NATURAL LANGUAGE PROCESSING

Textual Insight Engine

Uncovering Literary Patterns with Advanced NLP & Information Extraction

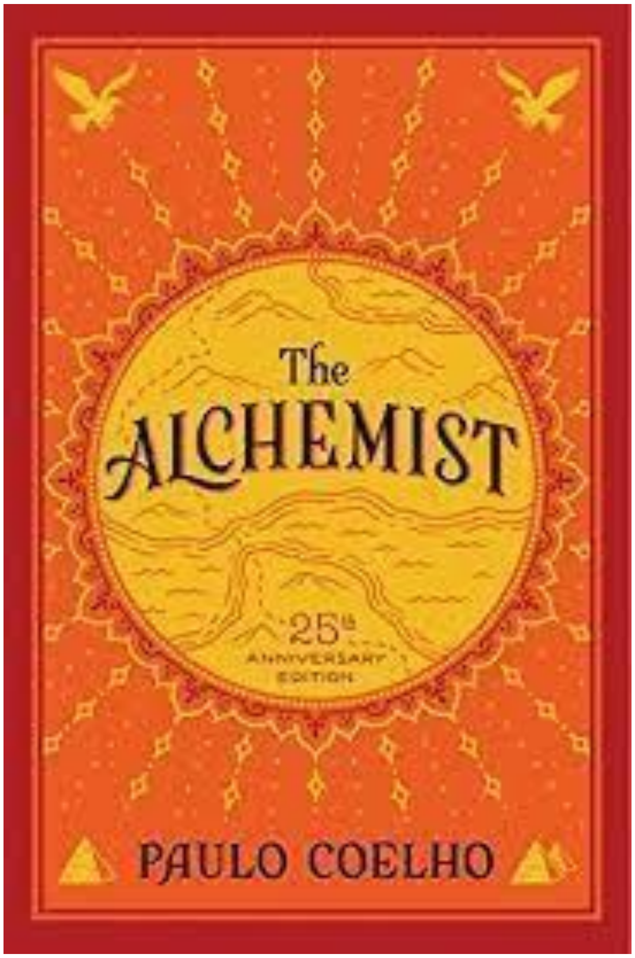
Book used for this project:

The Alchemist by Paulo Coelho

Overview

In this project, we will use different Natural Language Processing models to analyze a book. We will apply PoS tagging and create a Bi-gram table using the NLP frameworks defined in Python. The actual code is uploaded on GitHub but the screenshots of the output are uploaded using the code on Jupyter Notebook with some changes in the variable names.

Book Used



The Alchemist by Paulo Coelho

Github Link for the code and Implementation

https://github.com/SwastikDalal

Problem Statement

Performing the following steps on a book used for the project.

