NATURAL LANGUAGE PROCESSING

Team Infinity

Aarya Garg (20UCS002)

Harshal Jain (20UCS077)

Suhani Sharma (20UCS202)

Yash Jain (20UCS237)

Course Instructor

Dr. Sakthi Balan Muthiah

Departmentof Computer Science Engineering

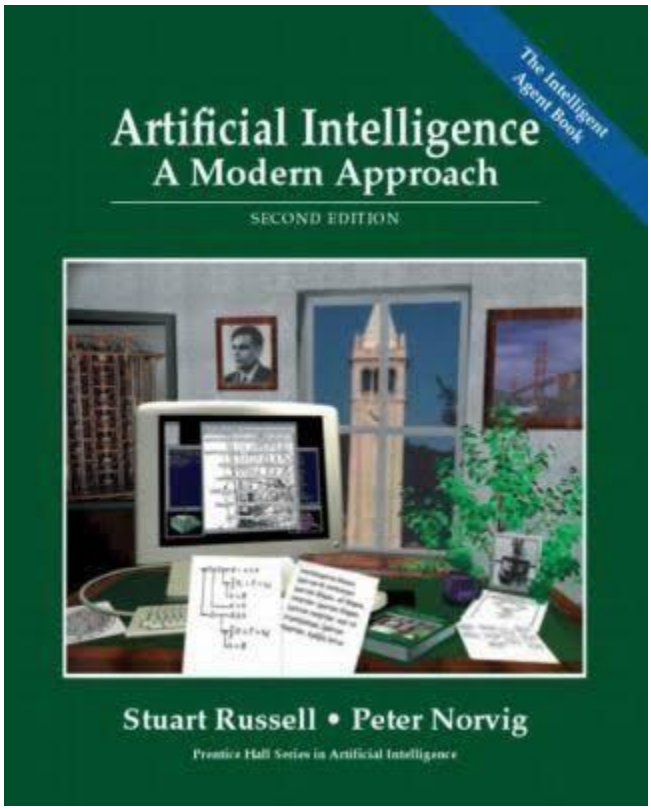
The LNM Institute of Information Technology

Project Round 1

Overview

In this project, we will be analyzing the textbook by Russell and Norvig. We will next proceed and apply POS Tagging to the textbook. With the aid of Python libraries, we will use NLP frameworks to accomplish all of this.

Book used



Artificial Intelligence: A Modern Approach

Textbook by Peter Norvig and Stuart J. Russell

Goals

1. Import the text from the book in text format and call it ‘text’.

   (Tool used https://cloudconvert.com/pdf-to-txt)
2. Pre-process the text
3. Tokenize the text
4. Analyze the frequency distribution of tokens in the text
5. Create the Word Cloud using tokens
6. Remove the stopwords from text and again create a word cloud.
7. Compare frequencies with word clouds after removal of stopwords.
8. Analyzing the word length and evaluating the relationship between the word length and frequency for T1.
9. Do PoS Tagging for text and Get the distribution of various tags.

Specifications

Python Libraries used in this project

- Pandas - Used for data manipulation and analysis
- NLTK - Used for Tokenizing, Lemmatization and Removing Stopwords
- Re - Used to remove URLs and Decontract Contractions in English Language
- Wordcloud - Used to create WordClouds from Tokenized Data
- Inflect - Used to replace numbers with words
- Maplotlib - Used to Visualize our text data

Description Of Data

After converting the pdf into txt format the text looks like this

```
In which we try to explain why we consider artificial intelligence to be a subject most worthy of study, and in which we try to decide what exactly it is,
ARTIFICIAL   more complicated than itself. The field of artificial intelligence, or AI, goes further still: it INTELLIGENCE attempts not just to understand
AI is one of the newest fields in science and engineering. Work started in earnest soon after World War II, and the name itself was coined in 1956. Along
AI currently encompasses a huge variety of subfields, ranging from the general (learning and perception) to the specific, such as playing chess, proving m
intellectual task; it is truly a universal field. 1.1 WHAT IS AI?
We have claimed that AI is exciting, but we have not said what it is. In Figure 1.1 we see eight definitions of AI, laid out along two dimensions. The def



                                                                          1


21
2                                                                                                        Chapter 1.         Introduction


                    Thinking Humanly                                             Thinking Rationally

                    “The exciting new effort to make comput-        “The study of mental faculties through the
                    ers think ...machines with minds,inthe use of computational models.”
                    full and literal sense.” (Haugeland, 1985)        (Charniak and McDermott, 1985)

                    “[The automation of] activities that we              “The study of the computations that make
                    associate with human thinking, activities         it possible to perceive, reason, and act.”
                    such as decision-making, problem solv-          (Winston, 1992)
                    ing, learning ...” (Bellman, 1978)

                    Acting Humanly                                                 Acting Rationally

                    “The art of creating machines that per-            “Computational Intelligence is the study
                    form functions that require intelligence            of the design of intelligent agents.” (Poole
                    when performed by people.” (Kurzweil,           et al., 1998)
                    1990)
```

Observations

1. There are certain numbers that are not immediately relevant to humans, therefore we must convert them to words.
2. In English, there are a lot of contractions and punctuation, and we need to turn them into useful data.
3. There are several special characters and URLs.
4. Emojis, emoticons, and chat phrases are not available
   #THIS CAN BE IMPROVE

Tasks

1. Importing the book

```
[31] #To open the file
     file = open(r"/content/NlpBook.txt",encoding='utf-8')
     listofwords = file.read().splitlines()
     listofwords = [i for i in listofwords if i!='']
     text = ""
     text = text.join(listofwords)
```

2. Text Preprocessing Steps

For preprocessing we have used following functions: -

a. Filtering the punctuations from the text and converting it into the lower case to make it relevant : -

```
#Defining string which contains the punctuations to reject
punctuations = '''!()-[]{};:'"\,<>./‘’?“”@#$%^&*_~'''
filteredtext = ""
for i in text:
    if i not in punctuations:
        filteredtext = filteredtext + i

#Making the result lowercase
filteredtext = filteredtext.lower()
```

