# NLP PROJECT

# REPORT

**Submitted by**

**Team** ***'SAAR'***


**Rupesh Maheshwari - 20UCS165**

**Aayush Ashokbhai Sheth - 20UCS004**

**Ajinkya Eknath Kadam - 20DCS001**

**Sankalp Jain - 20UCS172**


in partial fulfillment for the award of the degree of

**B.Tech.** in **CSE**


LNMIIT

The LNM Institute of Information Technology


**Github Repositary Link -:** https://github.com/jinks2882/NLP-project

## ACKNOWLEDGEMENT

We would like to express our special thanks to our project guide Dr. Sakthi Balan who gave us the golden opportunity to do this wonderful project on the topic **Text Analysis [NLP Project Round -1]** which also helped us in doing a lot of research and it was a great learning experience.

**DATE : 28 OCT 2022**



| | |
|---|---|
| **Rupesh Maheshwari** | **(20UCS165)** |
| **Aayush Ashokbhai Sheth** | **(20UCS004)** |
| **Ajinkya Eknath Kadam** | **(20DCS001)** |
| **Sankalp Jain** | **(20UCS172)** |

# TABLE OF CONTENTS

# ABSTRACT

Text Analytics is a very important aspect in the field of natural language processing and in this project, we worked on text preprocessing, PoS tagging, and various other operations the book ***Understanding Cryptography by Christopher Paar and Jan Pelzl***. Working on this project on this book was particularly interesting because the book is written in a very simple manner which give easy understanding of the book. For all the operations performed, we have used NLTK (natural language Tool Kit) to perform all the preprocessing, tokenization, and tagging. The project also described the frequency distribution and Word Cloud of the book and helped us understand some fields of text mining. 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 what it is today. Tools like Sentiment Analyzer, Parts of Speech (POS) Taggers, Chunking, Named Entity Recognition (NER), Emotion detection, 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.
- 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.
- The Sanskrit part of speech tagger specifically uses the treebank technique.
- 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.

## 2. INTRODUCTION

In this Project we imported a book in text format in order to perform text analysis using NLP techniques on it. We tokenized and lemmatized the imported text file, analyzed the frequency distribution and performed PoS tagging on the text file, further we are going to visualize the data which is going to be a good learning experience in the field of 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 tasks given in this project we have used NLTK which is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language. And we have imported some modules and functions in order to perform different activities such as preprocessing, tokenizing, removing stop words etc. And after performing such operations on the text file we have plotted frequency distribution for text file and created word clouds

## 3. Python Libraries and Modules Used

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

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

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

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

**Wordcloud package :** Provides modules to create wordcloud in python.

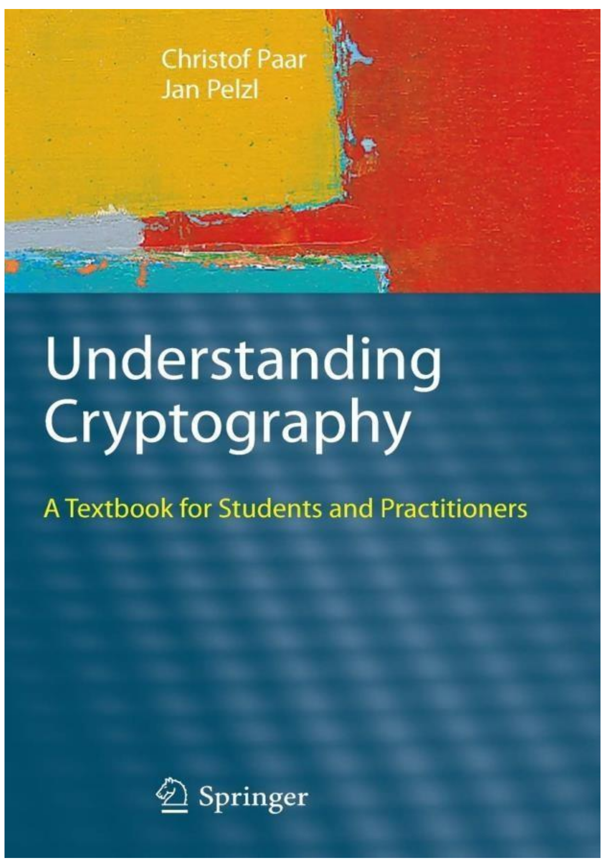
**Collection modules :** Provide different types of container data types**.**