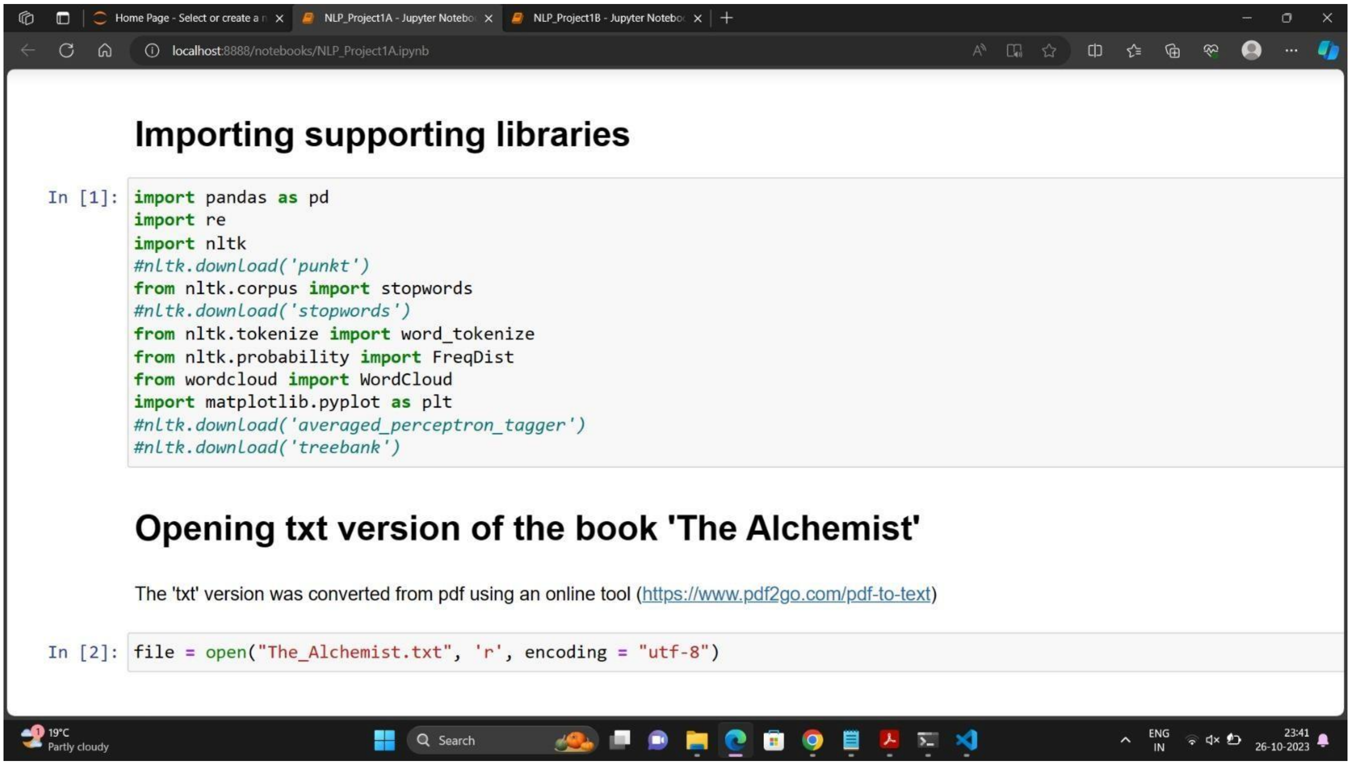
1. Import the text from the book in txt format.
2. Perform Pre-processing on the text.
3. Perform tokenization.
4. Create a frequency distribution table for the text.
5. Create a word cloud for the tokens.
6. Perform PoS tagging.
7. Create a bi-gram probability table for the largest chapter in the book.
8. Play the Shannon game using the bi-gram table.
9. Analyze the accuracy of the bi-gram model.

Data Description

The pdf book is first transformed into txt format. However, before we can use any of the Python NLP frameworks, we must first eliminate any irrelevant data from the converted text or data that has no impact on the frequency distribution table or the bi-gram model. The content includes photos, the name of the book that appears at the bottom-right of each page, page numbers, chapter names, and a watermark from the website where the book was downloaded. This data may have a negative impact on NLP frameworks and so must be eliminated. The actual application of Python begins in the following section.

Data Pre-Processing

Before starting data pre-processing, we will first import the book.


Home Page - Select or create a n
NLP_Project1A - Jupyter Notebo
NLP_Project1B - Jupyter Notebo
localhost:8888/notebooks/NLP_Project1A.ipynb
19°C
Partly cloudy
Search
ENG
IN
23:41
26-10-2023

Importing supporting libraries

In [1]:

```
import pandas as pd
import re
import nltk
#nltk.download('punkt')
from nltk.corpus import stopwords
#nltk.download('stopwords')
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from wordcloud import WordCloud
import matplotlib.pyplot as plt
#nltk.download('averaged_perceptron_tagger')
#nltk.download('treebank')
```

Opening txt version of the book 'The Alchemist'

