LNMIIT
The LNM Institute of Information Technology

Natural Language Processing

CSE 3201

Project Report by

Team: INTELLIGENCIA

Utkarsh Khurana(20UCS215)
Yash Ghatiya(20UCS235)
Hrishit Jhaveri (20UCS081)
Yash Paliwal(20UCS238)

Course Instructor
Dr. Sakthi Balan Muthiah
Associate Professor, Dept. of CSE

Department of Computer Science Engineering
The LNM Institute of Information Technology, Jaipur

LNMIIT
The LNM Institute of Information Technology

Table of Contents

Project Round 1

Project Round 2

LNMIIT
The LNM Institute of Information Technology

Project Round 1

Problem Statement

This project uses the following NLP topics to solve the given problems:

- Clean the Data/Text-Preprocessing
- Tokenization of text
- Removal of Stop Words
- Creation of Word Cloud
- Giving appropriate tags to the word/PoS Tagging

We have used the Python programming language for all language processing in this project. For text processing, we'll use the Natural Language Toolkit, often known as the NLTK library, and data visualization tools like matplotlib.pyplot and WordCloud**.** Data visualization is also an essential aspect of this project, which we employ to understand the text and prepare for the text processing procedure

LNMIIT
The LNM Institute of Information Technology

Data Description

We have used the following book for this project:

T1: *Software Engineering 9th-Edition by Ian Sommerville*
Pages: *790*
Words: *303,119*
Characters:*1660297*

It is the reference book for the course CSE-0326(Software Engineering).

The book has been converted into txt file for the further processing and extraction of data.

Text Pre-Processing Steps

Text pre-processing is a critical component of Natural Language Processing activities. It has focused on removing the running section, chapter names, and removing pictures and tables from the book. It aids in transforming data into a more consumable form that machine learning algorithms can efficiently process.

For data pre-processing a combination of 5 processes were used to make the data more cleaner and convert the raw data to a meaningful data.

- **Removing the newline character:** Making the text single-line makes the text unseparated.

- **Removing Punctuations:** The raw data contains punctuation which results in the same words being used for analysis more than once. Since words like done and done! Are considered different

- **Making all the words lowercase**: The raw data contains words having different letter cases, which results in the same words being used for analysis more than once. Since words like done and Done are considered different. This step reduces the list of words drastically by combining the same words.

- **Removing numbers**: The numbers do not provide significant changes in the analysis and are not much of a use for the data being analyzed.

- **Removing extra white spaces**: During the above 4 steps, a lot of extra white spaces get introduced between words. Therefore there is a need of removing the extra white spaces between the words.

- **Lemmatization:** Lemmatization converts words in the root form and ensures that the word is in the language, unlike stemming which converts to root word but does not check if the word is in the language.

```
# Removing all the punctuations
punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
cleantext = ""
for char in text:
    if char not in punctuations:
        cleantext = cleantext + char

#Converting the text into lower case
cleantext = cleantext.lower()
```

```
[31] cleantext[:]  #after removing the punctuations
```

```
' part 1 introduction to software engineeringmy aim in this part of the book is to provide a general introduction tosoftware engineering i introduce important concepts such as softwareproce
sses and agile methods and describe essential software developmentactivities from initial software specification through to system evolutionthe chapters in this part have been designed to s
upport a onesemestercourse in software engineering  chapter 1 is a general introduction that introduces professional software  engineering and defines some software engineering concepts i
have  also written a brief discussion of ethical issues in software engineering  i think that it is important for software engineers to think about the  wider implications of their work
this chapter also introduces three case  studies that i use in the book namely a system for managing records of  patients undergoing treatment for mental health problems a control  syste
m for a portable insulin pump and a wilderness weather system  chapters 2 and 3 cover software engineering processes and agile devel  opment in chapter 2 i introduce commonly used generic
software  process models such as the waterfall model and i discuss the basic  activities that are part of these processes chapter 3 supplements this  with a discussion of agile developme
nt methods for software engineer  ing i mostly use extreme programming as an example of an agile method  but also briefly introduce scrum in this chapter19the remainder of the  chapters i
n this  part  are extended descriptions  ofthe software  process activities  that  will  be introduced  in  chapter  2chapter 4 covers the critically important  topic of requirements  e
ngineering where the  requirements  for  what a  system should do  are definedchapter 5 introduces system modeling  using the uml where i  focus onthe use of  use case diagrams class diagra
ms sequence diagrams  andstate diagrams  for  modeling  a software  system  chapter 6 introducesarchitectural design and i discuss the importance of  architecture and theuse of architectura
l patterns in  software designchapter 7 introduces objectoriented  design  and the use  of design patterns i also introduce important implementation  issues here—reuse configuration  manage
ment and hosttarget  development and discuss opensource development chapter 8 focuses on software testing from unit testing during system development to the testing  of software releases i
alsodiscuss the use of  testdriven  development—an approach pioneered  inagile methods but  which  has wide applicability  finally chapter 9 presents  an  overview  of  software  evoluti
on  issues  i  cover  evolutionprocesses software maintenance and legacy system management20  1introduction  objectives  the objectives of this chapter are to introduc
e software engineering and  to provide a framework for understanding the rest of the book when you  have read this chapter you will  \x02 understand what softw
are engineering is and why it is important  \x02 understand that the development of different types of software  systems may require different software engineering t
echniques  \x02 understand some ethical and professional issues that are important  for software engineers  \x02 have been introduced to three systems of d
ifferent types that will be  used as examples throughout the book  contents  11 professional software development  12 software engineering ethics
13 case studies214 chapter  1 \x02 introduction  we cant run the modern world  without  software national  infrastructures and utili
ties are controlled by computerbased systems and most electrical products include a  computer  and controlling  software  industrial  manufacturing  and
```

Output for book

Tokenization of text

Next, we tokenize the cleaned text using the function **'word_tokenize'** from the **nltk.tokenize library.**

- Tokenizers divide strings into lists of substrings.
- This particular tokenizer 'word_tokenize' requires the **Punkt sentence tokenization model** to be installed.
- This Punkt sentence tokenizer divides text into a list of sentences by using an unsupervised algorithm to build a model for words.

```
[37] tokens = word_tokenize(cleantext)
     tokens[:]

     ['part',
      '1',
      'introduction',
      'to',
      'software',
      'engineeringmy',
      'aim',
      'in',
      'this',
      'part',
      'of',
      'the',
      'book',
      'is',
      'to',
      'provide',
      'a',
      'general',
      'introduction',
      'tosoftware',
      'engineering',
      'i',
```

Tokens Output of book

LNMIIT
The LNM Institute of Information Technology

Removal of Stop Words

The stopwords are then removed from our corpus. It would be hard to list all of the stopwords in the English language. As a result, NLTK gives a list of various stopwords used in various languages.

i) We use **nltk.download('stopwords')** to download that list

ii) Then, we match every word/token in the 'tokens' variable and if the word is a stopword, we tend to ignore it.

```
finaltext[:]

'part 1 introduction software engineeringmy aim part book provide general introduction tosoftware engineering introduce important concepts softwareprocesses agile methods
describe essential software developmentactivities initial software specification system evolutionthe chapters part designed support onesemestercourse software engineering ch
apter 1 general introduction introduces professional software engineering defines software engineering concepts also written brief discussion ethical issues software engi
neering think important software engineers think wider implications work chapter also introduces three case studies use book namely system managing records patients un
dergoing treatment mental health problems control system portable insulin pump wilderness weather system chapters 2 3 cover software engineering processes agile devel
opment chapter 2 introduce commonly used generic software process models waterfall model discuss basic activities part processes chapter 3 supplements discussion agile
development methods software engineer ing mostly use extreme programming example agile method also briefly introduce scrum chapter19the remainder chapters part extended
descriptions ofthe software process activities introduced chapter 2chapter 4 covers critically important topic requirements engineering requirements system definedchapter
5 introduces system modeling using uml focus onthe use use case diagrams class diagrams sequence diagrams andstate diagrams modeling software system chapter 6 introdu
cesarchitectural design discuss importance architecture theuse architectural patterns software designchapter 7 introduces objectoriented design use design patterns also in
troduce important implementation issues here-reuse configuration management hosttarget development discuss opensource development chapter 8 focuses software testing unit t
esting system development testing software releases alsodiscuss use testdriven development-an approach pioneered inagile methods wide applicability finally chapter 9 pres
ents overview software evolution issues cover evolutionprocesses software maintenance legacy system management20 1introduction objectives objectives chapter introduce softw
are engineering provide framework understanding rest book read chapter \x02 understand software engineering important \x02 understand development different types software
systems may require different software engineering techniques \x02 understand ethical professional issues important software engineers \x02 introduced three systems diffe
rent types used examples throughout book contents 11 professional software development 12 software engineering ethics 13 case studies214 chapter 1 \x02 introduction ca
nt run modern world without software national infrastructures utili ties controlled computerbased systems electrical products include computer controlling software indust
rial manufacturing distribution completely computerized financial system entertainment including music industry computer games film television software intensive therefore
software engineering essential functioning national inter national societies software systems abstract intangible constrained properties materials governed physical laws m
anufacturing processes simplifies software engineering natural limits potential software however lack physical constraints software systems quickly become extremely comple
x difficult understand expensive change many different types software systems simple embedded sys tems complex worldwide information systems pointless look universal no
tations methods techniques software engineering different types software require different approaches developing organizational information system completely different deve
loping controller scientific instru ment neither systems much common graphicsintensive com puter game applications need software engineering need software engineering te
```

Output after removing Stopwords

PoS Tagging

After fully cleaning our data, we go on to the text processing stage. In this case, we must give suitable Part-of-Speech Tags to the terms.

