NLP PROJECT REPORT

Submitted by

TEAM SPARX

Sajal Sharma	20UCS168
Rugwed Bodhankar	20UCS164
Kaustubh Narwade	20UCS095
Vinayak Mishra	20UCS228

ACKNOWLEDGEMENT

We would especially want to thank Dr. Sakthi Balan, our project mentor, for giving us the chance to work on this fantastic project on the

topic of text analysis (NLP Project Round 1). This project allowed us conduct a lot of research, and it was a terrific learning opportunity.

DATE: 29 OCT 2022

LNMIIT
The LNM Institute of
Information Technology

TABLE OF CONTENTS

TITLE

LNMIIT
The LNM Institute of
Information Technology

ABSTRACT

In this project, we worked on text preprocessing, PoS tagging, and several other procedures from the book Understanding Cryptography by Christopher Paar and Jan Pelzl. Text analytics is a crucial component of the field of natural language processing. Because of the book's straightforward writing style and ease of comprehension, working on this project on it was quite engaging. We conducted all of the preprocessing, tokenization, and tagging for all the operations using the NLTK (natural language Tool Kit). The project also explained the book's frequency distribution and Word Cloud and aided in our understanding of a few text mining subfields. The graphs that are plotted in the report also say a lot about the input text which is derived from the book and writing style of the author, the words that he used frequently, main terms, definitive words etc. This project can also be used in understanding vocabulary in a certain text file, frequency of words, difficulty in the text file.

1. LITERATURE REVIEW

Many researchers worked on NLP, building tools and systems which make NLP easier to understand. Tools like Sentiment Analyzer, Parts of Speech (POS) Taggers, Chunking, Named Entity Recognition (NER), Semantic Role Labelling made NLP a good topic for research.

Related Work :

- Sentiment analyzer (Jeonghee et al.,2003) [26] works by extracting sentiments about a given topic.
- Arabic uses the Support Vector Machine (SVM) (Mona Diab et al.,2004) [29] approach to automatically tokenize, tag parts of speech, and annotate base phrases in Arabic text.
- The Sanskrit part of speech tagger specifically uses the treebank technique.
- Parts of speech taggers for the languages like European languages, research is being done on making parts of speech taggers for other languages like Arabic, Sanskrit (Namrata Tapswi, Suresh Jain , 2012) [27], Hindi (Pradipta Ranjan Ray et al., 2003)[28], etc. It can efficiently tag and classify words as nouns, adjectives, verbs, etc.

2. INTRODUCTION

In this project, we imported a text file of a book and used NLP techniques to perform text analysis on it. We

performed PoS tagging on the imported text file, tokenization and lemmatization, frequency analysis, and frequency

distribution analysis on the text file. We are now going to visualise the data, which will be an excellent learning

opportunity for NLP.

We have chosen book related to our course Computer Security. This book is among the top downloaded

books in field of cryptography.

- ***Understanding Cryptography written by Christopher Paar and Jen Pelzl***

For the objectives assigned in this project, we used NLTK, a collection of Python-written modules and tools for

symbolic and statistical natural language processing of English. Additionally, we have imported a few modules and

functions to carry out various tasks like preprocessing, tokenizing, deleting stop words, etc. And after carrying out

these procedures on the text file, we generated word clouds and showed the frequency distribution of the text file.

3. Python Libraries and Modules Used

Nltk.stem package : Interfaces used to remove morphological affixes from words, leaving only the word stem.

Nltk.corpus : The modules in this package provide functions that can be used to read corpus files in a variety of formats.

Collection modules : Provide different types of container data types**.**

Nltk.probability : A probability distribution specifies how likely it is that an experiment will have any given output.

Nltk.tokenizer package : Tokenizer divides strings into a list of substrings.

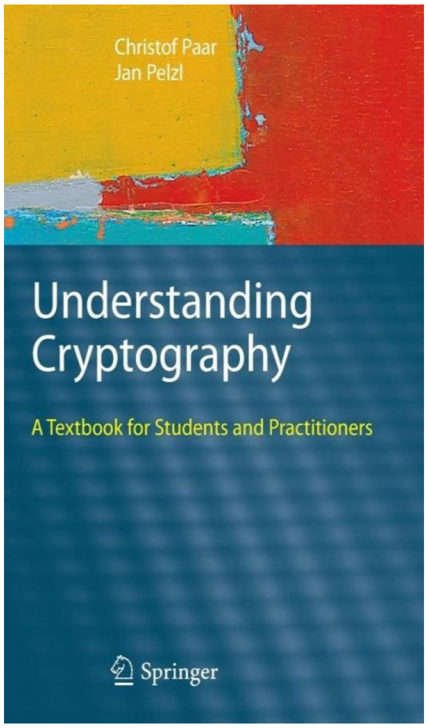
Wordcloud package : Provides modules to create wordcloud in python.

