords_genreBased[i]))

op.writeFrequenciesToText(authorBasedFrequencyTexts,
TypeFrequencies_authorBasedCorpora, "BeforeStopWord/AuthorBasedFrequencies/")
op.writeFrequenciesToText(genreBasedFrequencyTexts,
TypeFrequencies_genreBasedCorpora, "BeforeStopWord/GenreBasedFrequencies/")
op.writeFrequenciesToText(authorBasedFrequencyTexts,
TypeFrequencies_authorBasedCorporaWOStopWords,
"AfterStopWord/AuthorBasedFrequencies/")
op.writeFrequenciesToText(genreBasedFrequencyTexts,
TypeFrequencies_genreBasedCorporaWOStopWords,
"AfterStopWord/GenreBasedFrequencies/")

########## PART F
mergedBook =
op.mergeCorpora(authorBasedBooks[0],authorBasedBooks[1],authorBasedBooks[2])
with open ("C:/Users/enesk/PycharmProjects/NLPminiProject/Books/MergedBooks/" +
'MergedBook_Dostoyevski.txt', 'w', encoding="utf8") as fp:
    fp.write(mergedBook)

mergedBook = op.mergeCorpora(authorBasedBooks[3], authorBasedBooks[4],
authorBasedBooks[5])
with open("C:/Users/enesk/PycharmProjects/NLPminiProject/Books/MergedBooks/" +
'MergedBook_CharlesDickens.txt', 'w',encoding="utf8") as fp:
    fp.write(mergedBook)

mergedBook = op.mergeCorpora(authorBasedBooks[6], authorBasedBooks[7],
authorBasedBooks[8])
with open("C:/Users/enesk/PycharmProjects/NLPminiProject/Books/MergedBooks/" +
'MergedBook_Tolstoy.txt', 'w',encoding="utf8") as fp:
    fp.write(mergedBook)

mergedCorporaDostoyevski =
op.createCorpora("/MergedBooks/MergedBook_Dostoyevski.txt")
mergedFreqDostoyevski = op.countTypeFrequencies(mergedCorporaDostoyevski)
op.writeFrequencyToText("MergedFreq_Dostoyevski" ,mergedFreqDostoyevski,
"MergedFrequencies/" )

mergedCorporaCharles =
op.createCorpora("/MergedBooks/MergedBook_CharlesDickens.txt")
mergedFreqCharles = op.countTypeFrequencies(mergedCorporaCharles)
op.writeFrequencyToText("MergedFreq_CharlesDickens" ,mergedFreqCharles,
"MergedFrequencies/" )

mergedCorporaTolstoy = op.createCorpora("/MergedBooks/MergedBook_Tolstoy.txt")
```

```
mergedFreqTolstoy = op.countTypeFrequencies(mergedCorporaTolstoy)
op.writeFrequencyToText("MergedFreq_Tolstoy" ,mergedFreqTolstoy,
"MergedFrequencies/" )

op.plotLogZipfsForThree(mergedFreqTolstoy, mergedFreqDostoyevski,
mergedFreqCharles, "Tolstoy", "Dostoyevski", "Charles")
op.plotNormalZipfsForThree(mergedFreqTolstoy, mergedFreqDostoyevski,
mergedFreqCharles, "Tolstoy", "Dostoyevski", "Charles")


op.plotLogZipfsForThree(TypeFrequencies_authorBasedCorpora[0],
                        TypeFrequencies_authorBasedCorpora[1],
                        TypeFrequencies_authorBasedCorpora[2],
                        "Idiot", "Karamazov Brothers", "Possessed")

op.plotLogZipfsForThree(TypeFrequencies_authorBasedCorpora[3],
                        TypeFrequencies_authorBasedCorpora[4],
                        TypeFrequencies_authorBasedCorpora[5],
                        "Bleak House", "Dombey & House", "Our Mutual Friend")

op.plotLogZipfsForThree(TypeFrequencies_authorBasedCorpora[6],
                        TypeFrequencies_authorBasedCorpora[7],
                        TypeFrequencies_authorBasedCorpora[8],
                        "AnnaKarenina", "Resurrection", "WarPeace")