Now the text is filtered and labeled as ‘filteredtext’

3. Tokenizing

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

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

```
tokenisedtext = word_tokenize(filteredtext)
print(tokenisedtext[:10]) #Printing the first 10 tokens of filtered text.
```

Using nltk.FreqDist() function to calculate the frequency of tokens

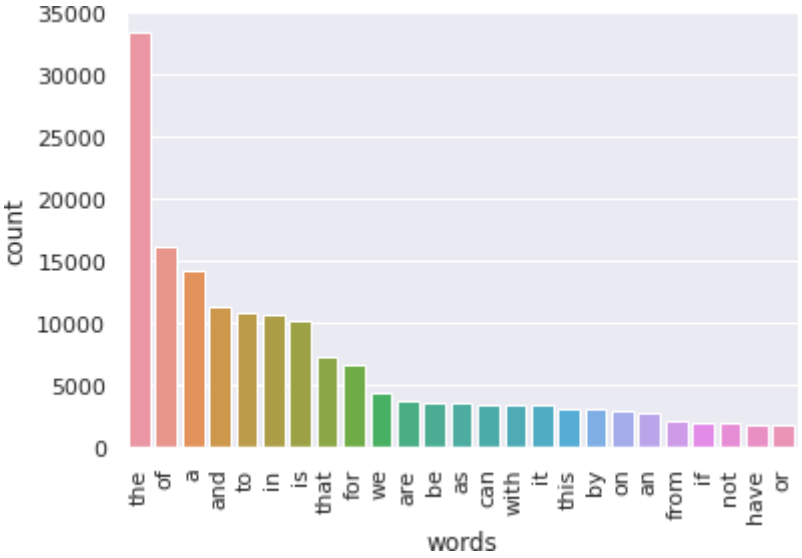
```
frequency_distribution=nltk.FreqDist(filteredtext)

print(frequency_distribution.most_common(10))
freq_dist = list(frequency_distribution)

[(' ', 2327005), ('e', 315885), ('t', 240984), ('a', 216459), ('i', 202808),
```

4. Visualising the frequency distribution of tokens

We use **seaborn** plot to analyze the frequency of the tokens on our dataset. We check the first 25 most occurring tokens.

```
import seaborn as sb
sb.set(style='darkgrid')
dataf=pd.DataFrame(tokenisedtext)
```

```
sb.countplot(x=dataf[0],order=dataf[0].value_counts().iloc[:25].index)
plt.xticks(rotation=90)
plt.xlabel('words')
plt.show()
#Plotting the counts of the most frequent words ordered in descending order of their frequencies.
```