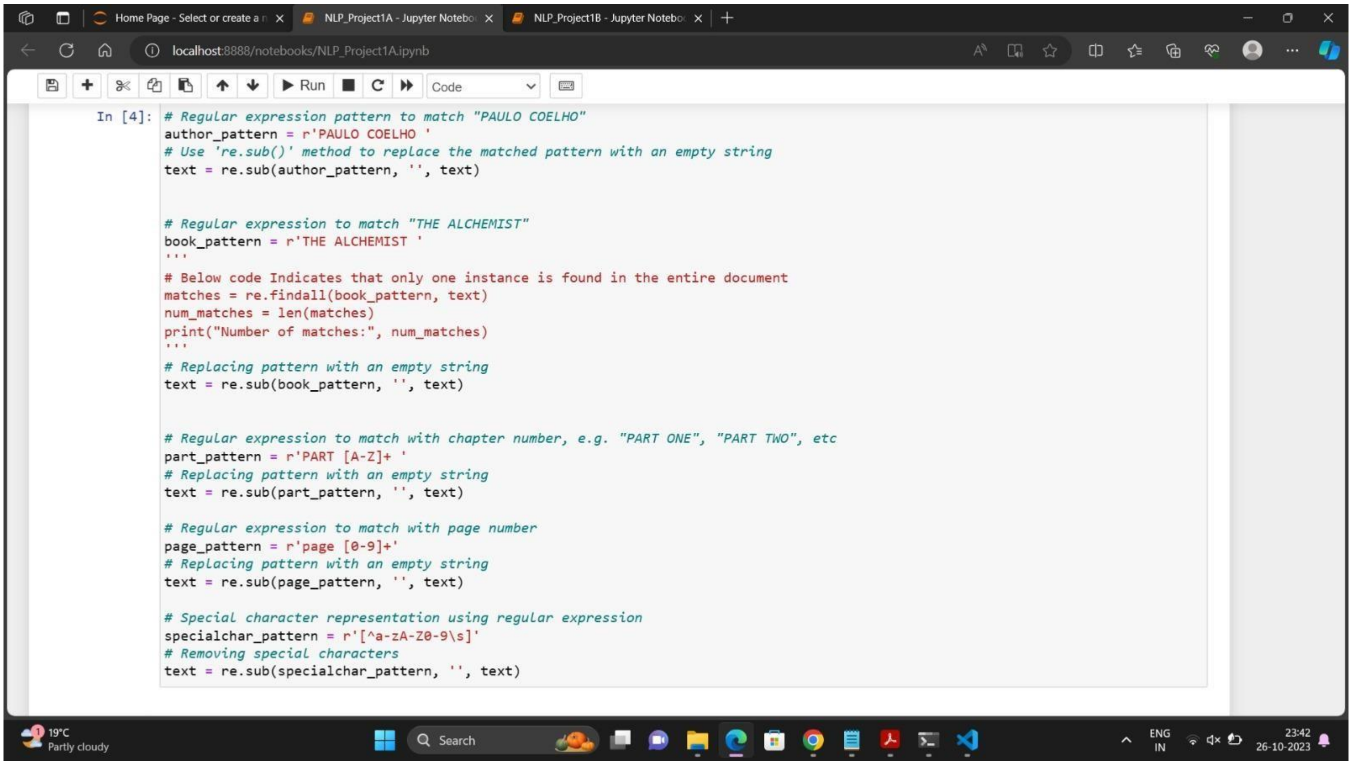
The 'txt' version was converted from pdf using an online tool (https://www.pdf2go.com/pdf-to-text)