########## PART G
op.plotNormalTokenVsVocabularySizeForThree(mergedCorporaCharles,
mergedCorporaDostoyevski, mergedCorporaTolstoy,
                                     "Charles", "Dostoyevski", "Tolstoy")
op.plotLogTokenVsVocabularySizeForThree(mergedCorporaCharles,
mergedCorporaDostoyevski, mergedCorporaTolstoy,
                                     "Charles", "Dostoyevski", "Tolstoy", False)
########## PART H
op.plotLogTokenVsVocabularySizeForThree(Corpora_authorBased[0],
Corpora_authorBased[1], Corpora_authorBased[2],
                                     "Idiot", "Karamazov Brothers", "Possessed",
False)
op.plotLogTokenVsVocabularySizeForThree(Corpora_authorBased[3],
Corpora_authorBased[4], Corpora_authorBased[5],
                                     "Bleak House", "Dombey & House", "Our Mutual
Friend", False)
op.plotLogTokenVsVocabularySizeForThree(Corpora_authorBased[6],
Corpora_authorBased[7], Corpora_authorBased[8],
                                     "AnnaKarenina", "Resurrection", "WarPeace",
False)

op.plotLogTokenVsVocabularySizeForNine(Corpora_authorBased[0],
Corpora_authorBased[1], Corpora_authorBased[2],
                                    Corpora_authorBased[3],
Corpora_authorBased[4], Corpora_authorBased[5],
                                    Corpora_authorBased[6],
Corpora_authorBased[7], Corpora_authorBased[8],
                                    "Dostoyevski", "Charles", "Tolstoy", False)

op.plotLogTokenVsVocabularySizeForThree(mergedCorporaCharles,
mergedCorporaDostoyevski, mergedCorporaTolstoy,
                                     "Dostoyevski", "Charles", "Tolstoy", False)
```

```
########## PART I
op.plotLogTokenVsVocabularySizeForNine(Corpora_authorBased[0],
Corpora_authorBased[1], Corpora_authorBased[2],
                                    Corpora_authorBased[3],
Corpora_authorBased[4], Corpora_authorBased[5],
                                    Corpora_authorBased[6],
Corpora_authorBased[7], Corpora_authorBased[8],
                                    "Dostoyevski", "Charles", "Tolstoy", True)

op.plotLogTokenVsVocabularySizeForThree(mergedCorporaCharles,
mergedCorporaDostoyevski, mergedCorporaTolstoy,
                                     "Dostoyevski", "Charles", "Tolstoy", True)

########## PART J
mergedBook =
op.mergeCorpora(genreBasedBooks[0],genreBasedBooks[1],genreBasedBooks[2])
with open ("C:/Users/enesk/PycharmProjects/NLPminiProject/Books/MergedBooks/" +
'MergedBook_Detective.txt', 'w', encoding="utf8") as fp:
    fp.write(mergedBook)

mergedBook = op.mergeCorpora(genreBasedBooks[3], genreBasedBooks[4],
genreBasedBooks[5])
with open("C:/Users/enesk/PycharmProjects/NLPminiProject/Books/MergedBooks/" +
'MergedBook_Biography.txt', 'w',encoding="utf8") as fp:
    fp.write(mergedBook)

mergedBook = op.mergeCorpora(genreBasedBooks[6], genreBasedBooks[7],
genreBasedBooks[8])
with open("C:/Users/enesk/PycharmProjects/NLPminiProject/Books/MergedBooks/" +
'MergedBook_Myths.txt', 'w',encoding="utf8") as fp:
    fp.write(mergedBook)

mergedCorporaDetective = op.createCorpora("/MergedBooks/MergedBook_Detective.txt")
mergedFreqDetective = op.countTypeFrequencies(mergedCorporaDetective)
op.writeFrequencyToText("MergedFreq_Detective" ,mergedFreqDetective,
"MergedFrequencies/" )