For this, we use the function "**pos_tag( )**" from the NLTK library of python.

- The pos_tag() function uses the **Penn Treebank Tag Set**, which has 36 tags to assign from to the words.
- The pos_tag returns **a tuple consisting of the token and the tag**.

```
tagged = nltk.pos_tag(tokens)
tagged[:] # pos tagging
```

```
[('part', 'NN'),
 ('1', 'CD'),
 ('introduction', 'NN'),
 ('to', 'TO'),
 ('software', 'NN'),
 ('engineeringmy', 'JJ'),
 ('aim', 'NN'),
 ('in', 'IN'),
 ('this', 'DT'),
 ('part', 'NN'),
 ('of', 'IN'),
 ('the', 'DT'),
 ('book', 'NN'),
 ('is', 'VBZ'),
 ('to', 'TO'),
 ('provide', 'VB'),
 ('a', 'DT'),
 ('general', 'JJ'),
 ('introduction', 'NN'),
 ('tosoftware', 'NN'),
 ('engineering', 'NN'),
 ('i', 'NN'),
 ('introduce', 'VBP'),
 ('important', 'JJ'),
 ('concepts', 'NNS'),
 ('such', 'JJ'),
```

Part-of-Speech Tag

Data Visualization

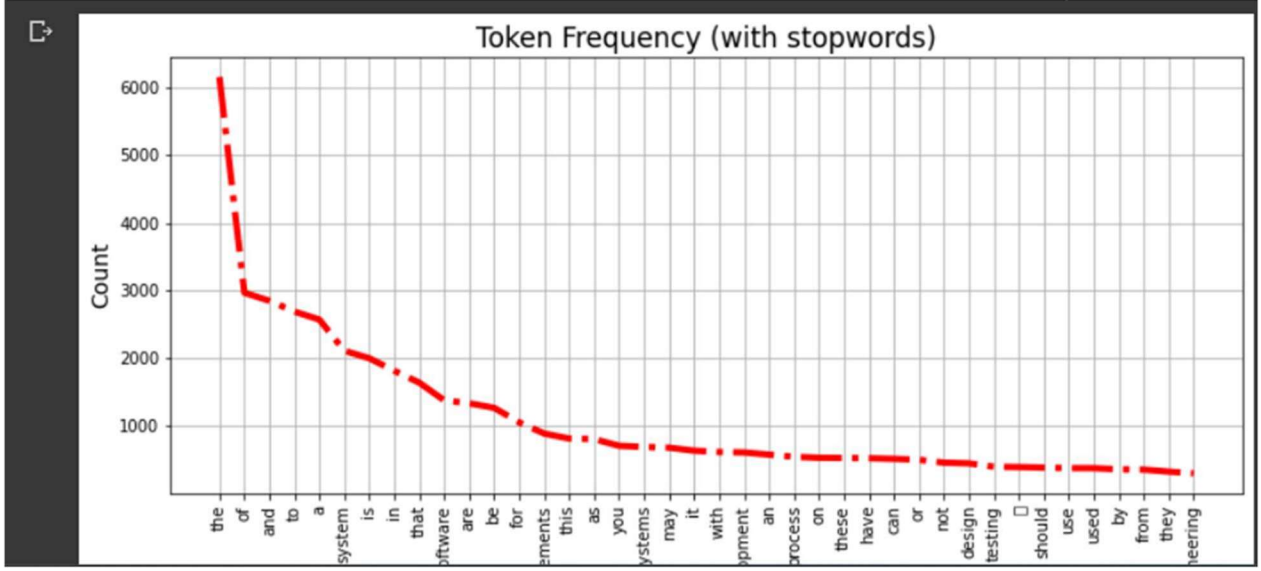
We utilized several graphs and plots and other visualizations throughout the assignment to gain a clear understanding of the material we had to work on. Data visualization is particularly vital throughout the pre-processing phases and text processing to accurately and readily comprehend the trends of our text, since humans prefer to understand and recall pictures rather than big blocks of text or tables.

Visualization during Pre-processing:

a) Keeping Stopwords in our Data:

- Stopwords are words that are not often used in natural language, such as articles (the, an, a).
- To compute the individual frequency of the tokens, we utilise the nltk.FreqDist() method.

```
tokens = word_tokenize(cleantext)
freq = nltk.FreqDist(tokens)
freq = {k: v for k, v in sorted(freq.items(), key=lambda item: item[1],reverse=True)}
x = list(freq.keys())[:40]
y = list(freq.values())[:40]
plt.figure(figsize=(12,5))
plt.plot(x,y,c='r',lw=4,ls='-.')
plt.grid()
plt.xticks(rotation=90)
plt.title('Token Frequency (with stopwords)',size=17)
plt.xlabel('Words',size=14)
plt.ylabel('Count',size=14)
plt.show()
```



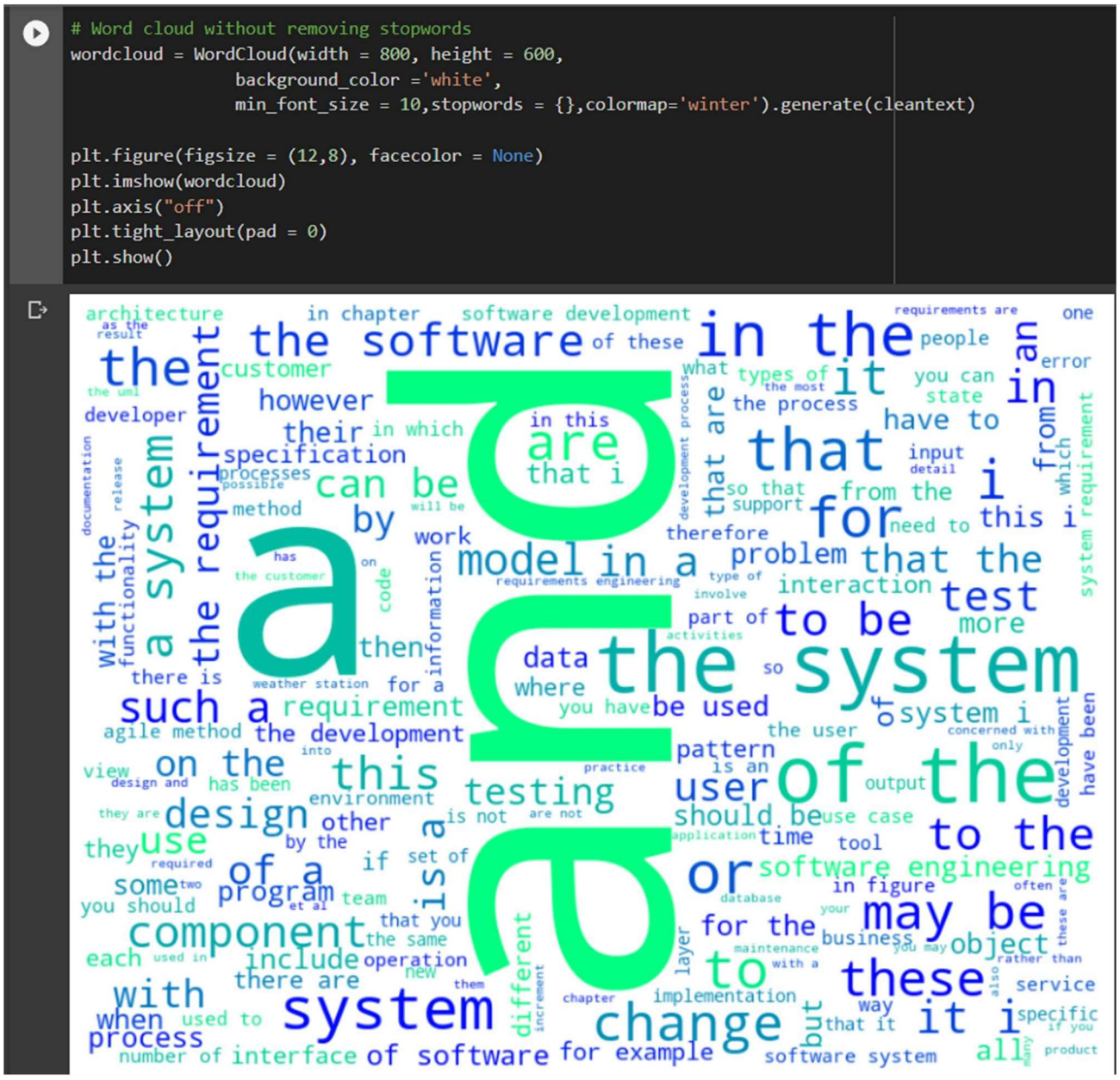
Frequency Distribution of Tokens (with Stopwords)

For visualization, we also use the Word Cloud library.