# 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%20Paar%20.pdf)

The downloaded file is .txt file




Cover of the book 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 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 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. 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 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 analyze 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
zero 000s fifty-seven thirty-two overview of the des algorithm zero zero cee eee eee fifty-eight thirty-three internal st
ructure of des two hundred and twenty eee ee sixty-one three hundred and thirty-one initial and final permutation20 sixty
```

Text after Preprocessing

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

Tokenization before removing stop words

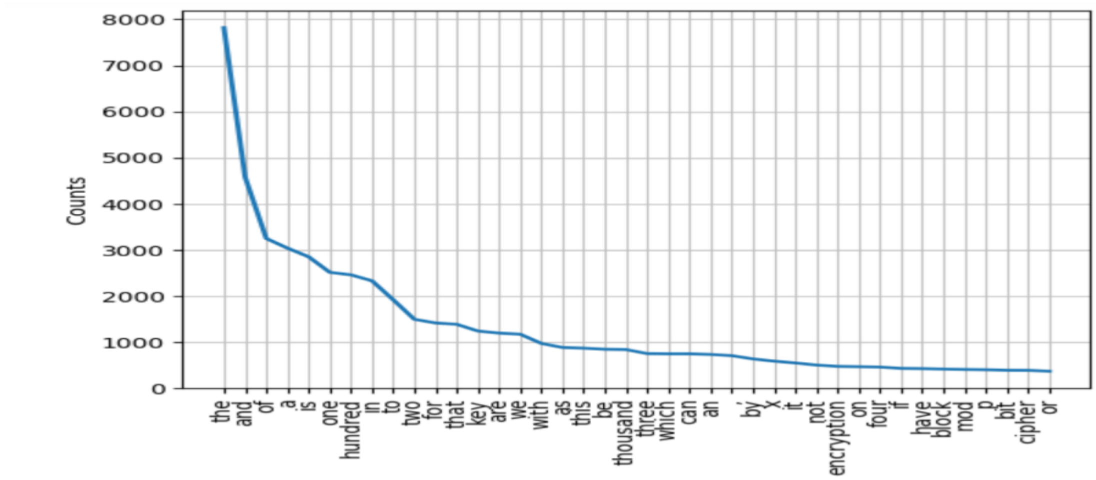
## Plotting Frequency distribution of tokens :

● 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.axis(), plt.show() 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.

● Infact, the overall word cloud is heavily dominated by stop words.

● Thus we can infer that the stop words occur in high frequency in the text.

## 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 :

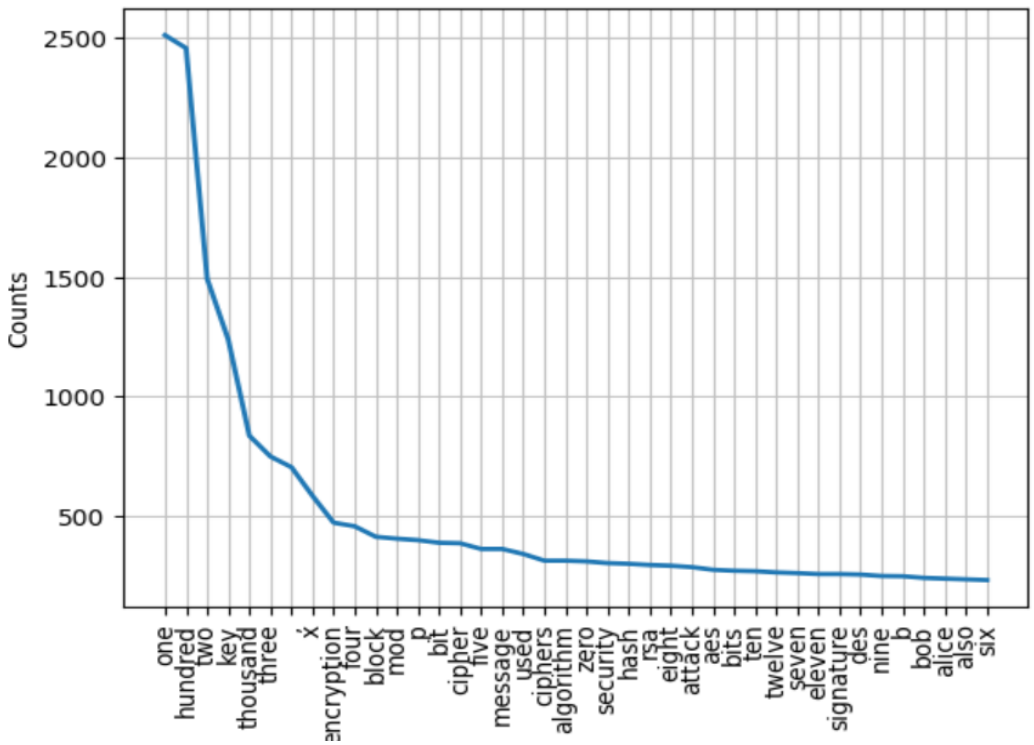