mergedCorporaBiography = op.createCorpora("/MergedBooks/MergedBook_Biography.txt")
mergedFreqBiography = op.countTypeFrequencies(mergedCorporaBiography)
op.writeFrequencyToText("MergedFreq_Biography" ,mergedFreqBiography,
"MergedFrequencies/" )

mergedCorporaMyths = op.createCorpora("/MergedBooks/MergedBook_Myths.txt")
mergedFreqMyths = op.countTypeFrequencies(mergedCorporaMyths)
op.writeFrequencyToText("MergedFreq_Myths" ,mergedFreqMyths, "MergedFrequencies/"
)

##############

op.plotLogTokenVsVocabularySizeForThree(Corpora_genreBased[0],
Corpora_genreBased[1], Corpora_genreBased[2],
                                     "Moonstone", "Sherlock",
"TwentyYearsDetective", False)
op.plotLogTokenVsVocabularySizeForThree(Corpora_genreBased[3],
Corpora_genreBased[4], Corpora_genreBased[5],
                                     "Robert Browning", "Froude", "William Ewart",
False)
op.plotLogTokenVsVocabularySizeForThree(Corpora_genreBased[6],
Corpora_genreBased[7], Corpora_genreBased[8],
```

```
                                     "Assyria", "China", "Rome", False)

op.plotLogTokenVsVocabularySizeForNine(Corpora_genreBased[0],
Corpora_genreBased[1], Corpora_genreBased[2],
                                    Corpora_genreBased[3], Corpora_genreBased[4],
Corpora_genreBased[5],
                                    Corpora_genreBased[6], Corpora_genreBased[7],
Corpora_genreBased[8],
                                    "Detective", "Biography", "Myths", False)

op.plotLogTokenVsVocabularySizeForThree(mergedCorporaDetective,
mergedCorporaBiography, mergedCorporaMyths,
                                     "Detective", "Biography", "Myths", False)

op.plotLogTokenVsVocabularySizeForNine(Corpora_genreBased[0],
Corpora_genreBased[1], Corpora_genreBased[2],
                                    Corpora_genreBased[3], Corpora_genreBased[4],
Corpora_genreBased[5],
                                    Corpora_genreBased[6], Corpora_genreBased[7],
Corpora_genreBased[8],
                                    "Detective", "Biography", "Myths", True)

op.plotLogTokenVsVocabularySizeForThree(mergedCorporaDetective,
mergedCorporaBiography, mergedCorporaMyths,
                                     "Detective", "Biography", "Myths", True)

##### PART L
op.plotLogTokenVsVocabularySizeForNine(op.removeStopWords(Corpora_authorBased[0]),
                                       op.removeStopWords(Corpora_authorBased[1]),
                                       op.removeStopWords(Corpora_authorBased[2]),
                                       op.removeStopWords(Corpora_authorBased[3]),
                                       op.removeStopWords(Corpora_authorBased[4]),
                                       op.removeStopWords(Corpora_authorBased[5]),
                                       op.removeStopWords(Corpora_authorBased[6]),
                                       op.removeStopWords(Corpora_authorBased[7]),
                                       op.removeStopWords(Corpora_authorBased[8]),
                                       "Dosto", "Charles", "Tolstoy", True)

op.plotLogTokenVsVocabularySizeForThree(op.removeStopWords(mergedCorporaDostoyevsk
i),
                                        op.removeStopWords(mergedCorporaCharles),
                                        op.removeStopWords(mergedCorporaTolstoy),
                                     "Dosto", "Charles", "Tolstoy", True)