- The magnitude of each word in a word cloud is related to its frequency or relevance in the text.
- Significant textual points might be emphasized as a result of this..

Output for Word Cloud (with Stopwords):

```
# Word cloud without removing stopwords
wordcloud = WordCloud(width = 800, height = 600,
                background_color ='white',
                min_font_size = 10,stopwords = {},colormap='winter').generate(cleantext)

plt.figure(figsize = (12,8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Inference:

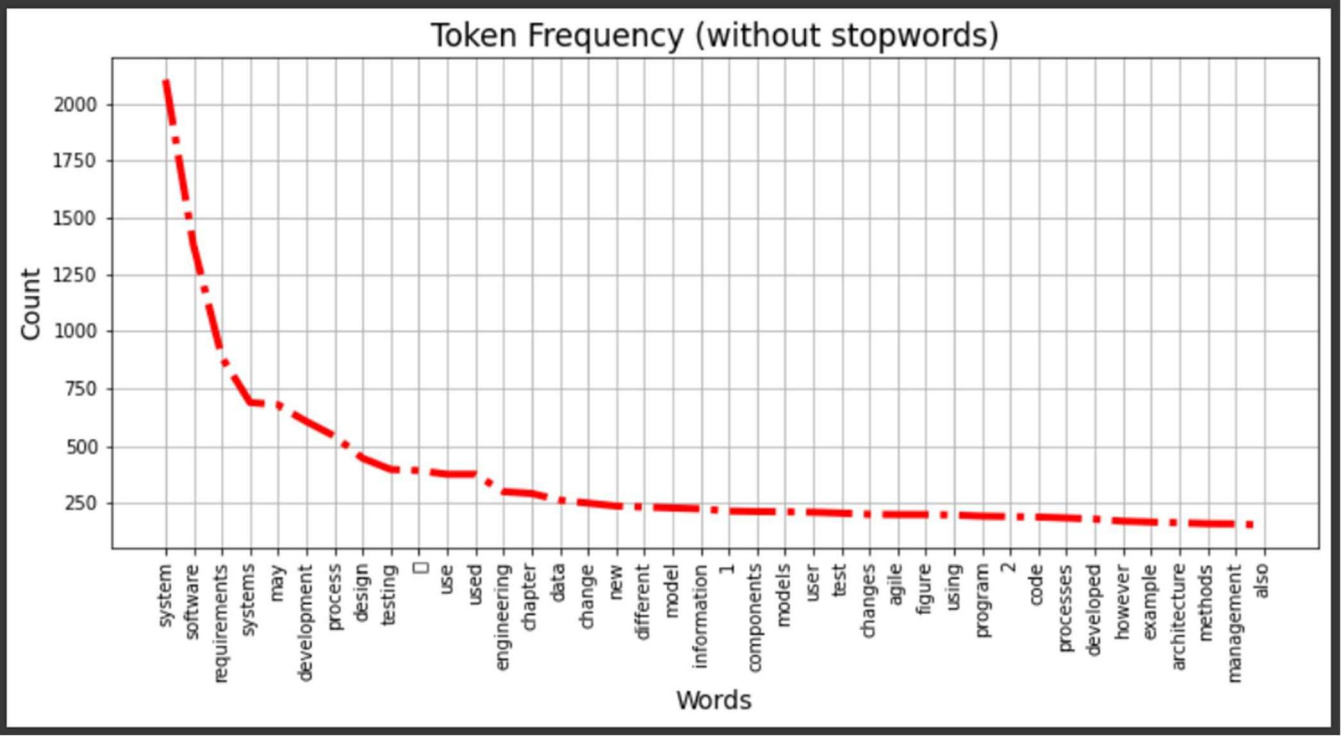
From these word clouds, we can infer that most words in this text are stop words such as “the, and, of, to” etc.

- These stopwords, as clearly seen, occupy an unnecessary high frequency count, which is meaningless in the processing of the text.
- We thus need to remove them from the corpus created.

b) Removing the Stopwords from our Data

We constructed frequency graphs of the remaining words after removing the stopwords by matching each token from the NLTK library's'stopwords' list..

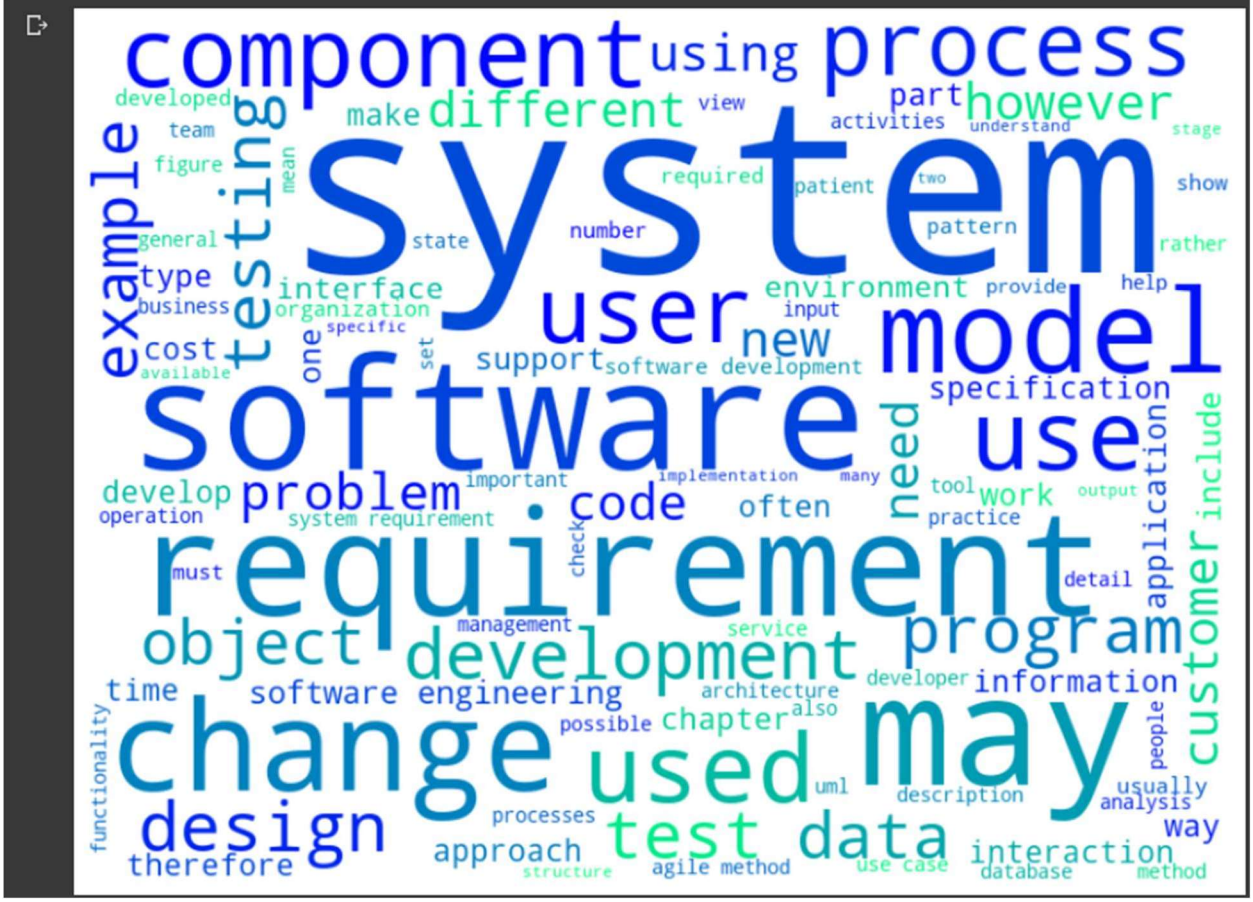
```
tokens = word_tokenize(finaltext)
freq = nltk.FreqDist(tokens)
freq = {k: v for k, v in sorted(freq.items(), key=lambda item: item[1],reverse=True)}
x = list(freq.keys())[:40]
y = list(freq.values())[:40]
plt.figure(figsize=(12,5))
plt.plot(x,y,c='r',lw=4,ls='-.')
plt.grid()
plt.xticks(rotation=90)
plt.title('Token Frequency (without stopwords)',size=17)
plt.xlabel('Words',size=14)
plt.ylabel('Count',size=14)
plt.show()
```



Frequency Distribution of Tokens

After cleaning the data of stopwords, we again create the word cloud.