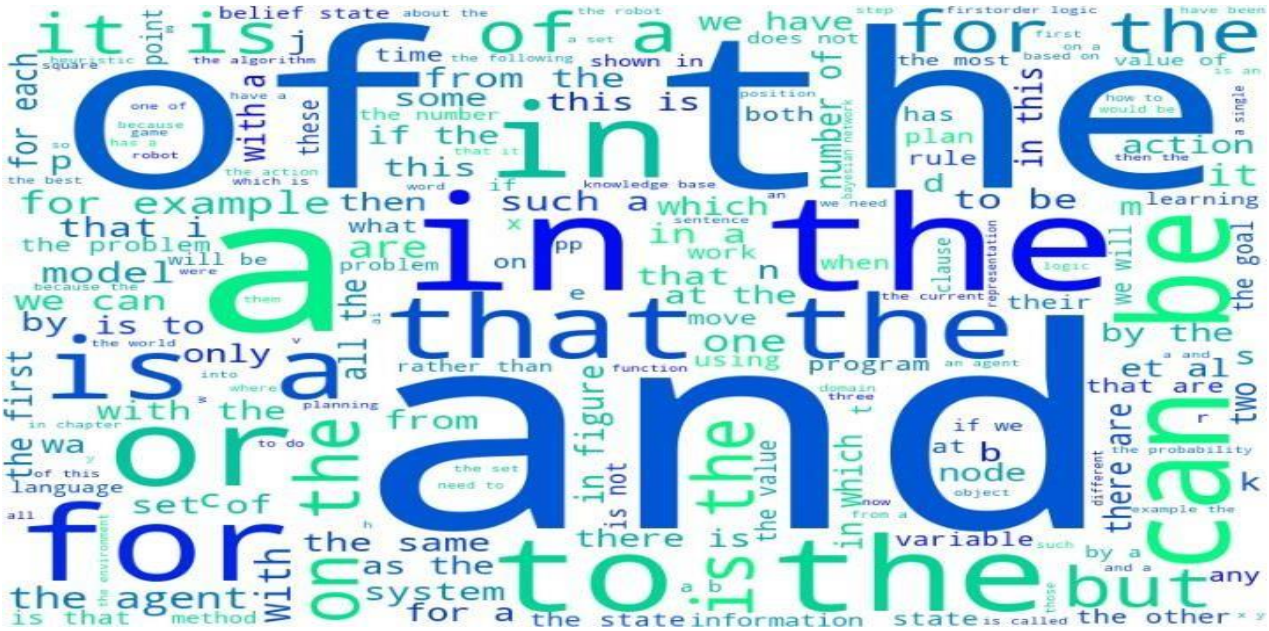


Here most frequent words are plotted with their counts respectively

5. Creating word-cloud with Stop Words

```
Stopwordswordcloud = WordCloud(width = 800, height = 600,
                background_color ='white',
                min_font_size = 10,stopwords = {},colormap='winter').generate(filteredtext)
```

```
plt.figure(figsize = (10,8), facecolor = None)
plt.imshow(Stopwordswordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



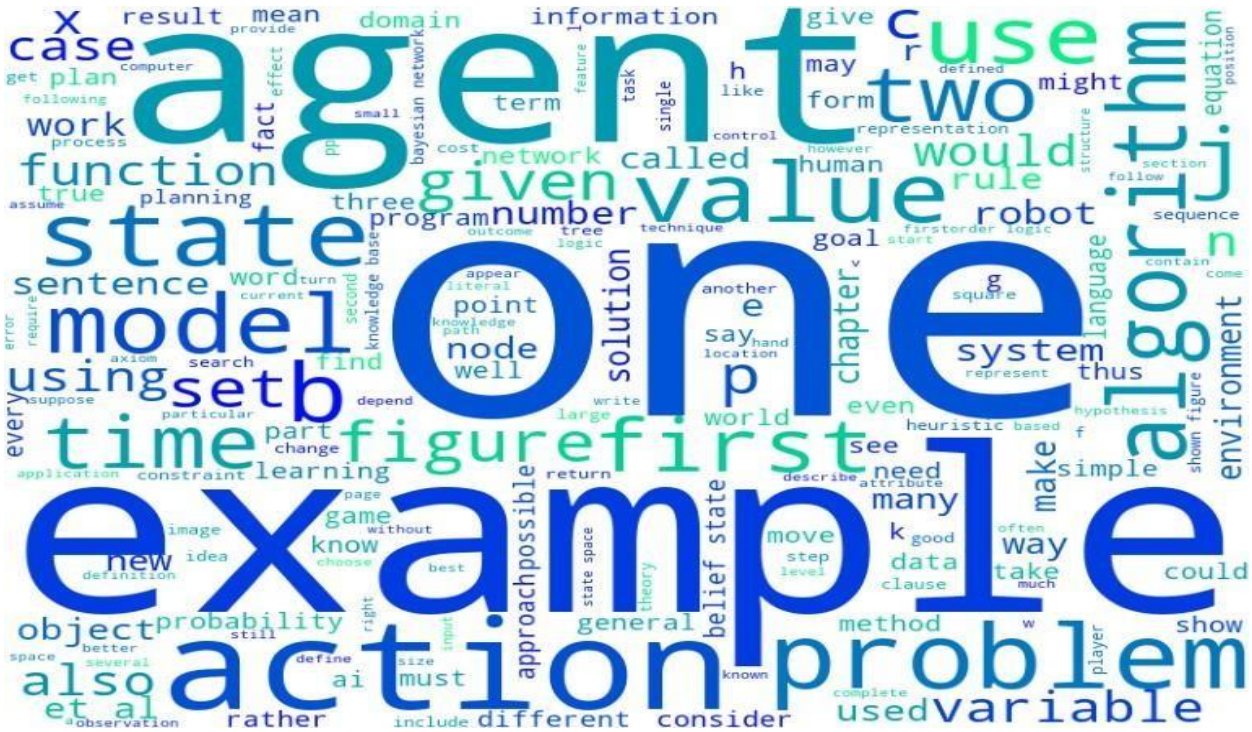
6. Removing the Stop Words and again creating the word cloud

a. Stop word removal :

```
[40] # Stop words need to be removed. So to remove them
     stop_words = set(stopwords.words('english'))
     tokens = word_tokenize(filteredtext)
     final_tokens = [i for i in tokens if not i in stop_words]
     finaltext = " "
     finaltext = finaltext.join(final_tokens)
```

b. Creating the word cloud

```
# Word cloud after Stopwords removal
wc_withoutStopwords = 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(wc_withoutStopwords)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



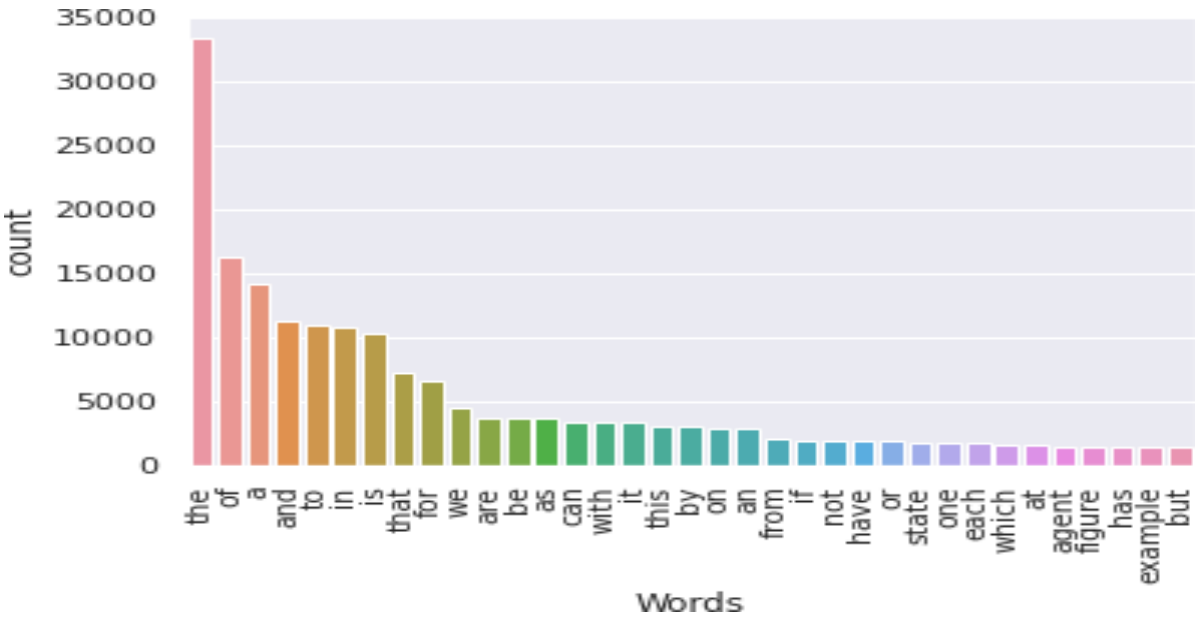
We used the WordCloud function to create word cloud of our dataset without stopwords
After comparing both the word clouds we can observe that previously there were less important words present in the word cloud but after filtering the dataset (i.e. removing the stop words), more important words are given importance in the word cloud.