Word cloud (without stop words)

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

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)

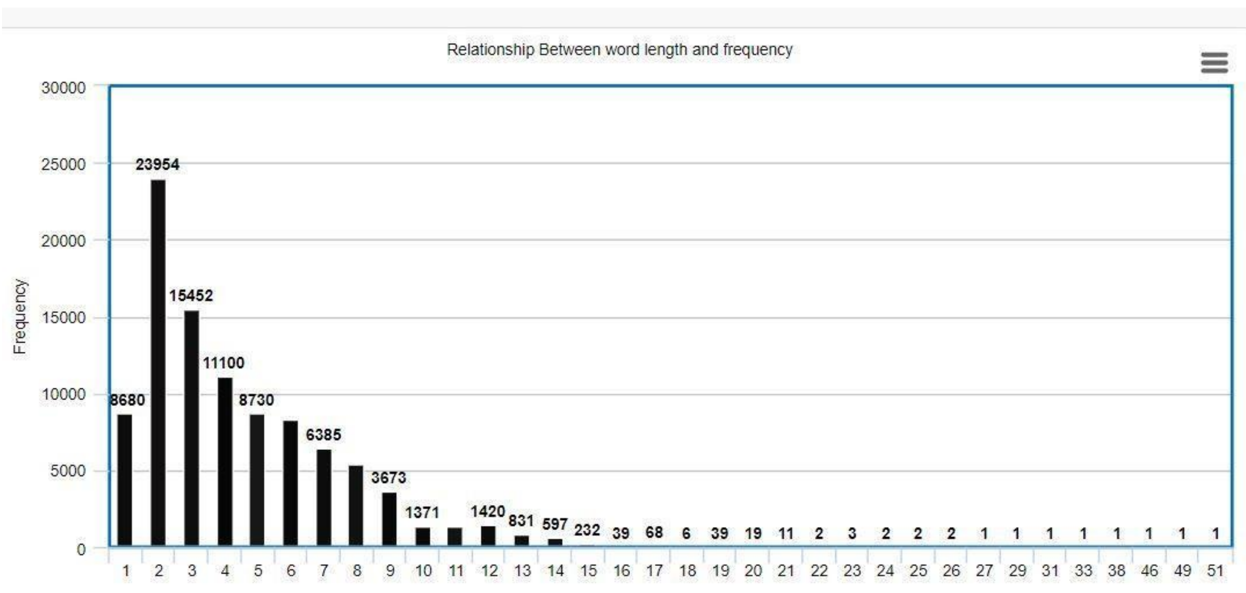
## Inference :

- It is giving the visual representation of the most frequent words in the book Understanding Cryptography after removal of stop words.

- 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.

## 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 :



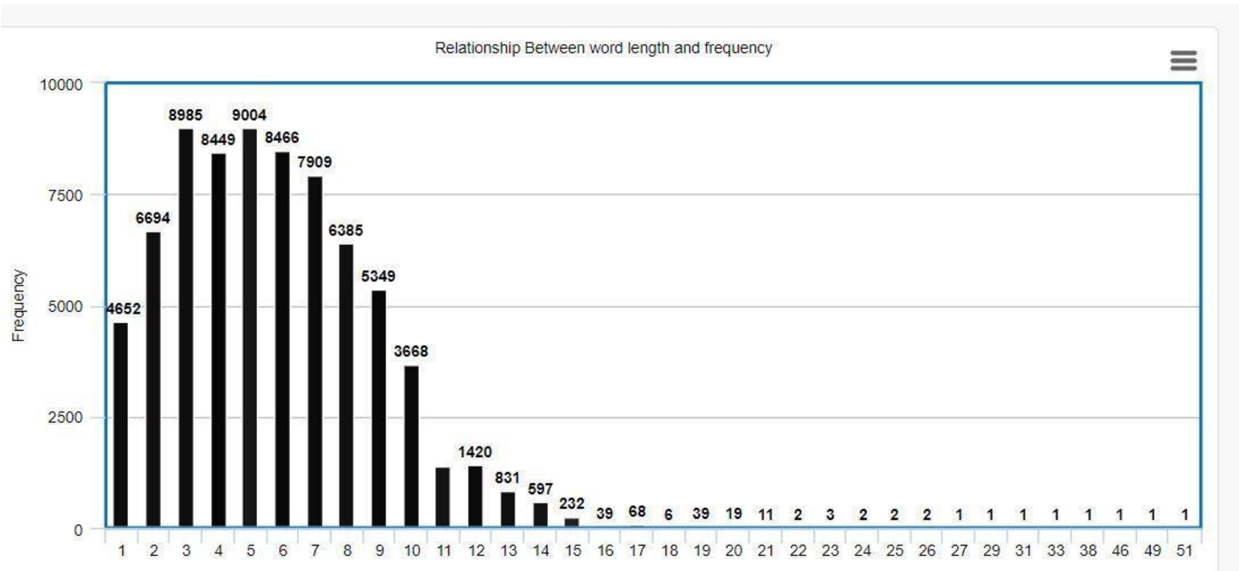

Relationship between the word length and frequency (with stop words)

## 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.

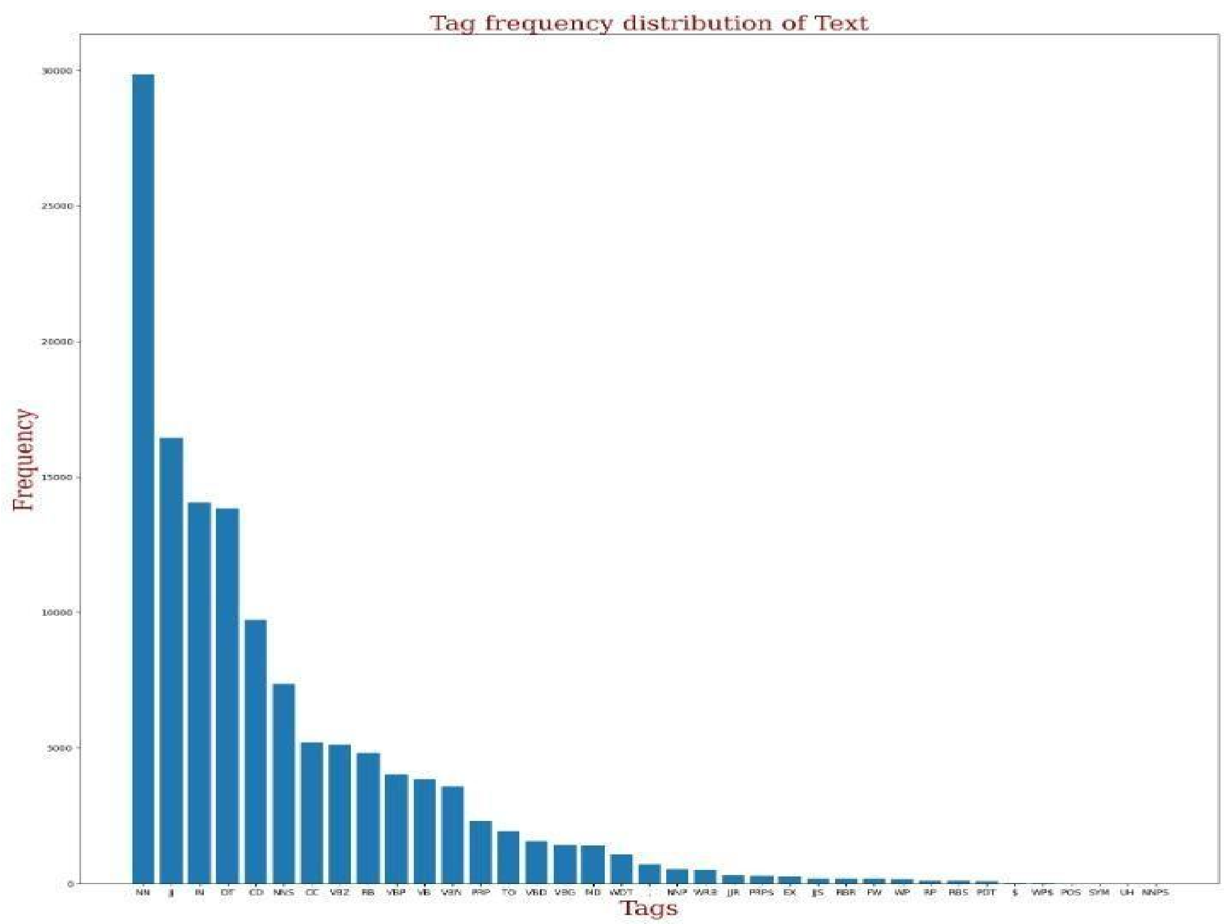
● Word length 5 has the highest frequency.

● 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).

## PoS Tagging and distribution of tags :

- For PoS tagging we used the original content of the eBooks without pre-processing it. We only removed the extra part to narrow down the string to the contents of the eBook. We used nltk.pos-tag() method for PoS tagging using the Penn Treebank Tagset. We word tokenized each sentence token before PoS tagging and then obtained the frequency distribution for the different tags for the texts. We plotted this distribution as a bar chart to show the frequency of occurrence of different PoS tags in the text.