```
# Word cloud after removing stopwords
wordcloud = WordCloud(width = 800, height = 600,
                background_color ='white',
                min_font_size = 10,stopwords = {},colormap='winter').generate(finaltext)

plt.figure(figsize = (12,8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



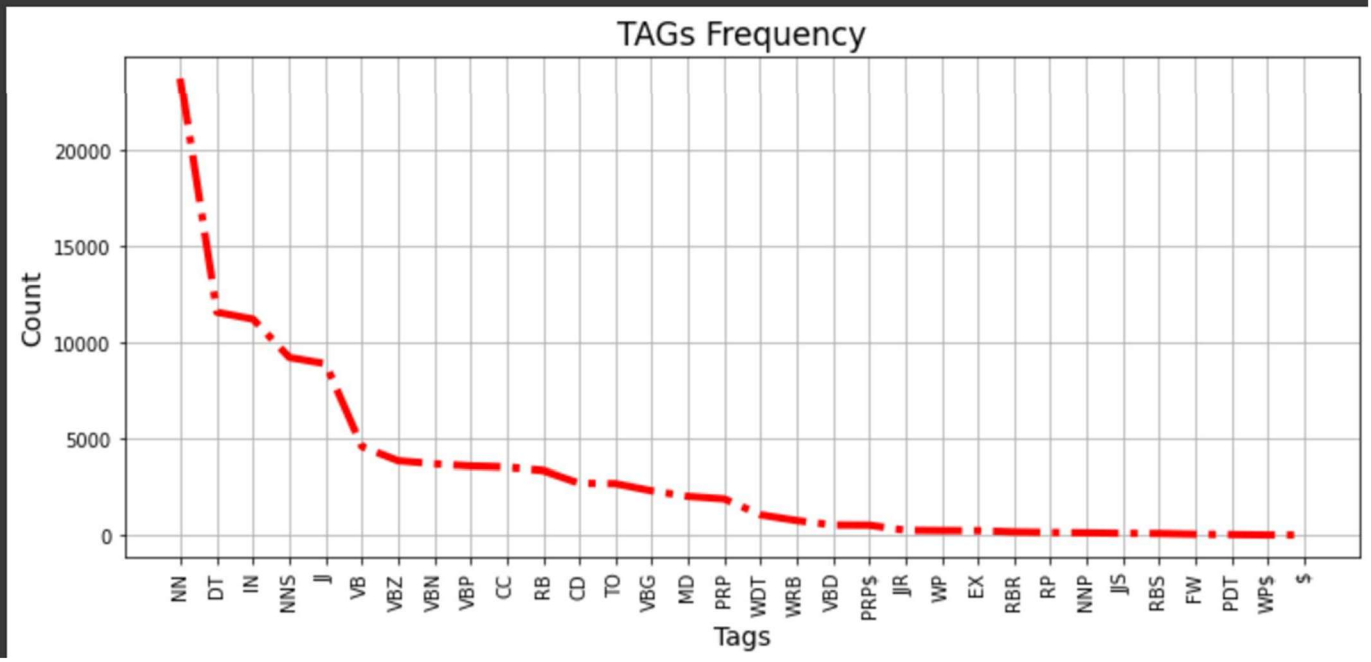
Output for Word Cloud (without Stopwords)

Distribution of Part-of-Speech Tags in our Data:

We attempted to assess the frequency of different tags after attaching PoS Tags to different terms.

- To do this, we first utilised the "Counter" library to call the function of the same name, which computes the frequency of each tag used in the final text.
- We then created a graph with the same data.

```
freq_tags = nltk.FreqDist(counts)
freq_tags = {k: v for k, v in sorted(freq_tags.items(), key=lambda item: item[1],reverse=True)}
x = list(freq_tags.keys())[:40]
y = list(freq_tags.values())[:40]
plt.figure(figsize=(12,5))
plt.plot(x,y,c='r',lw=4,ls='-.')
plt.grid()
plt.xticks(rotation=90)
plt.title('TAGs Frequency',size=17)
plt.xlabel('Tags',size=14)
plt.ylabel('Count',size=14)
plt.show()
```



Distribution of Tags

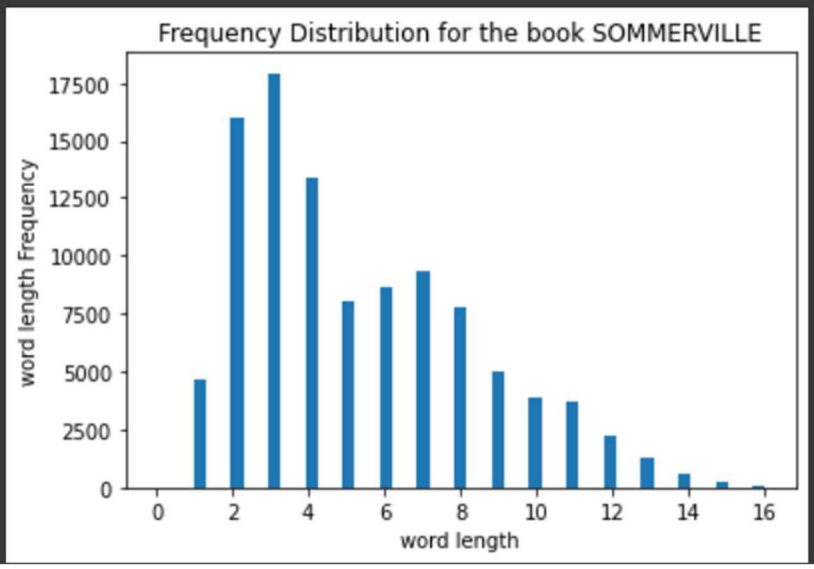
Relationship between the word length and frequency:

Here, we tend to analyse a relationship between the word length and how many words with such word length occurs.

- We first associate a bin for the bar graph using "**numpy**" library
- Then using **len( )** function we calculate the length of each token
- Then we plot a graph for frequency of such word lengths using **matplotlib.pyplot**

```
[49] import numpy as np
     bin_size=np.linspace(0,16)
```

```
#Finding Wordlength and storing it as a list
wordLength = [len(r) for r in tokens]

#Plotting histogram of Word length vs Frequency
plt.hist(wordLength, bins=bin_size)
plt.xlabel('word length')
plt.ylabel('word length Frequency')
plt.title('Frequency Distribution for the book SOMMERVILLE')
plt.show()
```



Output for Frequency Distribution

LNMIIT
The LNM Institute of Information Technology

PROJECT ROUND 2

First Part:

1. Find the nouns and verbs in the book. Get the categories that these words fall under in the WordNet. Note that there are 25 categories and 16 categories for Nouns and Verbs respectively.

2. Get the frequency of each category for each noun and verb in their corresponding hierarchies and plot a histogram for the same.

Finding Nouns and verbs in the Book and categorising using the WordNet:

Step1: We have used the NLTK library for POS tagging the words.
Here, tags_T1 = the POS tags results for Book Sommerville,

Step2: In subsequent steps, we identified the noun & verb categories and then stored them separately in the list for the WordNet Classification. We conclude from observing the POS tags that all the tags that start with 'N' are Nouns and those that begin with 'V' are Verbs.

Finding Nouns and verbs in the Book and categorising using the WordNet.

```
[26] nouns_T1=[]
     for i in tagged:
       if(i[1][0] == 'N'):
         nouns_T1.append(i[0])
```

```
[27] nouns_T1[:]
```

```
'service',
'engineering',
'software',
'development',
'services',
'software',
'systems',
'patterns',
'timing',
'analysis',
'realtime',
'systems',
'software',
'engineering',
'separation',
'concerns',
'aspects',
'points',
'software',
'engineering',
'contents',
'part',
'software',
'management'
```

We have used a list nouns_T1 to store the nouns in Book1. From the image it is evident that the stored words are nouns. For example service, software ,

development are all forms of Nouns. Similarly, verb_T1 is used to store the extracted results of the verbs in the given book.

```
+ Code  + Text

[29] verb_T1=[]
     for i in tagged:
       if(i[1][0] == 'V'):
         verb_T1.append(i[0])

verb_T1[:]
 'publicize',
 'unsafe',
 'comes',
 'validated',
 'according',
 'crite',
 'strict',
 'operate',
 'validated',
 'fail',
 'result',
 'damage',
 'failure',
 'disclose',
 'result',
 'make',
 'appropriate',
 'depends',
 'involved',
 'affected',
 'justified',
 'publicize',
 'using',
 'say',
 'try',
 'resolve',
 'respecting',
```

We then calculated the total number of nouns and verbs that are present in the given book, and the output is present below:

```
...]

print("There are Total "+str(len(nouns_T1))+ " Nouns and " +str(len(verb_T1))+" Verbs present in the book sommerville " )

There are Total 93936 Nouns and 34161 Verbs present in the book sommerville
```

Step 3: We downloaded the wordnet from the NLTK library and then imported it from the NLTK corpus.

```
[31] nltk.download('wordnet')
     nltk.download('omw-1.4')

     [nltk_data] Downloading package wordnet to /root/nltk_data...
     [nltk_data] Downloading package omw-1.4 to /root/nltk_data...
     True

[32] from nltk.corpus import wordnet
```

Step 4 : Using the WordNet the nouns and verbs got divided into the respective categories like in book 1, ‘software’ categories to noun.communication and ‘engineering’ categories to noun.act as shown in code snippet below. Here, nt1 represents the list that contains the final categorisation of the nouns in the given Book.