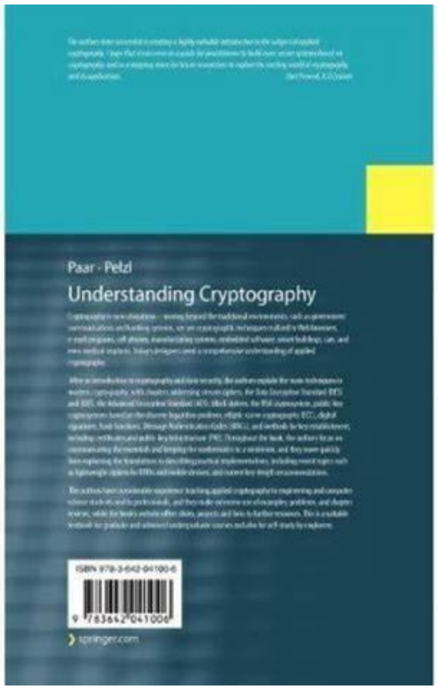
4. METHODOLOGY

Downloading Books :

● We have downloaded the book ***Understanding Cryptography written by Cristopher Paar and Jen Palzl*** in plain text format for text processing from (https://swarm.cs.pub.ro/~mbarbulescu/cripto/Understanding%20Cryptography%20by%20Christof%20 Paar%20.pdf)

The downloaded file is .txt file

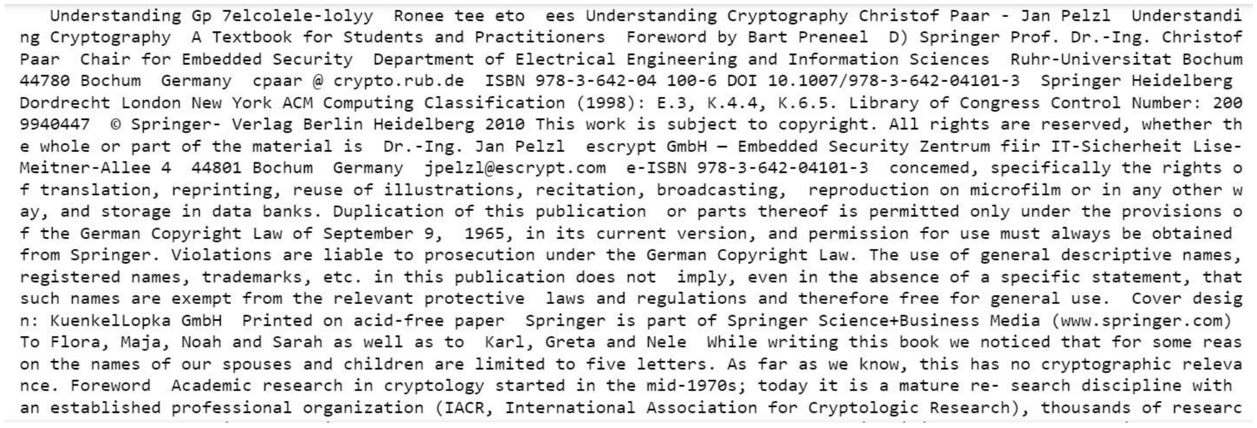

Christof Paar
Jan Pelzl
Understanding Cryptography
A Textbook for Students and Practitioners
Springer