```
OrderedDict([('NN', 29848), ('JJ', 16431), ('IN', 14048), ('DT', 13828), ('CD', 9721), ('NNS', 7358), ('CC', 5201), ('VBZ',
5119), ('RB', 4814), ('VBP', 4021), ('VB', 3846), ('VBN', 3578), ('PRP', 2301), ('TO', 1918), ('VBD', 1565), ('VBG', 1410),
('MD', 1391), ('WDT', 1070), (',', 705), ('NNP', 519), ('WRB', 506), ('JJR', 305), ('PRP$', 283), ('EX', 259), ('JJS', 185),
('RBR', 184), ('FW', 172), ('WP', 147), ('RP', 105), ('RBS', 100), ('PDT', 73), ('$', 17), ('WP$', 9), ('POS', 4), ('SYM',
3), ('UH', 2), ('NNPS', 1)])
```

Frequency distribution of tags

## Inference :

- We find that most frequent are
  - NN : Nouns(singular) (29848)
  - JJ : Adjective (16431)
  - IN : Preposition (14048)
  - And thus we get the idea about the type of content written in the text files

# PROJECT ROUND-2

## 1. INTRODUCTION

In this part of the Project we continue using the large book imported in text format previously. We also use an additional third book in the last part of this Round of the project. We will perform PoS tagging on the data in the book, extract entities from the book and check the performance of this operation.

We have chosen book related to our course Computer Security. This book is among the top downloaded books in the field of cryptography.

- Understanding Cryptography written by Christopher Paar and Jen Pelzl

For the tasks given in this project we have used *nltk* which is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language. And we have imported some modules and functions in order to perform different activities such as preprocessing, tokenizing, removing stop words, lemmatizing, information extraction etc. And after performing such operations on both of the text files we have plotted frequency distribution as required

## 2. Python Libraries and Modules Used Nltk.tokenizer package

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

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

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

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

**Wordcloud package :** Provides modules to create wordcloud in python.

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

**Nltk.stem.wordnet :** It is a module used for stemming and lemmatization.

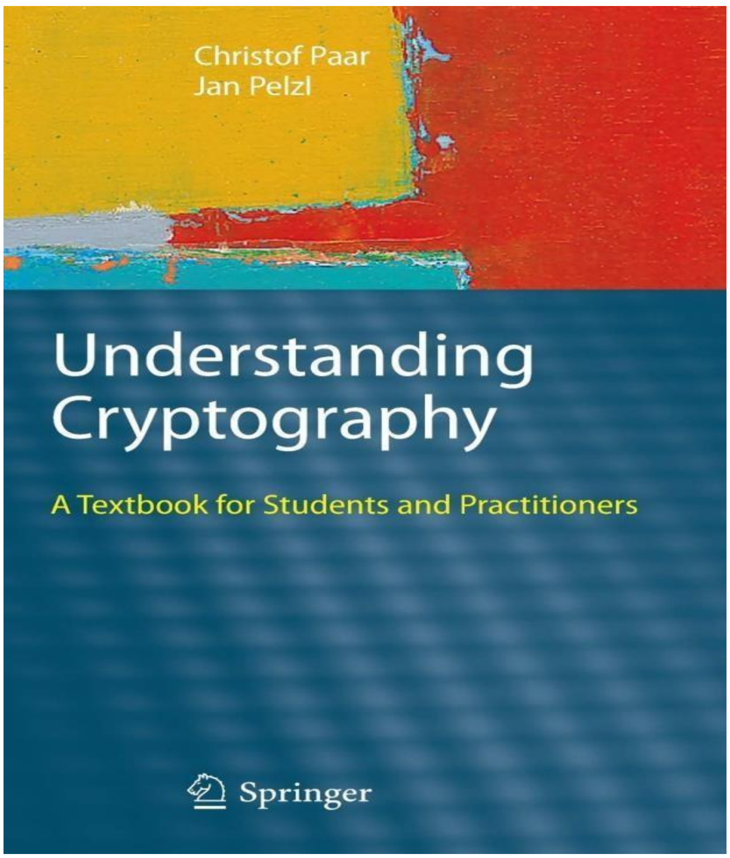
**Sklearn.feature_extraction.text** : This module can be used to extract features in a format supported by machine
learning algorithms from datasets consisting of formats such as text and image

## 5. 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%20Paar%20.pdf)

The downloaded file is .txt file




Cover of the book 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.

## Text Pre-Processing and Tokenization :

**Removing prefix and suffix to narrow down to text from Book** - Removing prefix and suffix to narrow down to text from eBook - Each book from the website had additional prefix and suffix in its .txt file, apart from the contents of the eBook. To narrow down our string to the relevant part only we considered only the substring of the original string which was marked with *** START OF THE PROJECT and *** END OF THE PROJECT in the .txt files.

**Expanding some Contractions** -We expanded some generic contractions. For example: can't to can not, all instances of 'll to will, 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.

**Expanding more Contractions according to general assumption** - For example: 've to have, 't to not, 'm to am, etc.

**Removing chapter number headings if any** - We remove the words named chapter because it increases the frequency of word chapter present in the text unnecessarily.

**Replacing one or more continuous whitespace characters by space** - Replacing one or more continuous white space characters with a single space to make the string evenly spaced.

## POS_Tagging

- We created a function to perform POS_Tagging of the text imported from the books.
- **What is POS_tagging -** It is the categorizing of words in the text with particular correspondence with the part of speech depending on the definition of the word and its context.
- For this we first sentence tokenized the entire text, then we word tokenized each sentence and finally we did POS Tagging for the words.
- We mainly used the *nltk.tokenize* library for this.
- **Function which extracts and returns lists of nouns and verbs in the Book whose pos_tag is given as parameter -** We created a function(*extract_nouns_and_verbs*) to find out the nouns and verbs present in the text with the help of their POS tags. We also find the categories that these words fall under in WordNet.