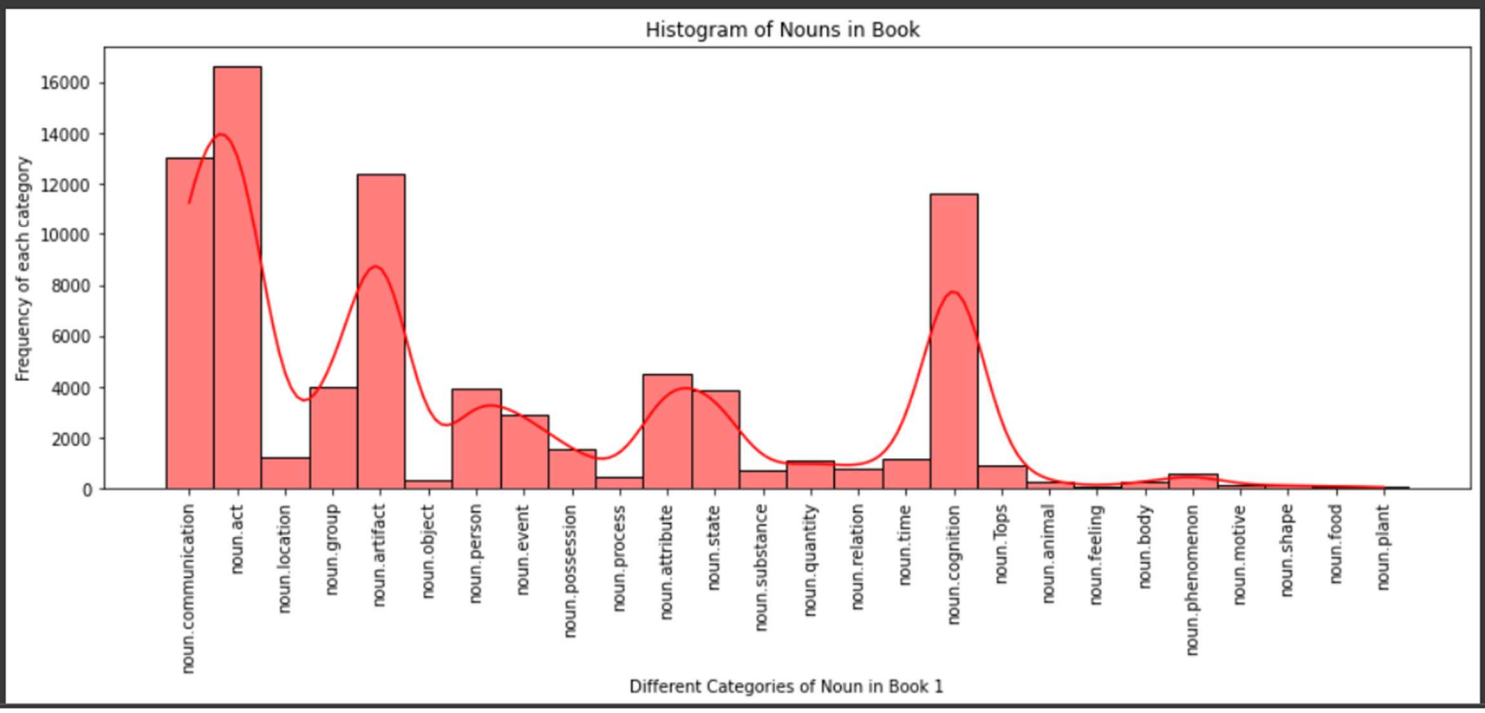
```
[33] nt1=[]

     for i in nouns_T1:
       syn =  wordnet.synsets(i,pos=wordnet.NOUN)
       if len(syn)>0:
         nt1.append(syn[0].lexname())

     print(list(zip(nouns_T1,nt1)))

[('software', 'noun.communication'), ('engineering', 'noun.act'), ('edition', 'noun.communication'), ('sommerville', 'noun.location'), ('boston', 'noun.location'), ('columbus', 'noun.
```

Similarly, The image below shows the categorization of verbs in the given Book and the results are stored in the list vt1. For example : ‘publishing’ categories to verb.consumption , ‘reserved’ categories to verb.communication and so on.

```
for i in verb_T1:
  syn =  wordnet.synsets(i,pos=wordnet.VERB)
  if len(syn)>0:
    vt1.append(syn[0].lexname())
#print(vt1)
print(list(zip(verb_T1,vt1)))

'publishing', 'verb.consumption'), ('reserved', 'verb.communication'), ('manufactured', 'verb.communication'), ('america', 'verb.perception'), ('protected', 'verb.creation'),
```

Step5: We plotted the histogram for the frequency plotting for different cateogries of nouns present in the book, and we can see from the histogram that for the given book sommerville, noun.communication has a frequency of around 13000+, and noun.act has more that 16000+ frequency which is the most in the book, and noun.plant has the least number of frequency count.

```
fig = plt.gcf()
fig.set_size_inches(15,5)
g = sns.histplot(x = nt1 , color = "red" , kde = True)
plt.setp(g.get_xticklabels() , rotation = 90)
plt.xlabel('Different Categories of Noun in Book 1')
plt.ylabel('Frequency of each category')
plt.title('Histogram of Nouns in Book ');
```

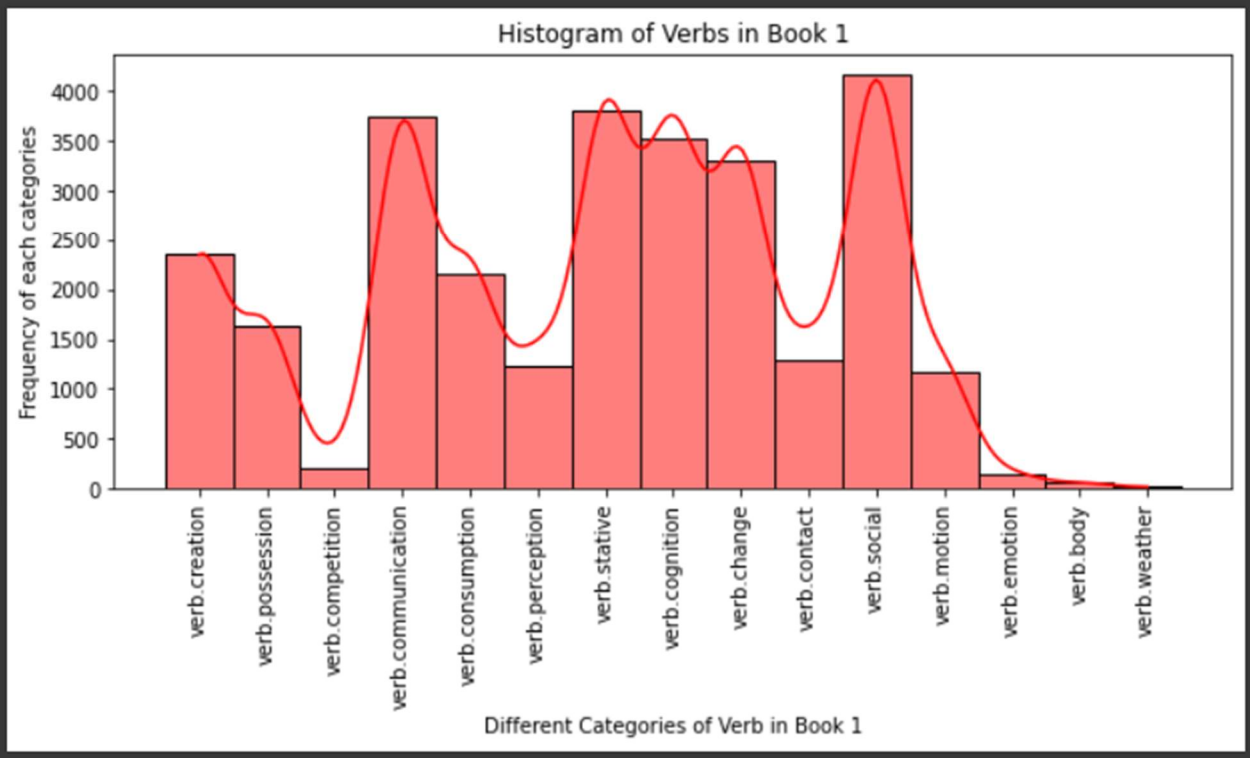


Noun labels in graph

['noun.act' 'noun.animal' 'noun.artifact' 'noun.attribute' 'noun.body' 'noun.cognition' 'noun.communication' 'noun.event' 'noun.feeling' 'noun.food' 'noun.group' 'noun.location' 'noun.motive' 'noun.object' 'noun.person' 'noun.phenomenon' 'noun.plant' 'noun.possession' 'noun.process' 'noun.quantity' 'noun.relation' 'noun.shape' 'noun.state' 'noun.substance' 'noun.time']