op.plotLogTokenVsVocabularySizeForNine(op.removeStopWords(Corpora_genreBased[0]),
                                       op.removeStopWords(Corpora_genreBased[1]),
                                       op.removeStopWords(Corpora_genreBased[2]),
                                       op.removeStopWords(Corpora_genreBased[3]),
                                       op.removeStopWords(Corpora_genreBased[4]),
                                       op.removeStopWords(Corpora_genreBased[5]),
                                       op.removeStopWords(Corpora_genreBased[6]),
                                       op.removeStopWords(Corpora_genreBased[7]),
                                       op.removeStopWords(Corpora_genreBased[8]),
                                       "Detective", "Bio", "Myths", True)

op.plotLogTokenVsVocabularySizeForThree(op.removeStopWords(mergedCorporaDetective)
,

op.removeStopWords(mergedCorporaBiography),
```

```
                                        op.removeStopWords(mergedCorporaMyths),
                                        "Detective", "Bio", "Myths",True)

########### PART M
randomText = op.createRandomText()
randomTextCorpora = op.createRandomCorpora(randomText)
print (randomTextCorpora)
frequenciesOfRandomText = op.countTypeFrequencies(randomTextCorpora)
print (frequenciesOfRandomText)
op.plotNormalZipfsForOne(frequenciesOfRandomText)
op.plotLogZipfsForOne(frequenciesOfRandomText)
op.plotNormalTokenVsVocabularySizeForOne(randomTextCorpora)
op.plotLogTokenVsVocabularySizeForOne(randomTextCorpora, False)
    # REFERENCES
    # https://machinelearningmastery.com/clean-text-machine-learning-python/
    # https://gist.github.com/larsyencken/1440509 -> stop words
```

OPERATIONS CLASS:

```
class Operations:
    def plotNormalTokenVsVocabularySizeForOne(self, Corpora1):
        import matplotlib.pyplot as plt
        plt.xlabel('Number of Tokens in Corpora')
        plt.ylabel('Number of Types')
        plt.title("Token vs Type Size")

        import math
        x = []
        y = []
        for i in range(math.floor(len(Corpora1) / 10000)):
            x.append(10000 * (i + 1))
            y.append(len(self.countTypeFrequencies(Corpora1[1:10000 * (i + 1)])))
        print("x = ", x)
        print("\n")
        print("y = ", y)
        plt.plot(x, y, "r")
        plt.show()


    def plotLogTokenVsVocabularySizeForOne(self, Corpora1, showBestFitLine):
        import matplotlib.pyplot as plt
        import numpy
        plt.xlabel('Log (Number of Tokens in Corpora)')
        plt.ylabel('Log (Number of Types)')
        plt.title("Token vs Type Size")
        import math
        x = []
        y = []
        for i in range(1, math.floor(len(Corpora1)/10000)):
            x.append(10000*(i+1))
            y.append(len(self.countTypeFrequencies(Corpora1[1:10000*(i+1)])))
        print("x = " , x)
        print("\n")
        print("y = ", y)
        plt.plot(numpy.log(x), numpy.log(y), "r--")
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
```

```
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m*numpy.log(x)+b, "r", label = "Best Fitting
Line. Slope is: " + str(round(m,4)))
        plt.show()


    def plotLogZipfsForOne(self, TypeFrequencies1):
        import numpy
        freqArray = []
        for letter, count in TypeFrequencies1.most_common(len(TypeFrequencies1)):
            freqArray.append(count)
        x = []
        y = []
        import matplotlib.pyplot as plt
        plt.figure(2)
        for i in range(1, len(freqArray)):
            x.append(i)
            y.append(freqArray[i])
        plt.xlabel('log(Rank of Type)')
        plt.ylabel('log(Frequency)')
        plt.title("Zipf's Law Plot")
        plt.plot(numpy.log(x), numpy.log(y), 'g')
        plt.show()

    def plotNormalZipfsForOne(self, TypeFrequencies1 ):