Cover of the book shown above is selected for text analysis

Importing the text

In this step we created a function (txt_file_to_string) to read the text imported from the book and convert it into string for processing in python. This function takes as input the path of the file to be read and returns the content of the file in string format.



```
   Understanding Gp 7elcolele-lolyy  Ronee tee eto  ees Understanding Cryptography Christof Paar - Jan Pelzl  Understandi
ng Cryptography  A Textbook for Students and Practitioners  Foreword by Bart Preneel  D) Springer Prof. Dr.-Ing. Christof
Paar  Chair for Embedded Security  Department of Electrical Engineering and Information Sciences  Ruhr-Universitat Bochum
44780 Bochum  Germany  cpaar @ crypto.rub.de  ISBN 978-3-642-04 100-6 DOI 10.1007/978-3-642-04101-3  Springer Heidelberg
Dordrecht London New York ACM Computing Classification (1998): E.3, K.4.4, K.6.5. Library of Congress Control Number: 200
9940447  © Springer- Verlag Berlin Heidelberg 2010 This work is subject to copyright. All rights are reserved, whether th
e whole or part of the material is  Dr.-Ing. Jan Pelzl  escrypt GmbH — Embedded Security Zentrum fiir IT-Sicherheit Lise-
Meitner-Allee 4  44801 Bochum  Germany  jpelzl@escrypt.com  e-ISBN 978-3-642-04101-3  concemed, specifically the rights o
f translation, reprinting, reuse of illustrations, recitation, broadcasting,  reproduction on microfilm or in any other w
ay, and storage in data banks. Duplication of this publication  or parts thereof is permitted only under the provisions o
f the German Copyright Law of September 9,  1965, in its current version, and permission for use must always be obtained
from Springer. Violations are liable to prosecution under the German Copyright Law. The use of general descriptive names,
registered names, trademarks, etc. in this publication does not  imply, even in the absence of a specific statement, that
such names are exempt from the relevant protective  laws and regulations and therefore free for general use.  Cover desig
n: KuenkelLopka GmbH  Printed on acid-free paper  Springer is part of Springer Science+Business Media (www.springer.com)
To Flora, Maja, Noah and Sarah as well as to  Karl, Greta and Nele  While writing this book we noticed that for some reas
on the names of our spouses and children are limited to five letters. As far as we know, this has no cryptographic releva
nce. Foreword  Academic research in cryptology started in the mid-1970s; today it is a mature re- search discipline with
an established professional organization (IACR, International Association for Cryptologic Research), thousands of researc
```

Text before Preprocessing

Text Pre-Processing and Tokenization :

1. Removing prefix and suffix to narrow down to text from eBook - The book from the website had additional prefix and suffix attached in its .txt file, apart from the contents of the eBook.

2. Lowercase - We converted our string to lowercase using the string function string.lower().

3. Expansion of some Contractions - We expanded some generic contractions. For example : can't to can not, all instances of 'll to will , won't to will not etc. This is not very accurate and will lead to some incorrect expansion since disambiguation to the right expansion is not deterministic but this will be correct for most cases.

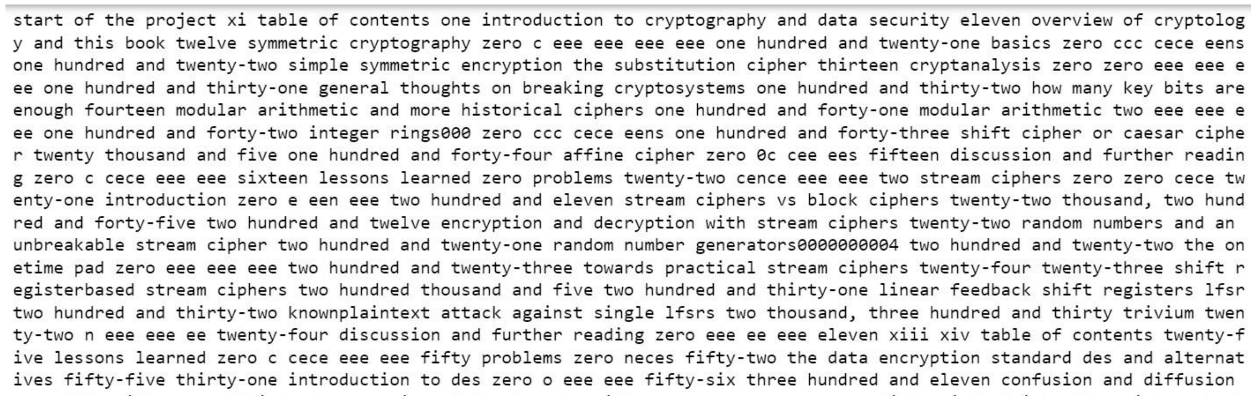
4. **4.** Removal Of Punctuations - We removed all the punctuation using regular expressions in two steps by first replacing everything other than word and whitespace characters with empty string and then replacing _ (underscore, which is considered part of word in python) by empty string.

5. Removing unnecessary repeated words

6. Replacing one or more continuous white space characters with single space to make the string evenly and correctly spaced.