In [2]:

```
file = open("The_Alchemist.txt", 'r', encoding = "utf-8")
```

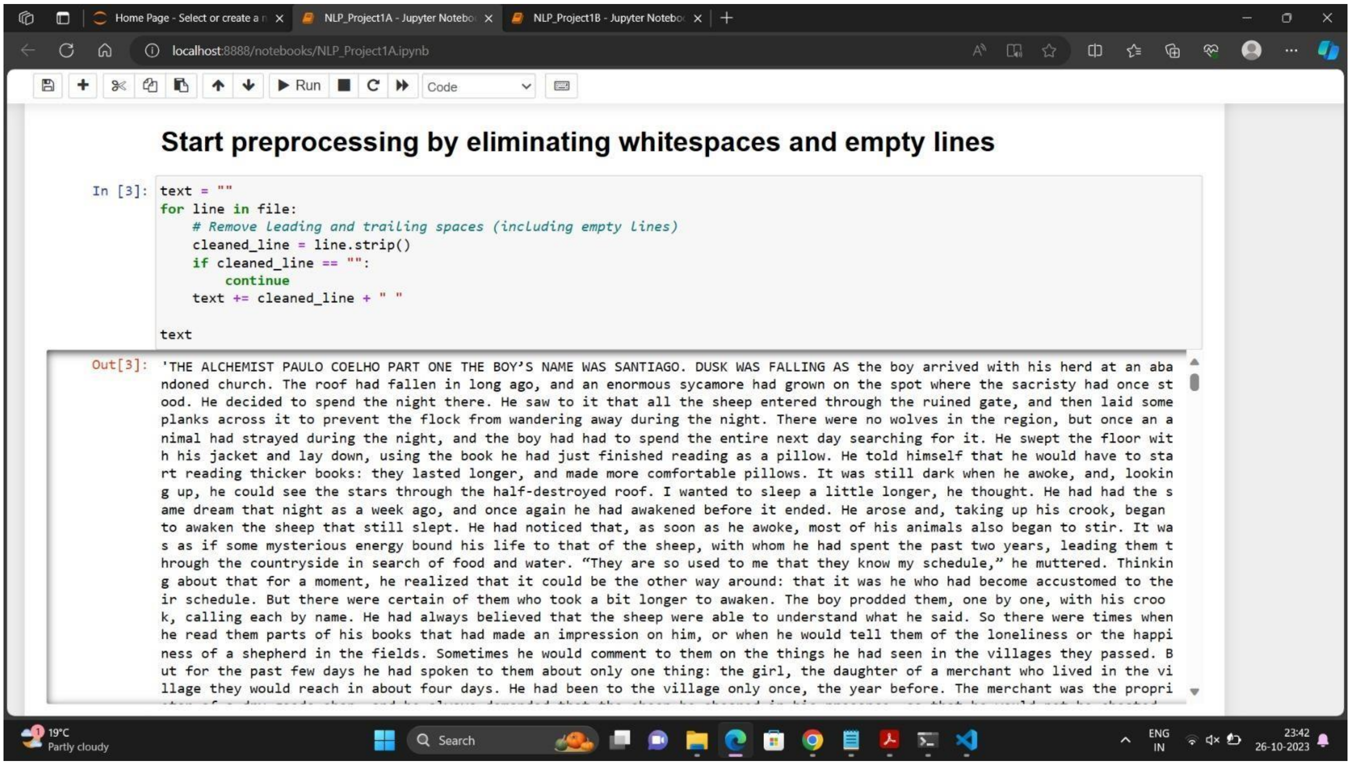
Data preprocessing is the process of cleaning and organizing the unstructured text data to prepare it for analysis. We will remove all the irrelevant text data as described in the previous section with the help of Regular Expressions.

1. Remove the name of the book.
2. Remove author name.
3. Remove chapter names.
4. Remove page numbers.
5. Remove special characters.


Home Page - Select or create a n
NLP_Project1A - Jupyter Notebo
NLP_Project1B - Jupyter Notebo
localhost:8888/notebooks/NLP_Project1A.ipynb
Run
Code
19°C
Partly cloudy
Search
ENG
IN
23:42
26-10-2023

```
In [4]: # Regular expression pattern to match "PAULO COELHO"
        author_pattern = r'PAULO COELHO '
        # Use 're.sub()' method to replace the matched pattern with an empty string
        text = re.sub(author_pattern, '', text)


        # Regular expression to match "THE ALCHEMIST"
        book_pattern = r'THE ALCHEMIST '
        '''
        # Below code Indicates that only one instance is found in the entire document
        matches = re.findall(book_pattern, text)
        num_matches = len(matches)
        print("Number of matches:", num_matches)
        '''
        # Replacing pattern with an empty string
        text = re.sub(book_pattern, '', text)


        # Regular expression to match with chapter number, e.g. "PART ONE", "PART TWO", etc
        part_pattern = r'PART [A-Z]+ '
        # Replacing pattern with an empty string
        text = re.sub(part_pattern, '', text)

        # Regular expression to match with page number
        page_pattern = r'page [0-9]+'
        # Replacing pattern with an empty string
        text = re.sub(page_pattern, '', text)

        # Special character representation using regular expression
        specialchar_pattern = r'[^a-zA-Z0-9\s]'
        # Removing special characters
        text = re.sub(specialchar_pattern, '', text)
```