7. Analysing the frequencies after stopwords removal

```
[42] new_freq_dist=nltk.FreqDist(finaltext)
     print(new_freq_dist.most_common(15))
     freq_dist=list(new_freq_dist)
```

```
     [(' ', 675318), ('e', 246578), ('i', 163375), ('a', 156054), ('t', 153563), ('n', 150480), ('s', 148483), ('r', 136382), (
```

```
new_dataframe=pd.DataFrame(tokenisedtext)
sb.countplot(x=new_dataframe[0], order=new_dataframe[0].value_counts().iloc[:35].index)
plt.xticks(rotation=90)
plt.xlabel('Words')
plt.show()
#Plotting the count after stopwords removal
```



8. Analysing the word length after removing the stop words

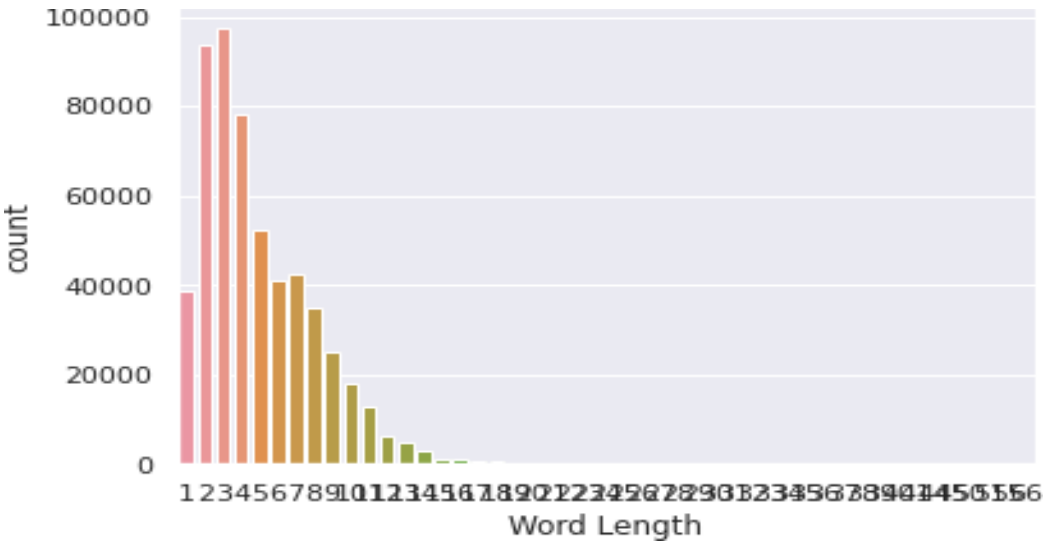
```
[44] length=[]
     for x in tokenisedtext:
      length.append(len(x))
     length[:20]
```

```
[45] frequency_distribution_length=nltk.FreqDist(length)
```

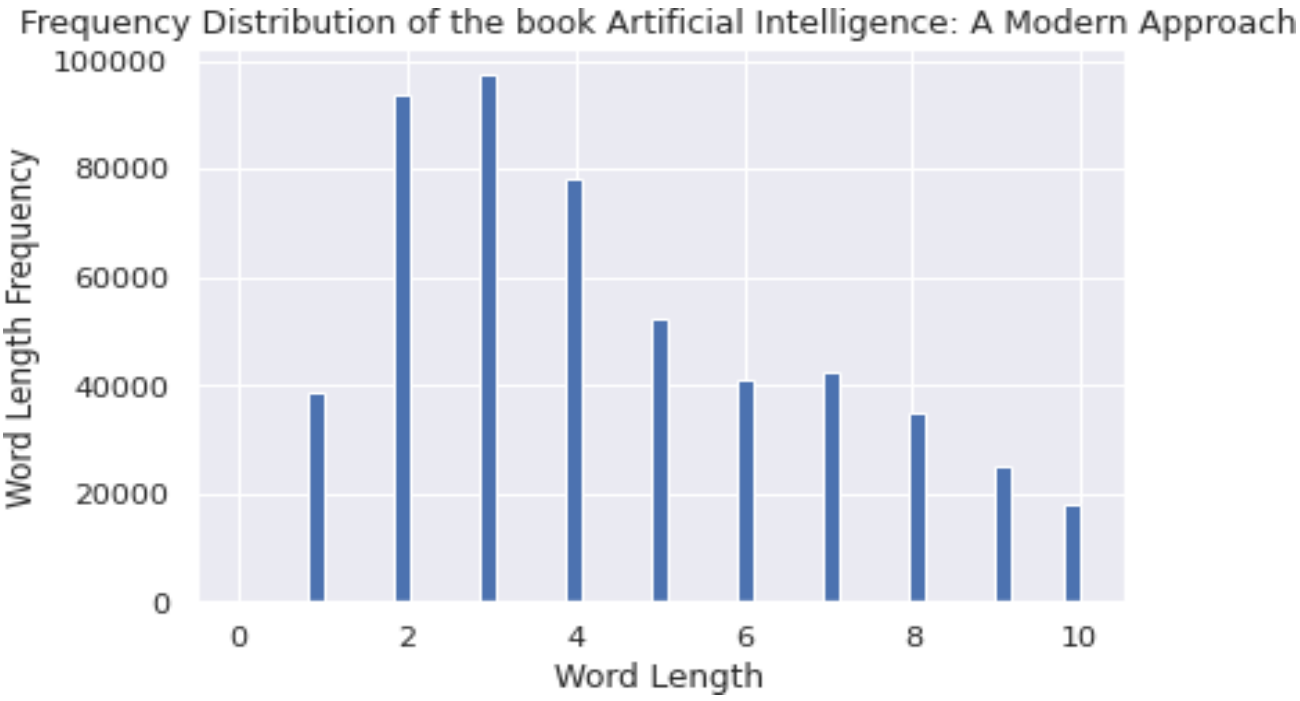
```
[46] frequency_distribution_length
```

```
     FreqDist({3: 97315, 2: 93448, 4: 78358, 5: 52272, 7: 42332, 6: 40948, 1: 38872, 8: 34730, 9: 24855, 10: 18177, ...})
```

```
dataf_length=pd.DataFrame(length)
sb.countplot(x=dataf_length[0])
plt.xlabel('Word Length')
plt.show()
```




```
import numpy as np
bin_size=np.linspace(0,10)

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

#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 of the book Artificial Intelligence: A Modern Approach')
plt.show()
```