Similarly, For the verb also we plotted the frequency count in the form of histogram. For the given book , Verb.stative has 3600+ frequency count , verb.social has 4000+ frequency count which is maximum of all and verb.weather has the least count which is not more than 10.

```
[37] fig = plt.gcf()
     fig.set_size_inches(10,4)
     g = sns.histplot(x = vt1 ,color = "red" , kde = True)
     plt.setp(g.get_xticklabels() , rotation = 90)
     plt.xlabel('Different Categories of Verb in Book 1')
     plt.ylabel('Frequency of each categories')
     plt.title('Histogram of Verbs in Book 1');
```



Verb labels in graph

['verb.body' 'verb.change' 'verb.cognition' 'verb.communication' 'verb.competition' 'verb.consumption' 'verb.contact' 'verb.creation' 'verb.emotion' 'verb.motion' 'verb.perception' 'verb.possession' 'verb.social' 'verb.stative' 'verb.weather']

Second Part:

Recognise all entities. For this ,perform two steps:

(1) First recognise all the entity and then

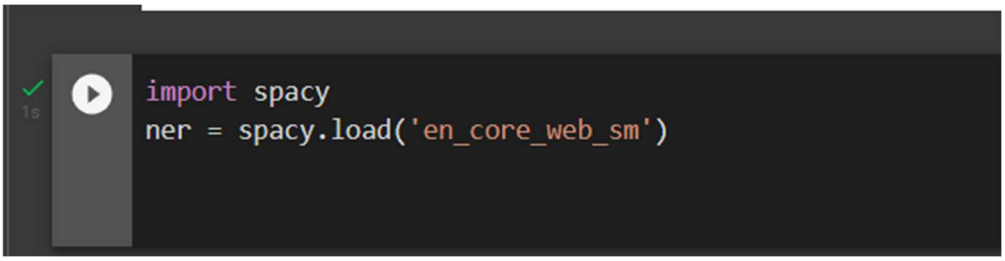
(2) recognise all entity types.

Use performance measures to measure the performance of the method used - For evaluation take the considerable amount of random passages from the book, perform manual labelling and then compare the result with it. Present the accuracy here and F1 score.

Named Entity Recognition (NER)

It is a sub part of Information extraction and retrieval. We have used the 'spacy' library of python for the same. Using 'en_core_web_sm.load()' inbuilt function we have named or labelled all the text of a book. Spacy supports PERSON, NORP (nationalities, religious and political groups), FAC (buildings, airports etc.), ORG (organizations), GPE (countries, cities etc.), LOC (mountain ranges, water bodies etc.), PRODUCT (products), EVENT (event names), WORK_OF_ART (books, song titles), LAW (legal document titles), LANGUAGE (named languages), DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL and CARDINAL entities.

Recognise all Persons, Location, Organisation (Types given in Fig 22.1) in book



The Spacy NER system contains a word embedding strategy using sub-word features, a "Bloom" embed, and a deep convolutional neural network with residual connections. To implement this, we imported spacy, "en_core_web_sm" is the pre-trained model that spacy uses, and with the help of this library, we recognized all the entities in the book.

```
[39] book1 = text[5000:1005000]
     entity_tag_book1 = []
     lst1 = ner(book1)
     for word in lst1.ents:
       x = word.text
       y = word.label_
       entity_tag_book1.append([x,y])

     print("there are total "+str(len(lst1.ents))+" entities in the book")

     there are total 3997 entities in the book
```

As we can see in the image , book is the raw text of our book and entity_tag_book is the list that has the entity name as well as it's corresponding entity type. And all the items of the list are displayed in the jupyter notebook. All the total entities present in the one million segment of the book are presented with the total count.

```
entity_tag_book1[:]

[['Andrea  Stefanowicz', 'PERSON'],
 ['GGS', 'ORG'],
 ['Higher Education  Resources', 'ORG'],
 ['Division', 'ORG'],
 ['PreMedia', 'ORG'],
 ['Global,  Inc.\nComposition  and Illustrations', 'ORG'],
 ['GGS', 'ORG'],
 ['Higher Education  Resources', 'ORG'],
 ['Division', 'ORG'],
 ['PreMedia Global', 'ORG'],
 ['Printer/Binder:  ', 'ORG'],
 ['Edwards', 'PERSON'],
 ['Lehigh-Phoenix  Color/Hagerstown\n\n\nCopyright  ©', 'ORG'],
 ['2011', 'DATE'],
 ['2006', 'DATE'],
```

As we can see in the image that all types of entities are recognized like CARDINAL, PERSON, ORG, DATE , MONEY and many more.

Extracting counts of Person (PER),Organization(Org) and Location(LOC)

```
[83] def entity_recognition(text):
       doc=nlp(text)
       person=[]
       org=[]
       location=[]
       for X in doc:
         if (X.ent_type_=='PERSON') and X.text not in person:
           person.append(X.text)
         if (X.ent_type_=='ORG')and X.text not in org:
           org.append(X.text)
         if ((X.ent_type_=='LOC') or (X.ent_type_=='GPE')) and X.text not in location:
           location.append(X.text)
       return person,org,location
```

```
person1,org1,location1=entity_recognition(book1)
print("number of person entities in the book are "+str(len(person1)))
print("number of organization entities in the book are "+str(len(org1)))
print("number of location entities in the book are "+str(len(location1)))
```

```
number of person entities in the book are 279
number of organization entities in the book are 601
number of location entities in the book are 141
```

Evaluating the accuracy of the Named Entity Recognition the book.

Here, finaltext is the whole preprocessed text of our book after removal of the headers and the footers and lst3 is the first 3000 characters of Book 1 used for the Named Entity Recognition. As we can see from the image entity_tag_passage_book1 is a list that shows the entities with their labels marked by the model itself . For example ‘new york’ is a name of a person so it has been labeled as GPE and so on.

```
entity_tag_passage_book1 = []
lst3 = ner(finaltext[0:3000])
for word in lst3.ents:
  x = word.text
  y = word.label_
  entity_tag_passage_book1.append([x,y])

entity_tag_passage_book1
```

```
[['2', 'CARDINAL'],
 ['ninth', 'ORDINAL'],
 ['ian', 'NORP'],
 ['addisonwesley  boston', 'ORG'],
 ['new  york  san  francisco  upper  saddle  ', 'GPE'],
 ['riveramsterdam  cape', 'PERSON'],
 ['dubai', 'GPE'],
 ['london', 'GPE'],
 ['madrid', 'GPE'],
 ['milan', 'GPE'],
 ['munich', 'GPE'],
 ['paris', 'GPE'],
 ['montreal', 'GPE'],
 ['toronto', 'GPE'],
 ['delhi', 'GPE'],
 ['mexico  city', 'GPE'],
 ['hong  kong', 'GPE'],
 ['seoul', 'GPE'],
 ['singapore', 'GPE'],
```