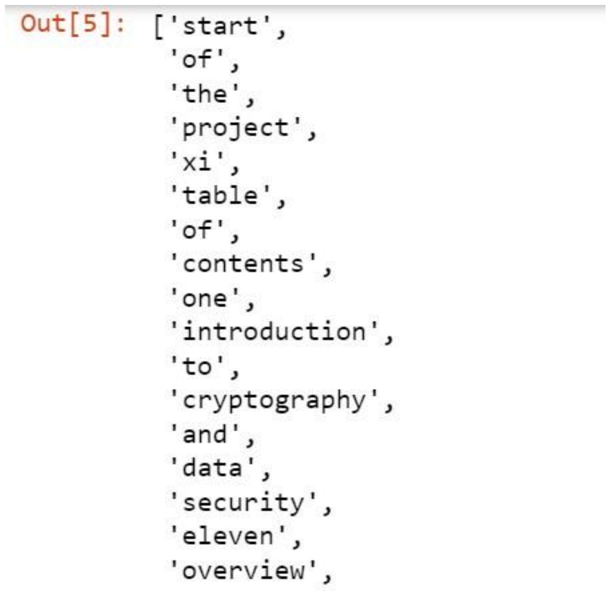
7. Replacing numbers from integer to word form.

8. We tokenized the string into single words using word_tokenize() function imported from NLTK and stored them. Then we lemmatized these lists as we plan to analyse word frequencies later, therefore reducing the words to their lemma form will be suitable.



```
start of the project xi table of contents one introduction to cryptography and data security eleven overview of cryptolog
y and this book twelve symmetric cryptography zero c eee eee eee eee one hundred and twenty-one basics zero ccc cece eens
one hundred and twenty-two simple symmetric encryption the substitution cipher thirteen cryptanalysis zero zero eee eee e
ee one hundred and thirty-one general thoughts on breaking cryptosystems one hundred and thirty-two how many key bits are
enough fourteen modular arithmetic and more historical ciphers one hundred and forty-one modular arithmetic two eee eee e
ee one hundred and forty-two integer rings000 zero ccc cece eens one hundred and forty-three shift cipher or caesar ciphe
r twenty thousand and five one hundred and forty-four affine cipher zero 0c cee ees fifteen discussion and further readin
g zero c cece eee eee sixteen lessons learned zero problems twenty-two cence eee eee two stream ciphers zero zero cece tw
enty-one introduction zero e een eee two hundred and eleven stream ciphers vs block ciphers twenty-two thousand, two hund
red and forty-five two hundred and twelve encryption and decryption with stream ciphers twenty-two random numbers and an
unbreakable stream cipher two hundred and twenty-one random number generators0000000004 two hundred and twenty-two the on
etime pad zero eee eee eee two hundred and twenty-three towards practical stream ciphers twenty-four twenty-three shift r
egisterbased stream ciphers two hundred thousand and five two hundred and thirty-one linear feedback shift registers lfsr
two hundred and thirty-two knownplaintext attack against single lfsrs two thousand, three hundred and thirty trivium twen
ty-two n eee eee ee twenty-four discussion and further reading zero eee ee eee eleven xiii xiv table of contents twenty-f
ive lessons learned zero c cece eee eee fifty problems zero neces fifty-two the data encryption standard des and alternat
ives fifty-five thirty-one introduction to des zero o eee eee fifty-six three hundred and eleven confusion and diffusion
```

Text after Preprocessing



```
Out[5]: ['start',
         'of',
         'the',
         'project',
         'xi',
         'table',
         'of',
         'contents',
         'one',
         'introduction',
         'to',
         'cryptography',
         'and',
         'data',
         'security',
         'eleven',
         'overview',
```

Tokenization before removing stop words

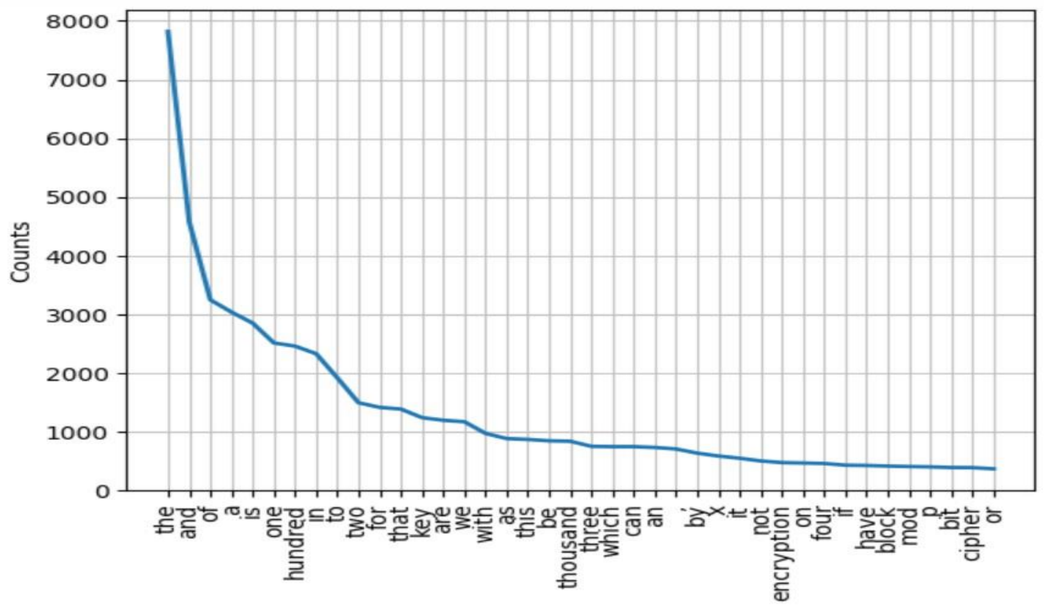
Plotting Frequency distribution of tokens :