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

9. Tagging the Parts of Speech using Treebank tagset

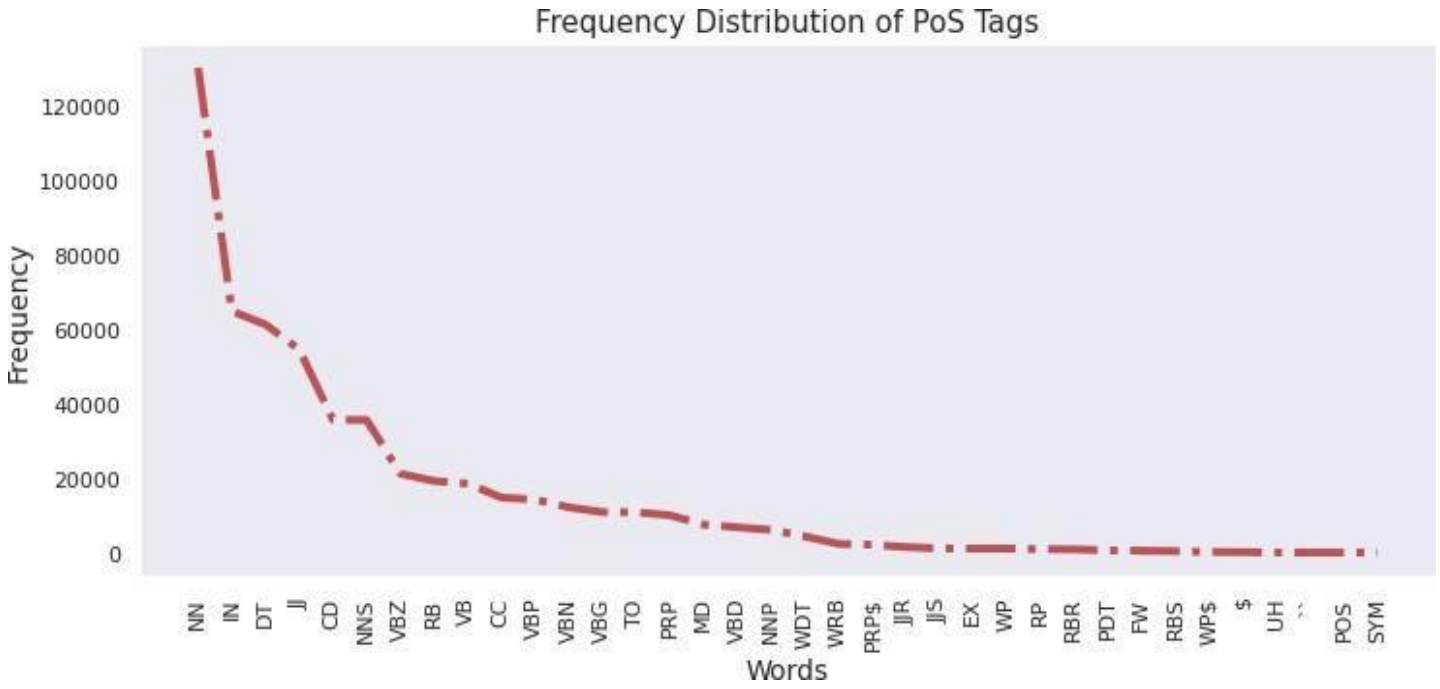
```
tagging = nltk.pos_tag(tokenisedtext)
tagging[:10] #First 10 tags
```

```
[('in', 'IN'),
 ('which', 'WDT'),
 ('we', 'PRP'),
 ('try', 'VBP'),
 ('to', 'TO'),
 ('explain', 'VB'),
 ('why', 'WRB'),
 ('we', 'PRP'),
 ('consider', 'VBP'),
 ('artificial', 'JJ')]
```

```
from collections import Counter
counts = Counter( tag for word,  tag in tagging)
print(counts)
```

```
Counter({'NN': 130391, 'IN': 64912, 'DT': 61347, 'JJ': 54606, 'CD': 35706, 'NNS': 35689, 'VBZ': 21215, 'RB': 19264, 'VB': 18565,
```

```
#Frequency Distribution of PoS Tags

pos_tags_freq = nltk.FreqDist(counts)
pos_tags_freq = {k: v for k, v in sorted(pos_tags_freq.items(), key=lambda item: item[1],reverse=True)}
x = list(pos_tags_freq.keys())[:40]
y = list(pos_tags_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('Frequency Distribution of PoS Tags',size=15)
plt.xlabel('Words',size=14)
plt.ylabel('Frequency',size=14)
plt.show()
```



After we thoroughly clean our data, we proceed with the text processing part. Here, we have to assign appropriate Part-of-Speech Tags to the words. For this, we use the function “pos_tag( )” from the NLTK library of python.

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

Inference

We applied pos_tagging on the two books using pos_tag function. The pos_tag(words) function uses the Penn treebank as the default tag set as per official documentation.

In The textbook the most frequently occurring POS Tag is ‘NN’ with count 130391 followed by ‘IN’ having count 64912.