0s completed at 18:24

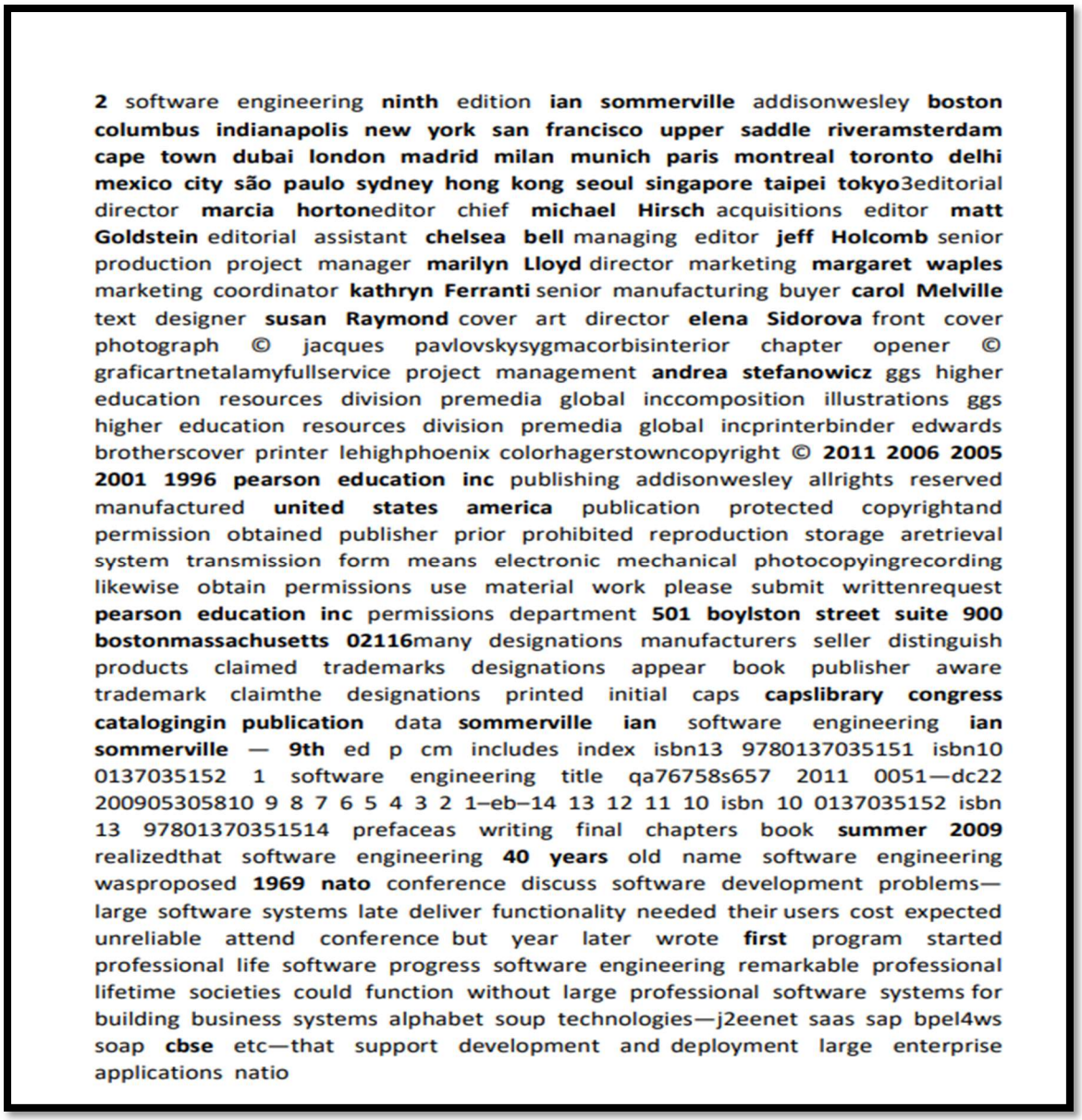
```
para_T1 = finaltext[0:3000]
docm = ner(para_T1)

displacy.render(docm, jupyter=True, style='ent')
```

2 CARDINAL software engineering ninth ORDINAL edition ian NORP sommerville addisonwesley boston ORG columbus indianapolis new york san francisco upper saddle GPE riveramsterdam cape PERSON own dubai GPE london GPE madrid GPE milan GPE munich GPE paris GPE montreal GPE toronto GPE delhi GPE mexico city GPE são paulo sydney hong kong GPE seoul GPE singapore GPE taipei GPE tokyo3editorial director marcia hortoneditor chief michael PERSON hirschacquisitions editor matt PERSON goldsteineditorial assistant chelsea bellmanaging editor jeff PERSON olcombsenior production project manager marilyn lloyddirector marketing margaret PERSON waplesmarketing coordinator kathryn PERSON ferrantisenior manufacturing buyer carol melvilletext designer susan aymondcover art director elena sidorovafront cover photograph © jacques pavlovskysygmacorbisinterior chapter opener © graficartnetalamyfullservice project management andrea PERSON stefanowicz ggs PERSON igher education resources division premedia global inccomposition illustrations ggs PERSON higher education resources division premedia global incprinterbinder edwards brotherscover printer lehighphoenix olorhagerstowncopyright © 2011 2006 2005 2001 1996 DATE pearson education inc publishing addisonwesley allrights reserved manufactured united states america GPE publication protected copyrightand permission btained publisher prior prohibited reproduction storage aretrieval system transmission form means electronic mechanical photocopyingrecording likewise obtain permissions use material work please submit writtenrequest earson education inc permissions department 501 CARDINAL boylston street suite 900 bostonmassachusetts 02116many designations manufacturers seller distinguish products claimed trademarks designations appear ook publisher aware trademark claimthe designations printed initial caps capslibrary congress ORG cataloginginpublication datasommerville ian NORP software engineering ian NORP sommerville — 9th ORDINAL d p cm includes index isbn13 PERSON 9780137035151 DATE isbn10 ORG 0137035152 1 CARDINAL software engineering title qa76758s657 2011 0051 DATE —dc22 200905305810 DATE 9 CARDINAL 8 7 6 5 4 3 2 1–eb–14 13 CARDINAL 12 11 10 CARDINAL isbn 10 CARDINAL 0137035152 isbn 13 CARDINAL 97801370351514 DATE prefaceas writing final chapters book summer 2009 DATE ealizedthat software engineering 40 years old DATE name software engineering wasproposed 1969 DATE nato ORG conference discuss software development problems—large software systems late deliver unctionality needed theirusers cost expected unreliable attend conferencebut year later DATE wrote first ORDINAL program started professional life software progress software engineering remarkable professional fetime societies could function without large professional software systemsfor building business systems alphabet soup technologies—j2eenet saas sap bpel4ws soap cbse etc—that support development anddeployment large

Total number of entities marked by the model is 53 in which can be depicted by the displacy render highlighting all the entities with coloured marks. These entities recorded by the modal are then compared with entities checked manually to calculate the accuracy and performance measures.

Manually Labelling of sample Text

The text below represents the first 3000 characters of the text of Book 1 and in the figure all the Bold words represent the Entities marked manually. There are a total 62 entities that we have found in the passage and the Model shown above catches 53 entities.



2 software engineering **ninth** edition **ian sommerville** addisonwesley **boston columbus indianapolis new york san francisco upper saddle riveramsterdam cape town dubai london madrid milan munich paris montreal toronto delhi mexico city são paulo sydney hong kong seoul singapore taipei tokyo**3editorial director **marcia hortoneditor** chief **michael Hirsch** acquisitions editor **matt Goldstein** editorial assistant **chelsea bell** managing editor **jeff Holcomb** senior production project manager **marilyn Lloyd** director marketing **margaret waples** marketing coordinator **kathryn Ferranti** senior manufacturing buyer **carol Melville** text designer **susan Raymond** cover art director **elena Sidorova** front cover photograph © jacques pavlovskysygmacorbisinterior chapter opener © graficartnetalamyfullservice project management **andrea stefanowicz** ggs higher education resources division premedia global inccomposition illustrations ggs higher education resources division premedia global incprinterbinder edwards brotherscover printer lehighphoenix colorhagerstowncopyright © **2011 2006 2005 2001 1996 pearson education inc** publishing addisonwesley allrights reserved manufactured **united states america** publication protected copyrightand permission obtained publisher prior prohibited reproduction storage aretrieval system transmission form means electronic mechanical photocopyingrecording likewise obtain permissions use material work please submit writtenrequest **pearson education inc** permissions department **501 boylston street suite 900 bostonmassachusetts 02116**many designations manufacturers seller distinguish products claimed trademarks designations appear book publisher aware trademark claimthe designations printed initial caps **capslibrary congress catalogingin publication** data **sommerville ian** software engineering **ian sommerville** — **9th** ed p cm includes index isbn13 9780137035151 isbn10 0137035152 1 software engineering title qa76758s657 2011 0051—dc22 200905305810 9 8 7 6 5 4 3 2 1–eb–14 13 12 11 10 isbn 10 0137035152 isbn 13 97801370351514 prefaceas writing final chapters book **summer 2009** realizedthat software engineering **40 years** old name software engineering waproposed **1969 nato** conference discuss software development problems—large software systems late deliver functionality needed their users cost expected unreliable attend conference but year later wrote **first** program started professional life software progress software engineering remarkable professional lifetime societies could function without large professional software systems for building business systems alphabet soup technologies—j2eenet saas sap bpel4ws soap **cbse** etc—that support development and deployment large enterprise applications natio

Figure 1The sample text from the book for the evaluation of accuracy

The numbers of total entities that needs to be marked = 62

Total number of entities labelled by the model = 53

Number of the correctly labelled entities = 42

Precision (p) = Total correctly labelled entities by the total number of labeled entities
= 42 / 53 = 0.7924

Recall (r) = Total correctly labelled entities by the total entities that need to be labelled.
= 42 / 62 = 0.6774

F- measure : F1 score = 0.7304

Third Part:

For extracting the relationship between the entities from the book - what are the features necessary for this? Use the ideas given in the book and presented in the class to augment the data of Entities and augment that data by extracting additional features and build the table and present it.