● After tokenizing the text, we imported the function FreqDist() from the module nltk.probability which is helpful in probability calculations, where frequency distribution counts the number of times that each outcome of an experiment occurs. We stored the frequency distribution of the two strings in the FreqDist object by passing the tokenized and lemmatized lists to the above function.

```
FreqDist({'the': 7820, 'and': 4580, 'of': 3247, 'a': 3037, 'is': 2850, 'one': 2512, 'hundred': 2458, 'in': 2328, 'to': 1917,
'two': 1493, ...})
```

Frequency before removing stop words

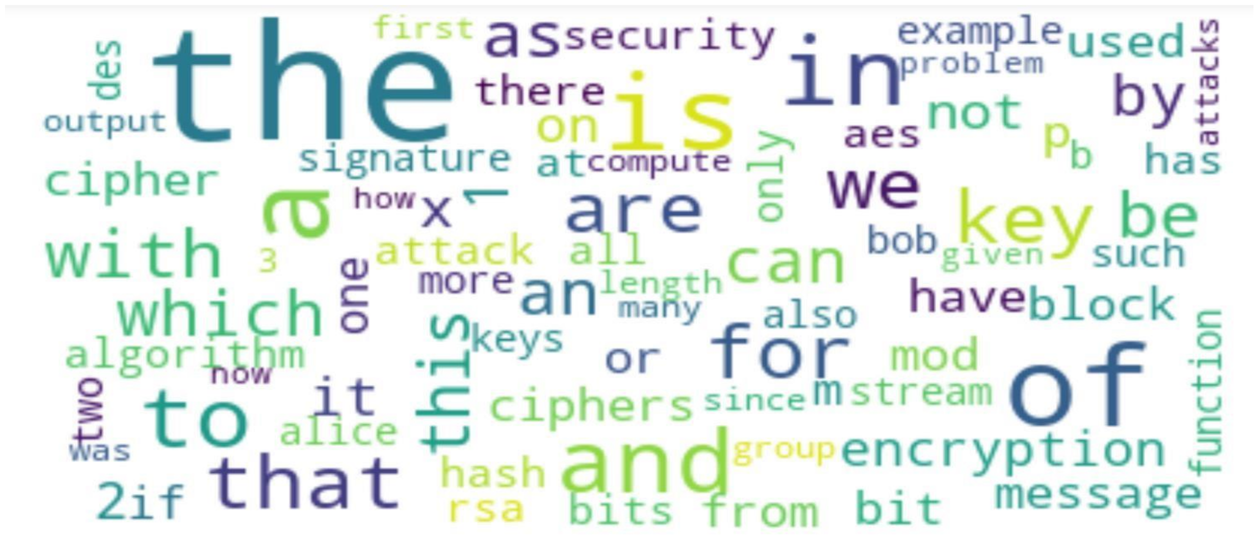
● For plotting the graph of frequency distribution we imported the function figure() from the module matplotlib.pyplot which is used to create a figure object. The whole figure is regarded as the figure object. Next we plotted frequency distribution graph:



Frequency distribution of top 40 words (with stop words)

Creating Word Cloud

- For creating word cloud we imported Counter from the module Collections. Then we imported wordcloud from the module wordcloud. Then we used functions plt.figure() , plt.show(), plt.axis() for the visualization of wordcloud. We made the word cloud for the top 80 most frequently used words which is attached below :