- To extract the nouns and verbs we first normalized the words in the tagged list above. In this process we change the words to lowercase and remove punctuations from it. Then we find the wordnet category for the word using its POS tag. Using this category we lemmatize the word using the *WordNetLemmatizer* and append the list in the proper list if it is a noun or a verb.

- We identify the word as a noun if its POS tag starts with an 'N', Similarly for a verb the tag starts with a 'V'. This is true for the 36 Tags present in the **Penn Treebank Tagset** which we have used for tagging in our code.

  The noun tags in the Tagset are -

| | |
|---|---|
| NN | Noun, singular or mass |
| NNS | Noun, plural |
| NNP | Proper noun, singular |
| NNPS | Proper noun, plural |

  The verb tags in the Tagset are -

| | |
|---|---|
| VB | Verb, base form |
| VBD | Verb, past tense |
| VBG | Verb, gerund or present participle |

| | |
|---|---|
| VBN | Verb, past participle |
| VBP | Verb, non-3rd person singular present |
| VBZ | Verb, 3rd person singular present |

Thus we can see that our observation of identifying categories by using the first letter of the Tag is correct for this tagset.

- **Frequency Distribution Plots and Interpretation -** Next we plot the frequency distribution of the nouns and the verbs in the text under different categories as per WordNet.

- The categories for nouns are as follows:

| | |
|---|---|
| noun.Tops | unique beginner for nouns |
| noun.act | Nouns denoting acts or action |
| noun.animal | Nouns denoting animal |
| noun.artifact | nouns denoting man-made objects |
| noun.attribute | nouns denoting attributes of people and objects |
| noun.body | Nouns denoting body parts |
| noun.cognition | nouns denoting cognitive processes and contents |
| noun.communication | nouns denoting communicative processes and contents |
| noun.event | nouns denoting natural events |
| noun.feeling | nouns denoting feelings and emotions |
| noun.food | nouns denoting foods and drinks |
| noun.group | nouns denoting groupings of people or objects |
| noun.location | nouns denoting spatial position |
| noun.motive | Nouns denoting goals |
| noun.object | nouns denoting natural objects (not man-made) |
| noun.person | Nouns denoting people |

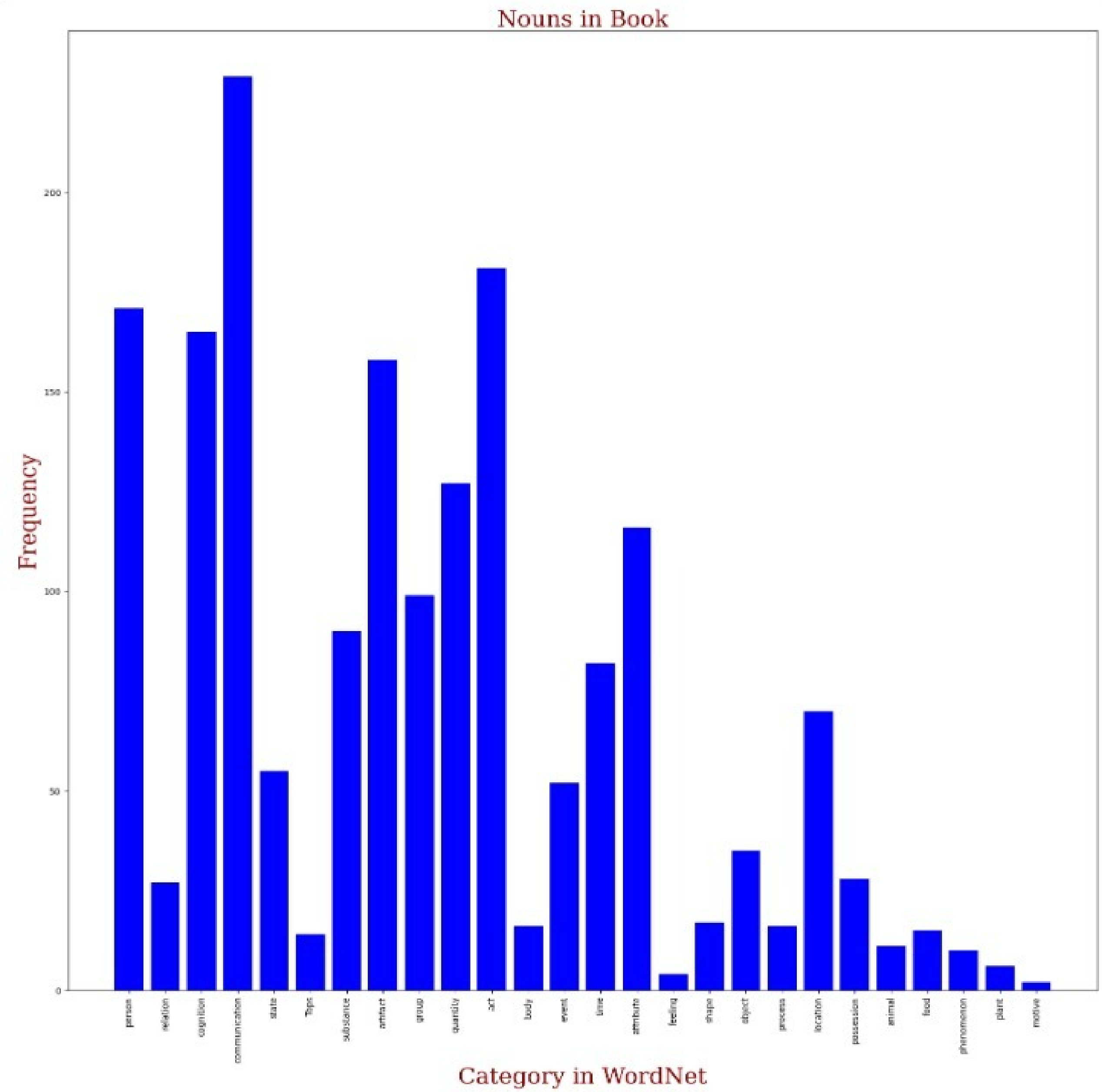
| | |
|---|---|
| noun.phenomenon | nouns denoting natural phenomena |
| noun.plant | Nouns denoting plants |
| noun.possession | nouns denoting possession and transfer of possession |
| noun.process | nouns denoting natural processes |
| noun.quantity | nouns denoting quantities and units of measure |
| noun.relation | nouns denoting relations between people or things or ideas |
| noun.shape | nouns denoting two and three dimensional shapes |
| noun.state | nouns denoting stable states of affairs |
| noun.substance | nouns denoting substances |
| noun.time | nouns denoting time and temporal relations |

- The categories for verbs are as follows:

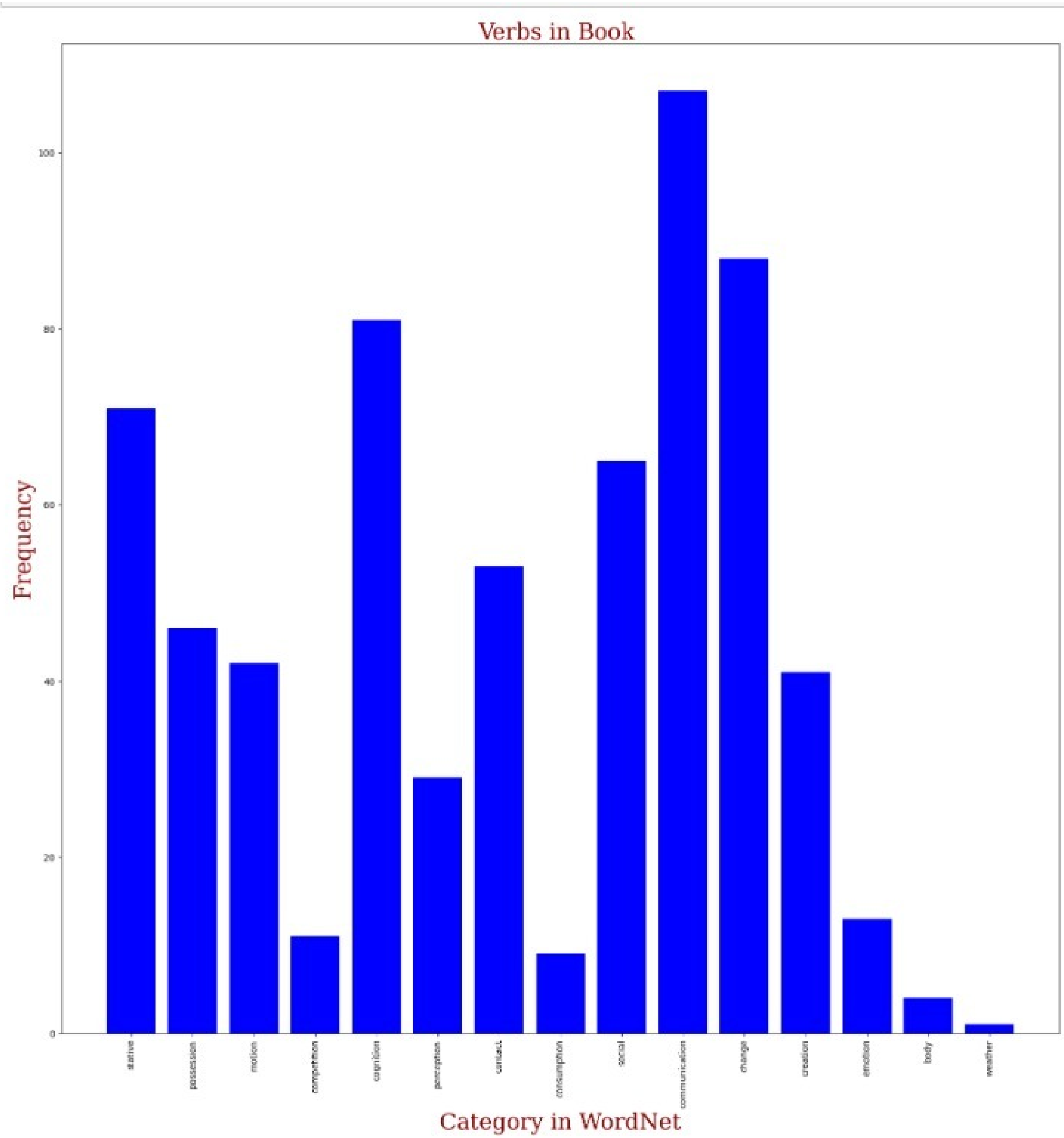
| | |
|---|---|
| verb.body | verbs of grooming, dressing and bodily care |
| verb.change | verbs of size, temperature change, intensifying, etc. |
| verb.cognition | verbs of thinking, judging, analyzing, doubting |
| verb.communication | verbs of telling, asking, ordering, singing |

| | |
|---|---|
| verb.competition | verbs of fighting, athletic activities |
| verb.consumption | verbs of eating and drinking |
| verb.contact | verbs of touching, hitting, tying, digging |
| verb.creation | verbs of sewing, baking, painting, performing |
| verb.emotion | verbs of feeling |
| verb.motion | verbs of walking, flying, swimming |
| verb.perception | verbs of seeing, hearing, feeling |
| verb.possession | verbs of buying, selling, owning |
| verb.social | verbs of political and social activities and events |
| verb.stative | verbs of being, having, spatial relations |
| verb.weather | verbs of raining, snowing, thawing, thundering |

- When we plot the graph of the frequency distribution of nouns in the Book, where we plot *Category in Wordnet* on the x-axis and we plot *frequency* for the category on the y-axis, we find that some of the most frequent noun categories are: act, attribute, communication, person



- When we plot the graph of the frequency distribution of the verbs in the book1 - *Pride and Prejudice*. Where we plot *Category in Wordnet* on the x-axis and we plot *frequency* for the category on the y-axis, we find that some of the most frequent verb categories are: communication, social.



## Extracting Entities:

- To identify all Persons, Location, and organisations in the novel we have used Named Entity Recognition (NER) in python. We used the nltk.ne_chunk function to do this.