Extract the relationship between the entities:

Get entity pairs

The first step is to get entity pairs on which the relations are present using the following Code:

```
def get_entities(sent):
  ## chunk 1
  ent1 = ""
  ent2 = ""

  prv_tok_dep = ""    # dependency tag of previous token in the sentence
  prv_tok_text = ""   # previous token in the sentence

  prefix = ""
  modifier = ""

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

  for tok in nlp(sent):
    ## chunk 2
    # if token is a punctuation mark then move on to the next token
    if tok.dep_ != "punct":
      # check: token is a compound word or not
      if tok.dep_ == "compound":
        prefix = tok.text
        # if the previous word was also a 'compound' then add the current word to it
        if prv_tok_dep == "compound":
          prefix = prv_tok_text + " "+ tok.text

      # check: token is a modifier or not
      if tok.dep_.endswith("mod") == True:
        modifier = tok.text
        # if the previous word was also a 'compound' then add the current word to it
        if prv_tok_dep == "compound":
          modifier = prv_tok_text + " "+ tok.text

      ## chunk 3
      if tok.dep_.find("subj") == True:
        ent1 = modifier +" "+ prefix + " "+ tok.text
        prefix = ""
        modifier = ""
        prv_tok_dep = ""
        prv_tok_text = ""

      ## chunk 4
      if tok.dep_.find("obj") == True:
        ent2 = modifier +" "+ prefix +" "+ tok.text
```

Then store and extract sentence by sentence in the csv file for effective evaluation and ease of relation extraction.

```
[60] sentences = [[i] for i in nlp(finaltext[1000:1001000]).sents]

[61] import csv
     myheaders = ['sentence']
     myvalues = sentences
     filename = 'text.csv'
     with open(filename, 'w',newline='') as myfile:
         writer = csv.writer(myfile)
         writer.writerow(myheaders)
         writer.writerows(myvalues)

[62] csv_sentences = pd.read_csv("text.csv")

[63] csv_sentences
```

List down the entity pairs from which relations are devised:

```
[64] entity_pairs = []

     for i in tqdm(csv_sentences["sentence"]):
       entity_pairs.append(get_entities(i))

     100%|██████████| 691/691 [00:39<00:00, 17.71it/s]

[65] entity_pairs[:]
     ['77 gram diagram', 'external system requests'],
     ['satcomms information system', 'external stick system'],
     ['various itresponds services', 'diagram'],
     ['normal  operation', 'reconfigure messages'],
     ['system', 'down2'],
     ['state messages', 'returns shutdown'],
     ['remote state diagrams', 'usually objects system'],
     ['et al 1977', 'effective in207190 chapter'],
     ['details', 'minimal knowledge optimizations'],
     ['impractical changes observers', 'theobserver pattern description'],
     ['common software problems', 'four book gamma'],
     ['important pattern descriptions', 'european technology buschmann'],
     ['often inheritance polymorphism', 'generality'],
     ['general  principle', 'experience'],
     ['one host system', 'separate target systems'],
     ['applicable development costs', 'large systems costs'],
     ['tools', 'bestknown environment carlson'],
```

Define the Relationship function:

```
[78] def get_relation(sent):

       doc = nlp(sent)

       # Matcher class object
       matcher = Matcher(nlp.vocab)

       #define the pattern
       pattern = [{'DEP':'ROOT'},
                  {'DEP':'prep','OP':"?"},
                  {'DEP':'agent','OP':"?"},
                  {'POS':'ADJ','OP':"?"}]

       matcher.add("matching_1",[pattern])

       matches = matcher(doc)
       k = len(matches) - 1

       span = doc[matches[k][1]:matches[k][2]]

       return(span.text)
```

Get the relationship table:

For the pattern defined above in the function , the above function will evaluate all the relations that exist between the entity pairs and list down all the relations as below.
For different patterns , different relations such as Person-Person Relationship Extraction , Person-Organization Relationship, Person - Location Relationship etc can be extracted as well.
We will use different patterns to get all the possible relations.

```
relations = [get_relation(i) for i in tqdm(csv_sentences['sentence'])]
```

```
[80] pd.options.display.max_rows = 400
     pd.Series(relations).value_counts()[:]
```

used	15
shows	11
include	11
think	11
developed	10
use	10
say	9
provides	9
reuse	9
shown	8
discuss	8
allows	8
allow	7
suggest	7
adds	7
objectoriented	6
need	6
suggests	6
make	6
define	6
made	5
means	5
requires	5
reused	5
understand	5
see	5
create	5
includes	5

Pattern 2:-

We are now defining a new pattern which we will be extracting from the text. The pattern defined as : the first words should be an entity with type ‘person’ followed by the noun and pos tag noun and ends with a verb. We used the matcher object from the spacy library to match sequences of tokens based on pattern rules. After running the following code snippet, we found 15 matches.

```
pattern = [{'ENT_TYPE':'PERSON'},
           {'POS':'NOUN','OP':"?"},
           {'POS':'NOUN','OP':"?"},

           {'POS':'VERB'}
           ]
```

```
[126] # Matcher class object
      matcher = Matcher(nlp.vocab)
      matcher.add("matching1",[pattern])

      matches = matcher(doc)
```

```
[127] len(matches)

      15
```

Relations founded are below for the pattern 2:

```
[128] for i in range(15):
        span = doc[matches[i][1]:matches[i][2]]
        print(span)

      Boehm identifies
      Krutchen describes
      Java using
      Scenariofor collecting
      Controller component manages
          testing
          testing
      Java allows
         testing
      Kate logs
      Jim feels
      Kate looks
      Kate enters
      Kate returns
      Belady suggest
```

Pattern 3:

Now again, we used some new patterns to obtain some more relationship between entities

```
[129] matcher2 = Matcher(nlp.vocab)

      #define the pattern
      pattern = [

                  {'ENT_TYPE':'PERSON'},
                  {'POS':'DET','OP':"?"},
                  {'LOWER': 'has', 'OP':"?"},
                  {'LOWER': 'lives', 'OP':"?"},
                  {'ENT_TYPE':'GPE','OP':"?"},
                  {'ENT_TYPE':'ORG','OP':"?"},
                  {'LOWER': 'on', 'OP':"?"},
                  {'POS':'NOUN'}]

      matcher2.add("matching_1", [pattern])

      matches = matcher2(doc)
      len(matches)

      31
```

Relations extracted:

```
for i in range(len(matches)):
  span = doc[matches[i][1]:matches[i][2]]
  print(span)

Software systems
                                    validation
Software reuse
that5538 Chapter
Tiny Fingers
                                                        Chapter
this7962 Chapter
     software
r1
Software systems
                  modeling
Rumbaugh et
Bass et
Bass et
View component
Controller component
mouse clicks
Gamma et
Schmidt et
Hunter et
                                    the systems
                    the principles
                                    teristics
Gamma et
Dijkstra et
    testing
    testing
      testing

C. A.
```

Pattern 4:

```
[130] pattern = [ {'POS':'PRON'},
                  {'POS':'NOUN'},

                   {'POS':'DET','OP':"?"},
                  {'POS':'VERB'},
                {'POS':'NOUN','OP':"?"},

                  ]

      matcher1 = Matcher(nlp.vocab)
      matcher1.add("matching_1", [pattern])

      matches = matcher1(doc)
      print(len(matches))

      14
```

Relations extracted:

```
[131] for i in range(len(matches)):
        span = doc[matches[i][1]:matches[i][2]]
        print(span)

      your answer based
      its success depends
      your company decides
      its meaning depends
      what system features
      Their processing involves
      your team proposes
      their changes have
      your company goes
      your computation involves
      your computation involves division
      his medication changed
      its structure tends
      its structure tends
```

Pattern 5:

```
[132] pattern = [ {'ENT_TYPE':'PERSON','OP':"?"},


                  {'LOWER': 'and','OP':"?"},
                  {'POS':'DET'},
                 {'ENT_TYPE':'GPE'},
                 {'POS':'NOUN','OP':"?"}]

      matcher1 = Matcher(nlp.vocab)
      matcher1.add("matching_1", [pattern])

      matches = matcher1(doc)
      print(len(matches))

      12
```

Relations extracted:

```
for i in range(len(matches)):
  span = doc[matches[i][1]:matches[i][2]]
  print(span)

this Preface
the SAP
the SAP system
the United
The Client
the
the                                    input
a Transmitting
a Transmitting state
the Transmitting
the Transmitting state
the Linux
```

We've updated the complete code on our GitHub repository and Jupyter Notebook on Google Colab.

Link for the same:

Github: https://github.com/utkarshk30/Intelligencia_NLP-Project-Rounds

Google Colab:
https://colab.research.google.com/drive/1Dnntl-6yXp-HEbVSz8yyoTMWbi0pa40j?usp=sharing

References:

- https://www.nltk.org/
- https://web.stanford.edu/~jurafsky/slp3/
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html
- https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-us
- e-it-2b68cf6f545d
- Class Video Lectures and Slides
- https://spacy.io/api/matcher