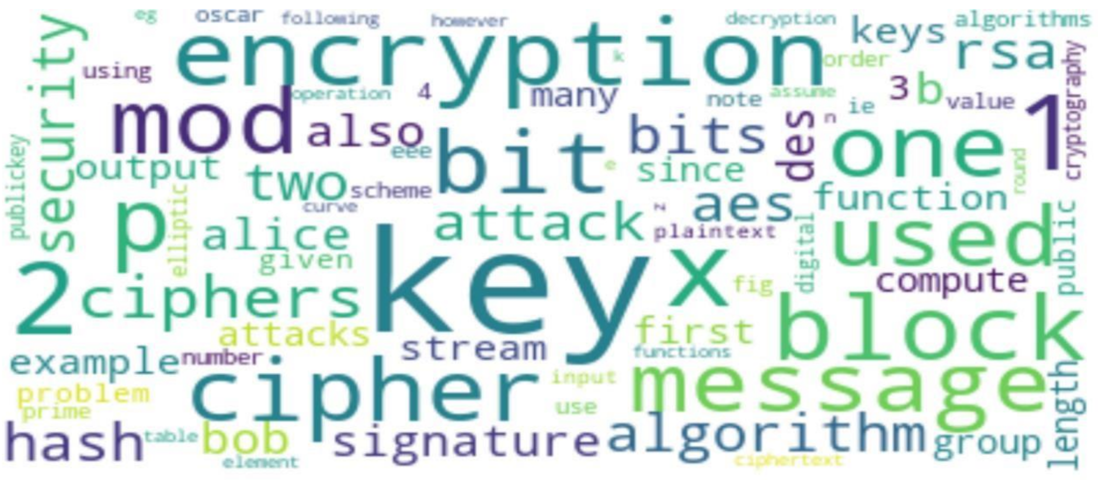
Word cloud (with stop words)

Inference :

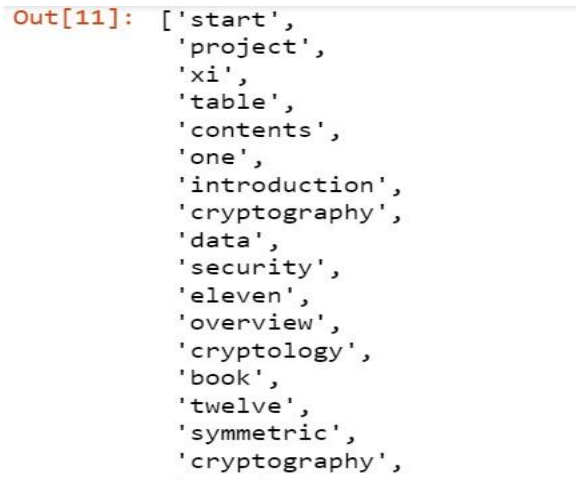
- It is giving the visual representation of the most frequent words in the book Understanding Cryptography
- We observe that words like the, and, of, a are among the words that appear bigger and their frequency is also high in the previous plot.
- Thus we can infer that the stop words occur in high frequency in the text.
- Infact, the overall word cloud is heavily dominated by stop words.

Removing stop words and creating word cloud again:

- We remove the stop words from the text. We used the nltk.corpus stopwords from English and removed them from the text. We updated the frequency distribution of words in the texts and repeated the above process to obtain a frequency ditribution, tokenization, frequency and Word cloud without stop words which is attached below :



Word cloud (without stop words)