Conclusion

In this Round 1 of our project, we performed the tasks of word pre-processing, word tokenization, Word Cloud generation,POS tagging and also deduced many inferences from them about the bookswhile also learning in the process.

Project Round 2

Goals :

Task - 1:

1. Find the nouns and verbs in the book. Get the categories that these words fall under in theWordNet. 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 heirarchiesand plot a histogram for the same.

Task - 2:

1. Recognise all entities (Types given in Fig 22.1). For this you have to do 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 you take a considerable amountof random passages from the book, do a manual labelling and then compare your result with it.Present the accuracy here and F1 score.

Task - 3:

1. For extracting the relationship between the entities from the book - what are the features necessaryfor 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.

SPECIFICATIONS:

1. Python Libraries used in this project:

2. Urllib : Used to fetch text data from Gutenberg URLs

3. NLTK : Used for Tokenizing, implementing wordnet, POS tagging etc.

4. Re : Used toremove URLs and Decontract Contractions in English Language

5. Matplotlib : Used to Visualize our text data Spacy- To perform Entity recognition in text

6. NumPy : To get frequency distributions of nouns and verbs

7. Typing : To perform evaluation of the Algorithm in Entity Recognition

Important libraries and imports:

```
import urllib.request
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import nltk
import re
import inflect
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
from collections import Counter

nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

Tasks:

Task - 1

1. In the book, look for the nouns and verbs. Find the wordnet categories that theseterms immediately belong to.

a. Finding Nouns and Verbs: -

```
def give_noun(tag):
    is_noun = lambda pos: pos[:1] == 'N'
    return [word for (word, pos) in tag if is_noun(pos)]

def give_verb(tag):
    is_verb = lambda pos: pos[:1] == 'V'
    return [word for (word, pos) in tag if is_verb(pos)]

tag1 = pos_tagging(text1)

noun_book_1 = give_noun(tag1)
verb_book_1 = give_verb(tag1)

print(len(noun_book_1), noun_book_1)
print(len(verb_book_1), verb_book_1)
```

Output: -

```
16595 ['start', 'project', 'ebook', 'war', 'world', 'cover', 'war', 'world', 'h', 'g', '', 'world', 'inhabit', 'world', 'thing', 'man', 'kepler', 'quot'
4517 ['dwell', 'made', 'come', 'begin', 'saw', 'fell', 'happen', 'exodu', 'saw', 'imprison', 'believ', 'centuri', 'busi', 'scrutinis', 'went', 'assur', '
```

b. Categories in WordNet: -

```
from nltk.corpus import wordnet as wn
def categories_word(words):
    categories = []
    for i in words:
        categorie = []
        for syn in wn.synsets(i):
            if ('noun' in syn.lexname()) & ('Tops' not in syn.lexname()):
                categorie.append(syn.lexname())
            if 'verb' in syn.lexname():
                categorie.append(syn.lexname())
        categories.append(categorie)
    return categories

noun_cat_1 = categories_word(noun_book_1)
verb_cat_1 = categories_word(verb_book_1)
print(verb_cat_1)
```

Outputs after executing above code:-

```
[['verb.cognition', 'verb.stative', 'verb.stative', 'verb.stative', 'verb.communication'], ['verb.social', 'verb.change', 'verb.creation',
```

We obtain 2 Dimensional lists, each of which contains the categories under which every verb and noun in the WordNet Database has been categorized.

a. **To create a histogram, find the frequency of each category for each noun andverb**

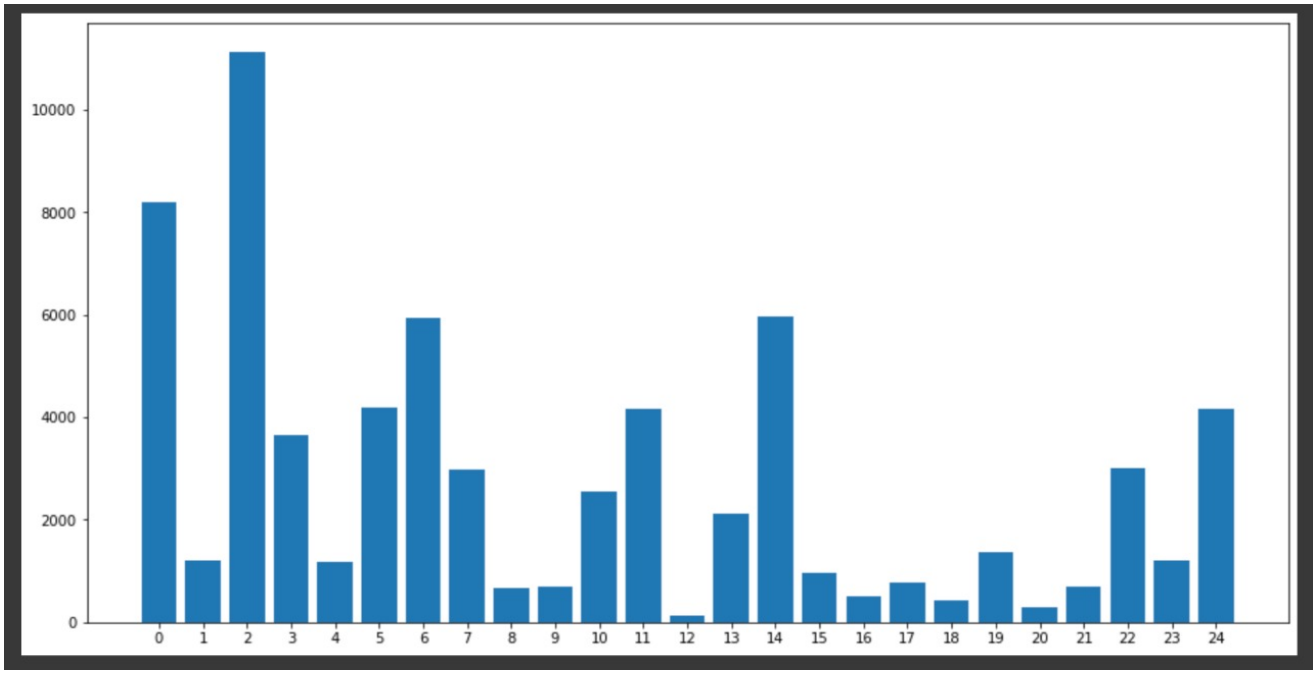
```
def synset(noun, verb):
    nouns = []
    verbs = []
    for word in noun:
        for syn in wn.synsets(word):
            if ('noun' in syn.lexname()) & ('Tops' not in syn.lexname()):
                nouns.append(syn.lexname())
            if 'verb' in syn.lexname():
                verbs.append(syn.lexname())
    for word in verb:
        for syn in wn.synsets(word):
            if ('noun' in syn.lexname()) & ('Tops' not in syn.lexname()):
                nouns.append(syn.lexname())
            if 'verb' in syn.lexname():
                verbs.append(syn.lexname())