      import math
      freqArray = []
      for letter, count in TypeFrequencies1.most_common(len(TypeFrequencies1)):
          freqArray.append(count)
      x = []
      y = []
      import matplotlib.pyplot as plt
      plt.figure(1)
      for i in range(1, math.floor(len(freqArray)/300)):
          x.append(i)
          y.append(freqArray[i])
      plt.xlabel('Rank of Type')
      plt.ylabel('Frequency')
      plt.title("Zipf's Law Plot")
      plt.plot(x, y, 'g' )
      plt.show()

    def createRandomText(self):
        import string
        import random
        from random import seed
        from random import randint
        seed(1)

        numberOfWords = 1500000
        maxNumberofLettersInaWord = 4

        text = ""

        for i in range(numberOfWords):
                numberOfLetters = randint(1, maxNumberofLettersInaWord)
                text = text + " " +
str(''.join(random.choices(string.ascii_uppercase +
```

```
                                                string.digits, k=numberOfLetters)))

        return text

    def plotLogTokenVsVocabularySizeForNine(self, Corpora1, Corpora2,
Corpora3,Corpora4, Corpora5, Corpora6,
                                            Corpora7, Corpora8, Corpora9,
                                            Corpora1Name, Corpora2Name,
Corpora3Name , showBestFitLine):
        import matplotlib.pyplot as plt
        import numpy

        plt.xlabel('Log (Number of Tokens in Corpora)')
        plt.ylabel('Log (Number of Types)')
        plt.title("Token vs Type Size")

        import math
        x = []
        y = []
        for i in range(1, math.floor(len(Corpora1)/10000)):
            x.append(10000*(i+1))
            y.append(len(self.countTypeFrequencies(Corpora1[1:10000*(i+1)])))
        print("x = " , x)
        print("\n")
        print("y = ", y)
        plt.plot(numpy.log(x), numpy.log(y), "r--", label = Corpora1Name + " -
Idiot")
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m * numpy.log(x) + b, "r",label = "Best Fitting
Line. Slope is: " + str(round(m,4)))

        x = []
        y = []
        for i in range(1, math.floor(len(Corpora2)/10000)):
            x.append(10000*(i+1))
            y.append(len(self.countTypeFrequencies(Corpora2[1:10000*(i+1)])))
        print("x = " , x)
        print("\n")
        print("y = " , y)
        plt.plot(numpy.log(x), numpy.log(y), "r.", label = Corpora1Name + " -
Karamazov Brothers")
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m * numpy.log(x) + b, "r",label = "Best Fitting
Line. Slope is: " + str(round(m,4)))

        x = []
        y = []
        for i in range(1, math.floor(len(Corpora3)/10000)):
            x.append(10000*(i+1))
            y.append(len(self.countTypeFrequencies(Corpora3[1:10000*(i+1)])))
        print("x = " , x)
        print("\n")
        print("y = " , y)
```

```
        plt.plot(numpy.log(x), numpy.log(y), "r*", label = Corpora1Name + " - 
Possessed")
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m * numpy.log(x) + b, "r",label = "Best Fitting 
Line. Slope is: " + str(round(m,4)))

        x = []
        y = []
        for i in range(1, math.floor(len(Corpora4) / 10000)):
            x.append(10000 * (i + 1))
            y.append(len(self.countTypeFrequencies(Corpora4[1:10000 * (i + 1)])))
        print("x = ", x)
        print("\n")
        print("y = ", y)
        plt.plot(numpy.log(x), numpy.log(y), "g--", label=Corpora2Name + " - Bleak 
House")
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m * numpy.log(x) + b, "g",label = "Best Fitting 
Line. Slope is: " + str(round(m,4)))

        x = []
        y = []
        for i in range(1, math.floor(len(Corpora5) / 10000)):
            x.append(10000 * (i + 1))
            y.append(len(self.countTypeFrequencies(Corpora5[1:10000 * (i + 1)])))
        print("x = ", x)
        print("\n")
        print("y = ", y)
        plt.plot(numpy.log(x), numpy.log(y), "g.", label=Corpora2Name + " - Dombey 
& Son")
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m * numpy.log(x) + b, "g",label = "Best Fitting 
Line. Slope is: " + str(round(m,4)))