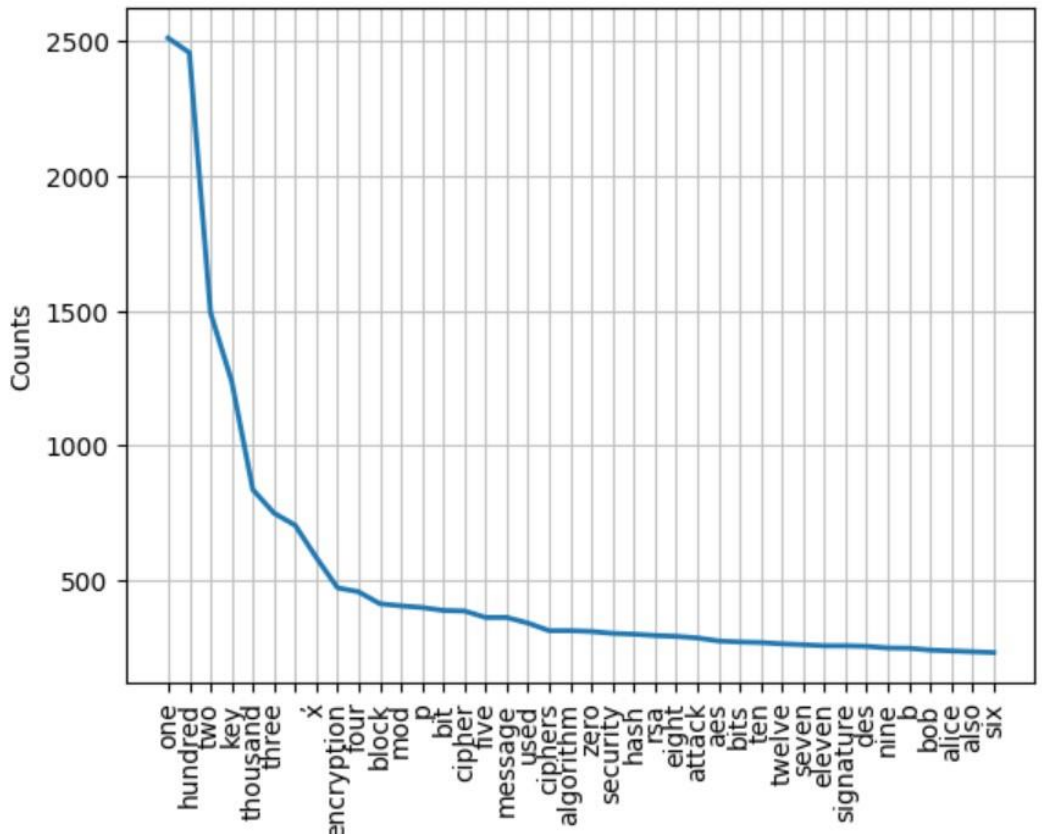


```
Out[11]: ['start',
          'project',
          'xi',
          'table',
          'contents',
          'one',
          'introduction',
          'cryptography',
          'data',
          'security',
          'eleven',
          'overview',
          'cryptology',
          'book',
          'twelve',
          'symmetric',
          'cryptography',
```

Tokenization after removing stop words

```
FreqDist({'one': 2512, 'hundred': 2458, 'two': 1493, 'key': 1239, 'thousand': 837, 'three': 750, ',': 705, 'x': 585, 'encryp
tion': 473, 'four': 458, ...})
```

Frequency after removing stop words



Frequency distribution of top 40 words(without stop words)

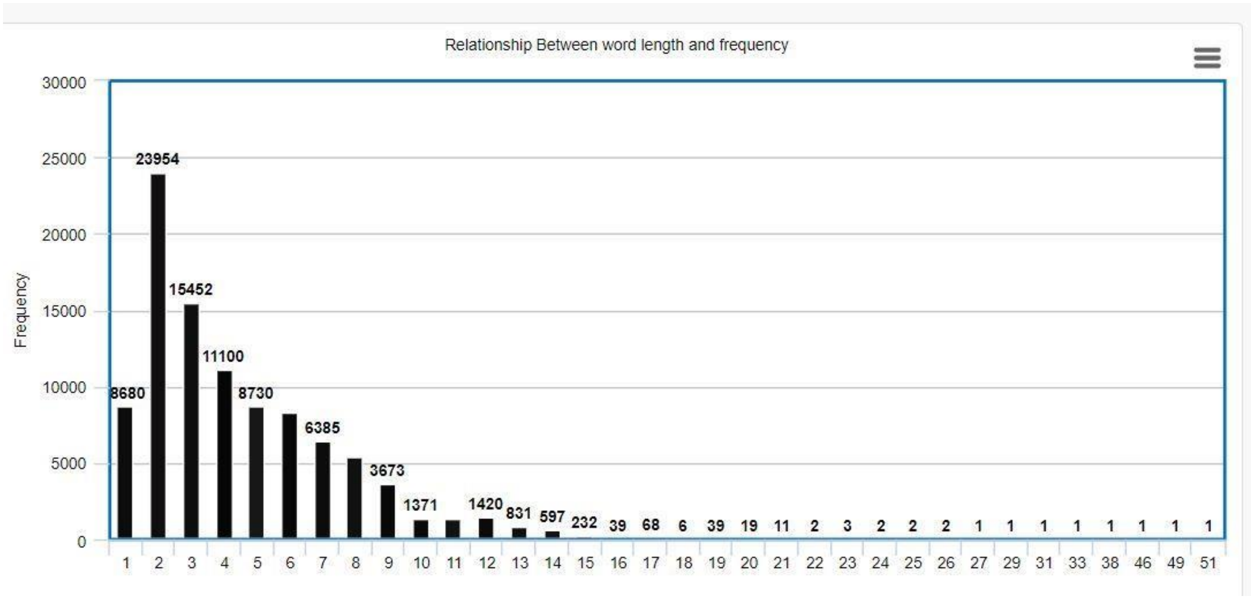
Inference :

- We observe that words like one, hundred, two and key are now amongst the words that appear bigger as their frequency is higher relatively after removal of stop words.
- It is giving the visual representation of the most frequent words in the book Understanding Cryptography after removal of stop words.