    return nouns, verbs

noun_superset1, verb_superset1 = synset(noun_book_1, verb_book_1)
print(verb_superset1)
```

Output:-

```
['verb.change', 'verb.change', 'verb.motion', 'verb.stative', 'verb.creation', 'verb.creation', 'verb.motion', 'verb.motion', 'verb.motion'
```

Code for Histogram:-

```
import matplotlib.pyplot as plt
import numpy as np
labels, counts = np.unique(noun_superset1, return_counts=True)
ticks = range(len(counts))
plt.figure(figsize=(15, 8))
plt.bar(ticks, counts, align='center')
plt.xticks(ticks, range(len(counts)))
ticks = range(len(counts))
```



Task – 2

Recognise entity types in the text, measure the performance of the method used by random sampling of manually-tagged passages and present the accuracy with the help of F-score.

1. **Importing required libraries:**

```
import spacy
from spacy import displacy
from spacy import tokenizer
from spacy.scorer import Scorer
from spacy.tokens import Doc
from spacy.training import Example
```

Creating an evaluation function for entity tagging with SpaCy:

```
#evaluation function of the given examples
def calculate(para):
    scorer = Scorer()
    exp = []
    nlp = spacy.load('en_core_web_sm')  # for spaCy's pretrained use 'en_core_web_sm'
    for input_, annot in para:
        pred = nlp.make_doc(input_)
        temp = Example.from_dict(pred, annot)
        exp.append(temp)
    values = nlp.calculate(exp)
    return values
```

```
#loading the model
nlp = spacy.load('en_core_web_sm')
t1 = open("/content/drive/MyDrive/NLP/passage2.txt", encoding="utf8").read() #read data from file
txt=""
ex = []
for line in t1:
    txt = line
    doc = nlp(txt)
    var = (line[:len(line)-1], {'entities':[(e.start_char, e.end_char, e.label_) for e in doc.ents]}) #extracting entites from the line
    if len(doc.ents)>0:
        ex.append(var)