        x = []
        y = []
        for i in range(1, math.floor(len(Corpora6) / 10000)):
            x.append(10000 * (i + 1))
            y.append(len(self.countTypeFrequencies(Corpora6[1:10000 * (i + 1)])))
        print("x = ", x)
        print("\n")
        print("y = ", y)
        plt.plot(numpy.log(x), numpy.log(y), "g*", label=Corpora2Name + " - Our 
Mutual Friend" )
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m * numpy.log(x) + b, "g",label = "Best Fitting 
Line. Slope is: " + str(round(m,4)))
```

```
        x = []
        y = []
        for i in range(1, math.floor(len(Corpora7) / 10000)):
            x.append(10000 * (i + 1))
            y.append(len(self.countTypeFrequencies(Corpora7[1:10000 * (i + 1)])))
        print("x = ", x)
        print("\n")
        print("y = ", y)
        plt.plot(numpy.log(x), numpy.log(y), "b--", label=Corpora3Name + " - Anna 
Karenine")
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m * numpy.log(x) + b, "b",label = "Best Fitting 
Line. Slope is: " + str(round(m,4)))

        x = []
        y = []
        for i in range(1, math.floor(len(Corpora8) / 10000)):
            x.append(10000 * (i + 1))
            y.append(len(self.countTypeFrequencies(Corpora8[1:10000 * (i + 1)])))
        print("x = ", x)
        print("\n")
        print("y = ", y)
        plt.plot(numpy.log(x), numpy.log(y), "b.", label=Corpora3Name + " - 
Resurrection")
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m * numpy.log(x) + b, "b",label = "Best Fitting 
Line. Slope is: " + str(round(m,4)))

        x = []
        y = []
        for i in range(1, math.floor(len(Corpora9) / 10000)):
            x.append(10000 * (i + 1))
            y.append(len(self.countTypeFrequencies(Corpora9[1:10000 * (i + 1)])))
        print("x = ", x)
        print("\n")
        print("y = ", y)
        plt.plot(numpy.log(x), numpy.log(y), "b*", label=Corpora3Name + " - War & 
Peace")
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m*numpy.log(x)+b, "b",label = "Best Fitting 
Line. Slope is: " + str(round(m,4)))

        plt.legend()
        plt.show()



    def plotLogTokenVsVocabularySizeForThree(self, Corpora1, Corpora2, Corpora3, 
Corpora1Name,
```

```
                                                        Corpora2Name, Corpora3Name,
showBestFitLine):
        import matplotlib.pyplot as plt
        import numpy

        plt.xlabel('Log (Number of Tokens in Corpora)')
        plt.ylabel('Log (Number of Types)')
        plt.title("Token vs Type Size")

        import math
        x = []
        y = []
        for i in range(1, math.floor(len(Corpora1)/10000)):
            x.append(10000*(i+1))
            y.append(len(self.countTypeFrequencies(Corpora1[1:10000*(i+1)])))
        print("x = " , x)
        print("\n")
        print("y = ", y)
        plt.plot(numpy.log(x), numpy.log(y), "r--", label = Corpora1Name)
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m*numpy.log(x)+b, "r", label = "Best Fitting
Line. Slope is: " + str(round(m,4)))

        x = []
        y = []
        for i in range(1, math.floor(len(Corpora2)/10000)):
            x.append(10000*(i+1))
            y.append(len(self.countTypeFrequencies(Corpora2[1:10000*(i+1)])))
        print("x = " , x)
        print("\n")
        print("y = " , y)
        plt.plot(numpy.log(x), numpy.log(y), "g--", label = Corpora2Name)
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m*numpy.log(x)+b, "g", label = "Best Fitting
Line. Slope is: " + str(round(m,4)))