- We pre-processed the text as above for this purpose, that is, we only reduced the text to the content of the book, expanded contractions, and removed chapter numbers from the text of the book.

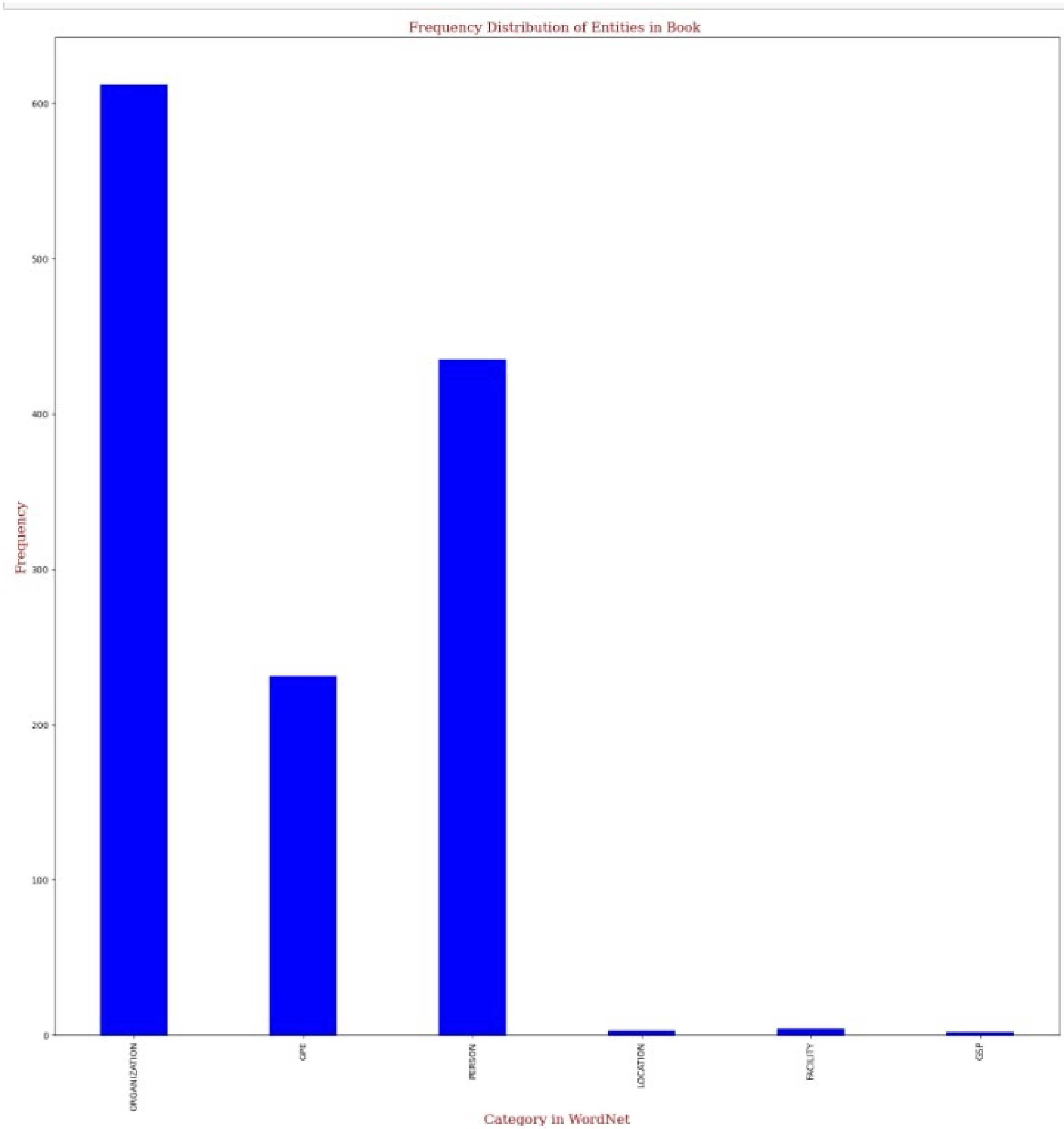
- Then we sentence tokenized the text using nltk.sent_tokenize(). We iterated over each sentence, word tokenizing it using nltk.word_tokenize() and found the part of speech tag for the words using nltk.pos_tag().

- Then we used the nltk.ne_chunk() function to chunk the list of tagged tokens. If the word was identified as an entity and was being encountered for the first time we inserted its label into the list entity_list.

- The bar graph plotted using the FreqDist() of the above list was then used to indicate the number of different types of entities in the text.

- The named entity types reported are : ORGANIZATION, PERSON, LOCATION, GPE(Geo-Political Entity), FACILITY and GSP(Geographical-Social-Political Entity).

- The bar graphs are attached:




When we plot the graph of the frequency distribution of the entities in the book ***Understanding Cryptography written by Cristopher Paar and Jen Palzl*** , where we plot *Entity type* on the x-axis and we plot frequency on the y-axis, we find that some of the most frequent entity is of type: organisation, least frequent is of type: GSP

## Performance Evaluation of NER:

- We selected 3 random passages from the book and manually identified the entities present in them.
- We then also extracted the entities from these passages using the nltk library as stated above.
- The passages can be found in the code.

PASSAGE 1

We are grateful for the excellent work of Daehyun Strobel ORG and Pascal WiSmann PERSON ,
who provided most of the artwork in the book and never complained about our many
changes. Axel Poschmann PERSON provided the section about the PRESENT GPE block cipher,
a very timely topic, and we are thankful for his excellent work. Help with technical
questions was provided by Frederick Armknecht PERSON (stream ciphers), Roberto Avanzi PERSON
(finite fields and elliptic curves), Alexander May PERSON (number theory), Alfred Menezes PERSON
and Neal Koblitz PERSON (history of elliptic curve cryptography), Matt Robshaw PERSON ( AES ORG ), and
Damian NORP Weber (discrete logarithms).

Many thanks go the members of the Embedded Security ORG group at the Univer- PERSON
sity of Bochum LOC — Andrey Bogdanov PERSON , Benedikt Driessen PERSON , Thomas Eisenbarth PERSON , Tim Giineysu ORG ,
Stefan Heyse PERSON , Markus Kasper PERSON , Timo Kasper PERSON , Amir Moradi PERSON and Daehyun Strobel ORG — who did
much of the technical proofreading and provided numerous sug-
gestions for improving the presentation of the material. Special thanks to Daehyun ORG
for helping with examples and some advanced I4TgX work, and to Markus PERSON for his
help with problems. Olga Paustjan's PERSON help with artwork and typesetting is also very
much appreciated.

An earlier generation of doctoral students from our group — Sandeep Kumar PERSON ,
Kerstin Lemke-Rust PERSON , Andy Rupp PERSON , Kai Schramm PERSON , and Marko Wolf PERSON — helped to cre-

---

An earlier generation of doctoral students from our group — Sandeep Kumar PERSON ,
Kerstin Lemke-Rust PERSON , Andy Rupp PERSON , Kai Schramm PERSON , and Marko Wolf PERSON — helped to cre-
ate an online course that covered similar material. Their work was very useful and
was a great inspiration when writing the book.

Result of Manual Labelling of Text 1
1.Daehyun Strobel | PERSON
2.Pascal WiSmann | PERSON
3.Axel Poschmann | PERSON
4.PRESENT | PRODUCT (BECAUSE HERE PRESENT SIGNIFIES TO A TYPE OF BLOCK CIPHER)
5.Frederick Armknecht | PERSON
6.Roberto Avanzi | PERSON
7.Alexander May | PERSON
8.Alfred Menezes | PERSON
9.Neal Koblitz | PERSON
10.Matt Robshaw | PERSON
11.AES | ORG
12.Damian | PERSON
13.Embedded Security | ORG
14.Univer- | ORG
15.Bochum | LOC
16.Andrey Bogdanov | PERSON
17.Benedikt Driessen | PERSON
18.Thomas Eisenbarth | PERSON
19.Tim Giineysu | PERSON
20.Stefan Heyse | PERSON
21.Markus Kasper | PERSON
22.Timo Kasper | PERSON
23.Amir Moradi | PERSON
24.Daehyun Strobel | PERSON
25.Daehyun | PERSON
26.Markus | PERSON
27.Olga Paustjan’s | PERSON
28.Sandeep Kumar | PERSON
29.Kerstin Lemke-Rust | PERSON
30.Andy Rupp | PERSON
31.Kai Schramm | PERSON
32.Wolf | PERSON

| | **Predicted Positive** | **Predicted Negative** |
|---|---|---|
| Actual Positive | 25 | 5 |
| Actual Neagative | 6 | 0 |

The values of Precision,Recall and F1 measure are as follows:

- Precision= 25/31= 0.806
- Recall = 25/30= 0.833
- F1 measure= 0.8196