ans = calculate(para)    #evaluating the model
print(ex)
print("Precision {:0.4f}\tRecall {:0.4f}\tF-score {:0.4f}".format(ans['ents_p'], ans['ents_r'], ans['ents_f']))
```

1. Manually tagging random passages from the chosen book:

We picked random passages and tagged them manually. We then passed them through our evaluation function to calculate the F-score. Precision is a score that tells us what ratio of the positives identified by our algorithm was actually correct.

$$Precision = \frac{TP}{TP + FP}$$

- **TP: True positive; the positives identified by your algorithm are actually positives.**
- **FP: False positive; the positives identified by your algorithm that are actually negatives.**

Recall is a score that tells us what ratio of actual positives was identified as positive by our algorithm.

$$Recall = \frac{TP}{TP + FN}$$

- FN: False negatives; the negatives identified by your algorithm that are actually positives.

The F-score is the weighted harmonic mean of Precision and Recall. This means that this score is only high when both Precision and Recall are high, and low when either is low. The F-score helps the user get the true score of how well their algorithm is performing.

$$Fscore = \frac{\left(B^2 + 1\right)PR}{B^2P + R}$$

- B = Weight that defines the importance of Precision and Recall
- P = Precision
- R = Recall

PARAGRAPH-1:

The Thing ORG itself lay almost entirely buried in sand, amidst the scattered splinters of a fir tree it had shivered to fragments in its descent. The uncovered part had the appearance of a huge cylinder, caked over and its outline softened by a thick scaly dun-coloured incrustation. It had a diameter of about thirty yards QUANTITY . He approached the mass, surprised at the size and more so at the shape, since most meteorites are rounded more or less completely. It was, however, still so hot from its flight through the air as to forbid his near approach. A stirring noise within its cylinder he ascribed to the unequal cooling of its surface; for at that time it had not occurred to him that it might be hollow.

He remained standing at the edge of the pit that the Thing ORG had made for itself, staring at its strange appearance, astonished chiefly at its unusual shape and colour, and dimly perceiving even then some evidence of design in its arrival. The early morning TIME was wonderfully still, and the sun, just clearing the pine trees towards Weybridge ORG , was already warm. He did not remember hearing any birds that morning TIME , there was certainly no breeze stirring, and the only sounds were the faint movements from within the cindery cylinder. He was all alone on the common.

As we can see that there are many entities which are incorrectly classified such as:
- Weybridge: tagged as ORG while it should have been LOC

We thus entered the tags ourselves and calculated the scores which came out to be:
Precision: 0.7314 **Recall:** 0.6567 **F-score:** 0.5831

PARAGRAPH-2:

In spite of all that has happened since, I still remember that vigil very distinctly: the black and silent observatory, the shadowed lantern throwing a feeble glow upon the floor in the corner, the steady ticking of the clockwork of the telescope, the little slit in the roof—an oblong profundity with the stardust streaked across it. Ogilvy ORG moved about, invisible but audible. Looking through the telescope, one CARDINAL saw a circle of deep blue and the little round planet swimming in the field. It seemed such a little thing, so bright and small and still, faintly marked with transverse stripes, and slightly flattened from the perfect round. But so little it was, so silvery warm—a pin's head of light! It was as if it quivered, but really this was the telescope vibrating with the activity of the clockwork that kept the planet in view.

As I watched, the planet seemed to grow larger and smaller and to advance and recede, but that was simply that my eye was tired. Forty millions CARDINAL of miles it was from us— more than forty millions CARDINAL of miles of void. Few people realise the immensity of vacancy in which the dust of the material universe swims.

As we can see that there are many entities which are incorrectly classified such as:
Ogilvy tagged as ORG while it should have been PERSON

We thus entered the tags ourselves and calculated the scores which came out to be:
Precision: 0.8214 **Recall:** 0.7667 **F-score:** 0.7931

TASK-3:

The most important step in named entity identification is to find surface features that will help with relation classification. As the main source of information, the characteristics of the recognised entities themselves should be considered, specifically:

1. Named entity types of the two candidate arguments
2. Concatenation of the two entity types
3. Head words of the arguments
4. Bag of words from each of the arguments

There are several methods that can be used for extracting the relationships between entities from a book:

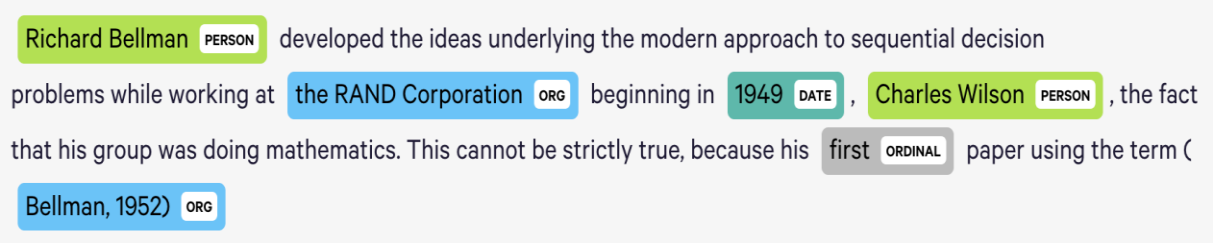
1. Named entity recognition: This method can be used to locate and categorise identified entities in a text, such as individuals, groups, places, and more. This can be helpful for locating the entities referenced in a book and determining the relationships between them.

2. Part-of-speech tagging: This method can be used to determine the word's part of speech in a text. This can be helpful for figuring out the functions that various words do in a sentence and comprehending the connections between them.

3. Dependency parsing: This method can be applied to examine a sentence's grammatical structure and determine the relationships between its words. Understanding the connections between entities in a text can be helped by doing this.

Extract the relationship between the entities from the book. Since we have already picked out and tagged theentities, the next step is to be able to deduce relationships between these entities using NLP.

Let's take the following example from our book itself :



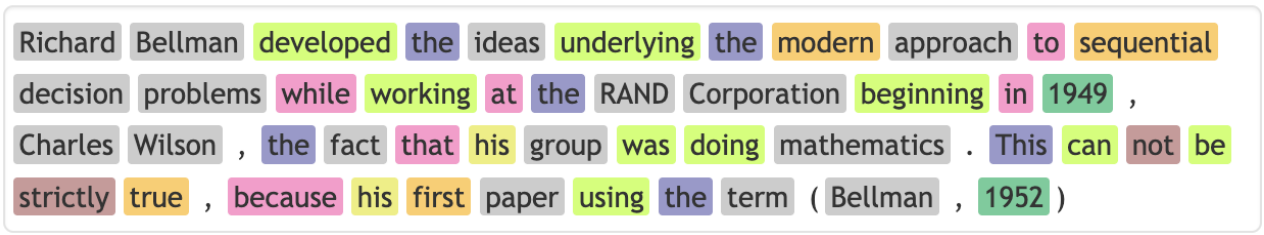
Here, we can see the following entities being recognised:

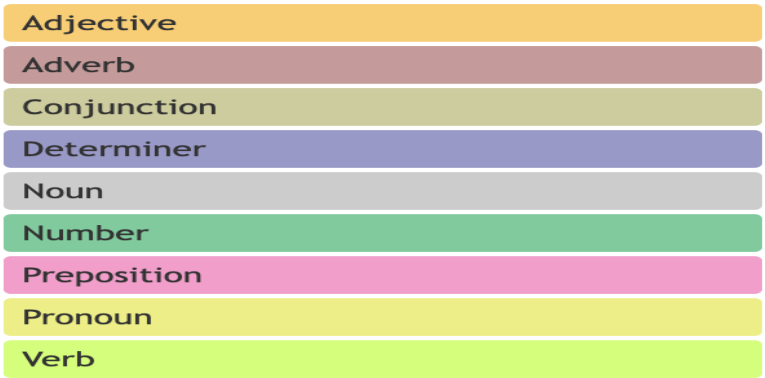
- ORG: the RAND Corporation, Bellman,1952
- PERSON: Richard Bellman, Charles Wilson.
- DATE: 1949.

We aim to derive relationships as such:

- decision problems (ORG: the RAND Corporation, PERSON: Richard Bellman)

If we look at the POS tags for the sentence:





As we see, the POS tags carry strong signals towards identifying the relational phrases. Thus, combining the information in the POS tags with those in the NER tags works well to identify the relational phrases:

- NOUN (problems) - PREPOSITION (while) - VERB (working) - PREPOSITION (at) – NOUN (Rand Corporation) => helps in identifying decision problems relations.

Conclusions

Through this project we are able to learn some very advanced concepts like Wordnet, how to use it, Named Entity Recognition. We learnt their practical applications, when and how they are used.

Github Link:- https://github.com/jYash2309/NLP_PROJECT_ROUND-1-2