        x = []
        y = []
        for i in range(1, math.floor(len(Corpora3)/10000)):
            x.append(10000*(i+1))
            y.append(len(self.countTypeFrequencies(Corpora3[1:10000*(i+1)])))
        print("x = " , x)
        print("\n")
        print("y = " , y)
        plt.plot(numpy.log(x), numpy.log(y), "b--", label = Corpora3Name)
        if showBestFitLine:
            x = numpy.array(x)
            y = numpy.array(y)
            m, b = numpy.polyfit(numpy.log(x), numpy.log(y), 1)
            plt.plot(numpy.log(x), m*numpy.log(x)+b, "b", label = "Best Fitting
Line. Slope is: " + str(round(m,4)))

        plt.legend()
```

```
        plt.show()

    def plotNormalTokenVsVocabularySizeForThree(self, Corpora1, Corpora2,
Corpora3, Corpora1Name, Corpora2Name, Corpora3Name):
        import matplotlib.pyplot as plt
        plt.xlabel('Number of Tokens in Corpora')
        plt.ylabel('Number of Types')
        plt.title("Token vs Type Size")

        import math
        x = []
        y = []
        for i in range(math.floor(len(Corpora1)/10000)):
            x.append(10000*(i+1))
            y.append(len(self.countTypeFrequencies(Corpora1[1:10000*(i+1)])))
        print("x = " , x)
        print("\n")
        print("y = " , y)
        plt.plot(x, y, "r", label = Corpora1Name)

        x = []
        y = []
        for i in range(math.floor(len(Corpora2)/10000)):
            x.append(10000*(i+1))
            y.append(len(self.countTypeFrequencies(Corpora2[1:10000*(i+1)])))
        print("x = " , x)
        print("\n")
        print("y = " , y)
        plt.plot(x, y, "g", label = Corpora2Name)

        x = []
        y = []
        for i in range(math.floor(len(Corpora3)/10000)):
            x.append(10000*(i+1))
            y.append(len(self.countTypeFrequencies(Corpora3[1:10000*(i+1)])))
        print("x = " , x)
        print("\n")
        print("y = " , y)
        plt.plot(x, y, "b", label = Corpora3Name)
        plt.legend()
        plt.show()


    def plotNormalZipfsForThree(self, TypeFrequencies1,TypeFrequencies2,
TypeFrequencies3, name1, name2, name3 ):

      import math

      freqArray = []
      for letter, count in TypeFrequencies1.most_common(len(TypeFrequencies1)):
          freqArray.append(count)
      x = []
      y = []
      import matplotlib.pyplot as plt
      plt.figure(1)
      for i in range(1, math.floor(len(freqArray)/300)):
          x.append(i)
          y.append(freqArray[i])
      plt.xlabel('Rank of Type')
```

```
        plt.ylabel('Frequency')
        plt.title("Zipf's Law Plot")
        plt.plot(x, y, 'g', label = name1 )

        freqArray = []
        for letter, count in TypeFrequencies2.most_common(len(TypeFrequencies2)):
            freqArray.append(count)
        x = []
        y = []
        for i in range(1, math.floor(len(freqArray)/300)):
            x.append(i)
            y.append(freqArray[i])
        plt.plot(x, y, 'r', label = name2)

        freqArray = []
        for letter, count in TypeFrequencies3.most_common(len(TypeFrequencies3)):
            freqArray.append(count)
        x = []
        y = []
        for i in range(1, math.floor(len(freqArray)/300)):
            x.append(i)
            y.append(freqArray[i])
        plt.plot(x, y, 'b', label = name3)
        plt.legend()
        plt.show()

    def plotLogZipfsForThree(self, TypeFrequencies1,TypeFrequencies2,
TypeFrequencies3, name1, name2, name3 ):

        import numpy
        freqArray = []
        for letter, count in TypeFrequencies1.most_common(len(TypeFrequencies1)):
            freqArray.append(count)
        x = []
        y = []
        import matplotlib.pyplot as plt
        plt.figure(2)
        for i in range(1, len(freqArray)):
            x.append(i)
            y.append(freqArray[i])
        plt.xlabel('log(Rank of Type)')
        plt.ylabel('log(Frequency)')
        plt.title("Zipf's Law Plot")
        plt.plot(numpy.log(x), numpy.log(y), 'g', label = name1 )