```
In [10]:  #For TEXT1
          metrics = calc(text1_details)
          print("Text1 - \n")
          print("ACCURACY - ")
          print( metrics['acc'])
          print("F-score - ")
          if metrics['FS'] == -1 :
              print("Cannot be Calculated \n")
          else :
              print( metrics['FS'])
              print("\n")
```

```
Text1 -

ACCURACY -
0.6944444444444444
F-score -
0.819672131147541
```

In short, the verifier accepts a signature (r,s) only if the relation B’ -r° = a mod

p is satisfied. Otherwise, the verification fails. In order to make sense of the rather

arbitrary looking rules for computing the signature parameters r and s as well as the

verification, it is helpful to study the following proof.

Proof. We'll prove the correctness of the Elgamal NORP signature scheme. More specif-

ically, we show that the verification process yields a “true” statement if the verifier

uses the correct public key and the correct message, and if the signature parameters

(r,s) were chosen as specified. We start with the verification equation.

The Elgamal NORP signature scheme, which was published in 1985 DATE , is based on the diffi-

culty of computing discrete logarithms (cf. Chap. 8) CARDINAL . Unlike RSA ORG , where encryption

and digital signature are almost identical operations, the Elgamal digital signature

is quite different from the encryption scheme with the same name.

The attacker impersonates Bob PERSON , i.e., Oscar PERSON claims to Alice PERSON that he is in fact Bob PERSON .

Because Alice PERSON performs exactly the same computations as Oscar PERSON , she will verify

the signature as correct. However, by closely looking at Steps 1 and 2 LAW that Oscar PERSON

performs, one CARDINAL sees that the attack is somewhat odd. The attacker chooses the signa-

ture first ORDINAL and then computes the message. As a consequence, he cannot control the

semantics of the message x. PERSON For instance, Oscar PERSON cannot generate a message such as

“Transfer $ 1000 MONEY into Oscar ORG ’s account”. Nevertheless, the fact that

an automated verification process does not recognize the forgery is certainly not a

desirable feature. For this reason, schoolbook RSA ORG signature is rarely used in prac-

tice, and padding schemes are applied in order to prevent this and other attacks.

The signature consists of the pair (7,s). Both have roughly the same bit length

as p, so that the total length of the package (x, {r,s)) is about three CARDINAL times as long

as only the message x. Computing ORG r requires an exponentiation modulo p, which

can be achieved with the square-and-multiply algorithm. The main operation when

computing s is the inversion of kg. This can be done using the extended Euclidean NORP

algorithm. A speed-up is possible through precomputing. The signer can generate

the ephemeral key kg and r in advance and store both values. When a message is to

be signed, they can be retrieved and used to compute s. The verifier performs two CARDINAL

exponentiations that are again computed with the square-and-multiply algorithm,

and one CARDINAL multiplication.