Relationship between the word length and frequency :

- We made an ordered dictionary to store the frequency of different word lengths in the text. The word lengths varied between 1 to 51. Next we plotted a bar chart showing the frequency of different word lengths. We did this for the book twice, once with the stop words and once after removal of stop words.

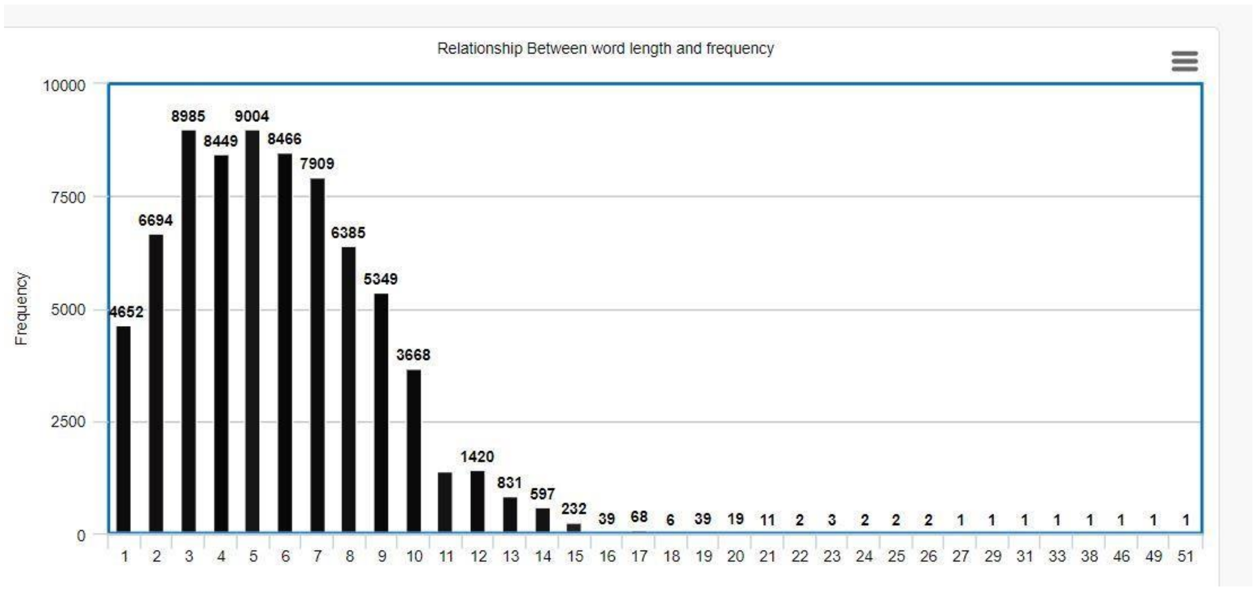
These plots are attached below :



Relationship between the word length and frequency (with stop words)

Inference :

- We have plotted a bar chart for the words of different length and their frequency. But in this chart we have included stop words, so the words of small length have large frequencies.
- Words of length 2 have the highest frequency.
- We can infer that words of lengths between 2 and 4 (inclusive) form the majority of the text.



Relationship between the word length and frequency (without stop words)

Inference :

- We have plotted a bar chart for the words of different length and their frequency. But for this chart we have not included stop words.
- Word length 5 has the highest frequency.
- We can infer that words of lengths between 3 and 6 (inclusive) form the majority of the text (after stop word removal).
- Comparing the above range with the chart for stop words included, we can also infer that stop words mainly have length between 2 and 3 (inclusive).

5. CONCLUSION

We learned a lot from working on this project, both in terms of subject comprehension and teamwork. All the procedures in text analytics were familiar to us on a superficial level, but this project has improved our knowledge of text processing with Python's NLP tools.

We got to the conclusion that this project shows the fundamental concepts of text preparation, PoS tagging, tokenization, etc. using python libraries and built-in toolkits. The graphs, bar charts, and word clouds produced by matplotlib improved our comprehension of the results and were useful for the visual representation of the data. Overall, this was a fantastic learning opportunity that has inspired us to learn more about NLP.

REFERENCES

- https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/
- https://www.mygreatlearning.com/blog/nltk-tutorial-with-python/#3
- http://librarycarpentry.org/lc-tdm/index.html
- https://www.researchgate.net/publication/319164243_Natural_Language_Processing_State_of_The Art_Current_Trends_and_Challenges