        freqArray = []
        for letter, count in TypeFrequencies2.most_common(len(TypeFrequencies2)):
            freqArray.append(count)
        x = []
        y = []
        for i in range(1, len(freqArray)):
            x.append(i)
            y.append(freqArray[i])
        plt.plot(numpy.log(x), numpy.log(y), 'r', label = name2)

        freqArray = []
        for letter, count in TypeFrequencies3.most_common(len(TypeFrequencies3)):
            freqArray.append(count)
        x = []
```

```
        y = []
        for i in range(1, len(freqArray)):
            x.append(i)
            y.append(freqArray[i])
        plt.plot(numpy.log(x), numpy.log(y), 'b', label = name3)
        plt.legend()
        plt.show()

    def mergeCorpora(self, book1, book2, book3):
        file1 = open("C:/Users/enesk/PycharmProjects/NLPminiProject/Books/" + book1,
encoding="utf8")
        text1 = file1.read()
        file1.close()
        file2 = open("C:/Users/enesk/PycharmProjects/NLPminiProject/Books/" + book2,
encoding="utf8")
        text2 = file2.read()
        file2.close()
        file3 = open("C:/Users/enesk/PycharmProjects/NLPminiProject/Books/" + book3,
encoding="utf8")
        text3 = file3.read()
        file3.close()
        text1 += "/n"
        text1 += text2
        text1 += "/n"
        text1 += text3
        return text1
    def writeFrequenciesToText(self, names, frequencies, dirName):
        for i in range(9):
            with open("Frequencies/" + dirName + names[i], 'w', encoding="utf8") as
f:
                for k, v in frequencies[i].most_common():
                    f.write("{} {}\n".format(k, v))

    def writeFrequencyToText(self, name, frequency, dirName):
            with open("Frequencies/" + dirName + name, 'w', encoding="utf8") as f:
                for k, v in frequency.most_common():
                    f.write("{} {}\n".format(k, v))

    def createRandomCorpora (self, text):

        ##### split into words by white space
        words = text.split()
        # print(words[:100])
        ##### remove punctuation from each word
        import string
        table = str.maketrans('', '', string.punctuation)
        strippedWords = [w.translate(table) for w in words]
        # print(strippedWords[:100])

        ##### convert to lower case
        strippedWordsLower = [strippedWords.lower() for strippedWords in
strippedWords]
        # print(strippedWordsLower[:100])
        return strippedWordsLower

    def createCorpora (self, bookName):
        ##### load text
        file = open("C:/Users/enesk/PycharmProjects/NLPminiProject/Books/" +
bookName, encoding="utf8")
```

```
        text = file.read()
        # print(text)
        file.close()

        ##### split into words by white space
        words = text.split()
        # print(words[:100])
        ##### remove punctuation from each word
        import string
        table = str.maketrans('', '', string.punctuation)
        strippedWords = [w.translate(table) for w in words]
        # print(strippedWords[:100])

        ##### convert to lower case
        strippedWordsLower = [strippedWords.lower() for strippedWords in
strippedWords]
        # print(strippedWordsLower[:100])
        return strippedWordsLower

    def removeStopWords(self, strippedWordsLower):
        ##### remove stopwords
        filename = 'stopwords.txt'
        file = open(filename, 'r+')
        stopWords = file.read()
        file.close()
        strippedWordsWOStopWords = [w for w in strippedWordsLower if not w in
stopWords]
        return strippedWordsWOStopWords
        # print(strippedWordsWOStopWords[:100000])

    def countTypeFrequencies(self, strippedWords):
       from collections import Counter
       wordFrequencies = Counter(strippedWords)
       return wordFrequencies
       #print(wordFrequencies)
```