```
Manual Labelling of Text2
1.Bob | PERSON
2.Oscar | PERSON
3.Alice | PERSON
4.Bob | PERSON
5.Alice | PERSON
6.Oscar | PERSON
7.Oscar | PERSON
8.Steps 1 and 2 | cardinal
9.one | CARDINAL
10.first | ORDINAL
11.x | WORK OF ART
12.Oscar | PERSON
13.1000 | MONEY
14.Oscar |  person
15.RSA | ORG
16.about three | CARDINAL
17.1985 | DATE
18.Euclidean | WORK OF ART
19.Two| CARDINAL
20.one| CARDINAL
21.Elgamal | PERSON
22.Elgamal | PERSON
23.8) | CARDINAL
24.RSA | ORG
```

| | **Predicted Positive** | **Predicted Negative** |
|---|---|---|
| Actual Positive | 19 | 6 |
| Actual Neagative | 3 | 0 |

The values of Precision,Recall and F1 measure are as follows:

- Precision= 19/22= 0.863
- Recall = 19/25= 0.76
- F1 measure= 0.8

We can see that the nltk entity recognition is not very accurate, however its performance is quite decent.

```
In [33]:  #For TEXT2
          metrics = calc(text2_details)
          print("Text1\2 - \n")
          print("ACCURACY - ")
          print( metrics['acc'])
          print("F-score - ")
          if metrics['FS'] == -1 :
              print("Cannot be Calculated \n")
          else :
              print( metrics['FS'])
              print("\n")
```

```
Text1 -

ACCURACY -
0.6785714285714286
F-score -
0.8085106382978724
```

Recognize relationship between entities
identify the domains of the entities. Assign each of them a serial number.

Extracting additional features and presenting the
table for the classes and relations
classified them into respective domains.

There are several features that can be useful for extracting the relationships between entities in a book:

Context: The context in which the entities are mentioned can provide clues about their relationship. For example, if two entities are mentioned in the same sentence or paragraph, it is likely that they are related in some way.

Co-occurrence: The frequency with which two entities appear together can also be a clue about their relationship. If two entities appear together frequently, it is likely that they are related in some way.

Verb usage: The verbs used to describe the actions of the entities can also provide clues about their relationship. For example, if one entity is described as "helping" another entity, it is likely that they have a positive relationship.

Named entity recognition (NER): This is the process of identifying and classifying named entities (such as people, organizations, locations, etc.) in text. This can be useful for identifying the entities that are involved in a relationship.

Part-of-speech tagging (POS): This is the process of identifying the part of speech (such as noun, verb, adjective, etc.) of each word in a sentence. This can be useful for understanding the role that a word plays in a sentence and how it is related to other words.

Dependency parsing: This is the process of analyzing the grammatical structure of a sentence and identifying the relationships between the words in the sentence (such as subject, object, modifier, etc.). This can be useful for understanding the relationships between entities in a sentence.

Semantic role labeling: This is the process of identifying the roles that entities play in a sentence (such as agent, patient, theme, etc.). This can be useful for understanding the relationships between entities and the actions they are performing.

Co-reference resolution: This is the process of identifying and linking mentions of the same entity in a text. This can be useful for understanding the relationships between entities that are mentioned multiple times in a text.

Overall, extracting the relationships between entities in a book may require a combination of these features, as well as additional natural language processing techniques.
For example, if the book is about a fictional crime investigation, you could identify the following entities:

1. The victim: The person who was harmed or killed in the crime
2. The suspects: The people who are suspected of committing the crime
3. The investigators: The people who are responsible for solving the crime
4. You could then define the relationships between these entities, such as:

The suspects are related to the victim through the crime they are suspected of committing
The investigators are related to the suspects through their role in the investigation
You could then extract additional features for these entities, such as:

1. The victim's age, gender, and occupation
2. The suspects' alibis, motives, and previous criminal records
3. The investigators' backgrounds, expertise, and methods for solving crimes

But in our case the book covers various cryptographic algorithms and protocols, you could identify the following entities:

Cryptographic algorithms: The mathematical techniques used to encrypt and decrypt data
Cryptographic protocols: The rules and procedures used to securely transmit data
Key sizes: The length of the keys used in cryptographic algorithms

(Table representation of augmented data by extracting additional features)

```
In [18]:
# Define the data for the table
data = [
    ["AES", "Symmetric key algorithm", "128, 192, or 256 bits", "Fast, widely used", "Block cipher", "Encrypts and decrypts d
    ["RSA", "Asymmetric key algorithm", "1024, 2048, or 4096 bits", "Secure, widely used", "Public key cipher", "Encrypts and
    ["Diffie-Hellman", "Key exchange algorithm", "1024, 2048, or 4096 bits", "Secure, widely used", "Public key cipher", "Ena
    ["MD5", "Hash function", "128 bits", "Fast, widely used", "Hash function", "Generates a fixed-size message digest from a
]

# Define the headers for the table
headers = ["Entity Name", "Type", "Key size", "Features", "Category", "Description"]

# Print the table
print("{:<15} {:<20} {:<15} {:<30} {:<15} {:<50}".format(*headers))
print("-" * 125)
for row in data:
    print("{:<15} {:<20} {:<15} {:<30} {:<15} {:<50}".format(*row))
```

```
Entity Name     Type                 Key size        Features                       Category        Description
----------------------------------------------------------------------------------------------------------------------------
-
AES             Symmetric key algorithm 128, 192, or 256 bits Fast, widely used              Block cipher    Encrypts and de
crypts data using the same key
RSA             Asymmetric key algorithm 1024, 2048, or 4096 bits Secure, widely used            Public key cipher Encrypts
and decrypts data using different keys
Diffie-Hellman  Key exchange algorithm 1024, 2048, or 4096 bits Secure, widely used            Public key cipher Enables sec
ure communication between two parties without exchanging a shared key in advance
MD5             Hash function        128 bits        Fast, widely used              Hash function   Generates a fixed-size m
essage digest from a message
```

# 5. CONCLUSION

Working on this project was a great learning experience for us in understanding the subject as well as team coordination. We all had surface-level knowledge about all the processes in text analytics but this project has helped us gain a better understanding of text processing using NLP techniques in python.

Using python libraries and in-built toolkits, we came to a conclusion that this project highlights the basic understanding of text preprocessing, PoS tagging, Tokenization, etc. The Graphs, Bar Charts, Word clouds represented by using matplotlib helped us more in understanding the output and it is also beneficial for the visual representation of the data. Overall this was a great learning experience and it has encouraged us to explore more in the fields of NLP.

# REFERENCES


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