6. Remove extra whitespace, tabs and newlines.

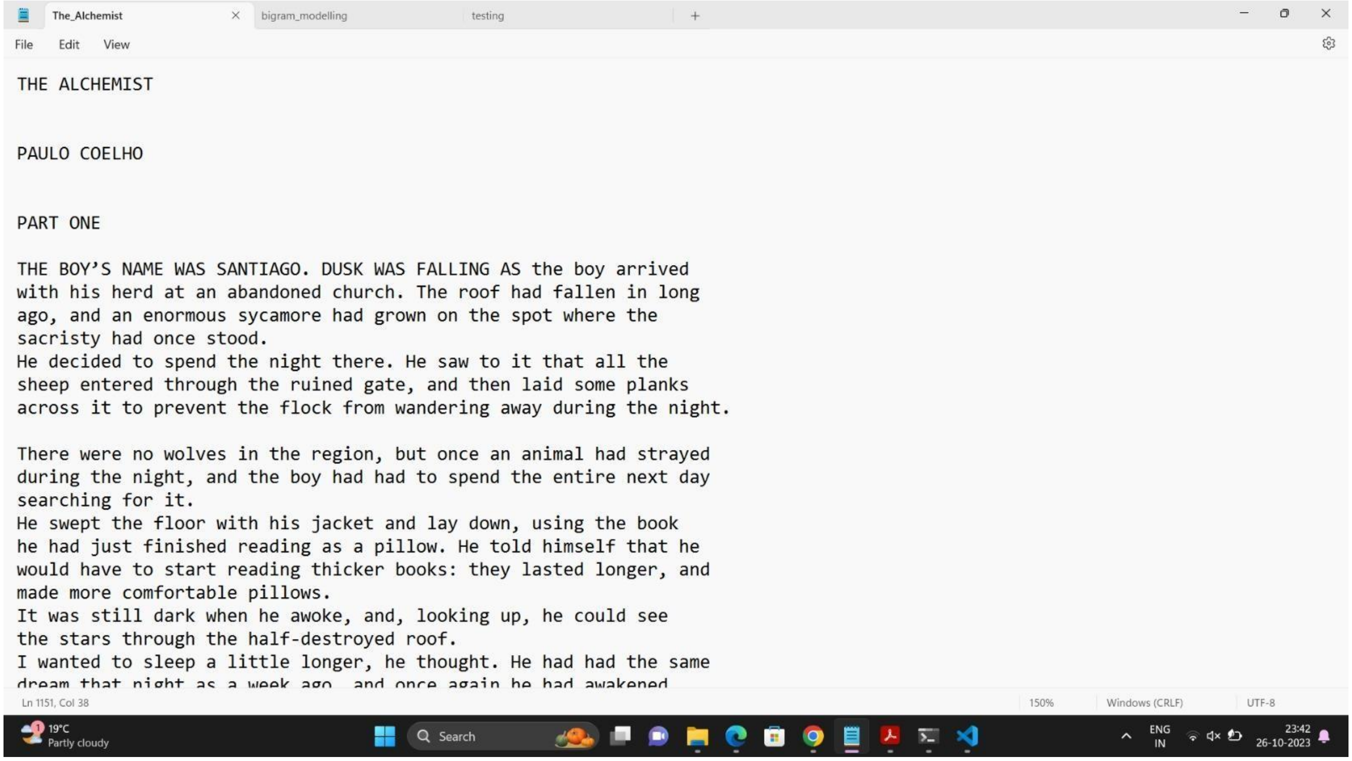

Home Page - Select or create a n
NLP_Project1A - Jupyter Notebo
NLP_Project1B - Jupyter Notebo
localhost:8888/notebooks/NLP_Project1A.ipynb
Run
Code
19°C
Partly cloudy
Search
ENG
IN
23:42
26-10-2023

Start preprocessing by eliminating whitespaces and empty lines

```
In [3]: text = ""
        for line in file:
            # Remove leading and trailing spaces (including empty lines)
            cleaned_line = line.strip()
            if cleaned_line == "":
                continue
            text += cleaned_line + " "

        text
```

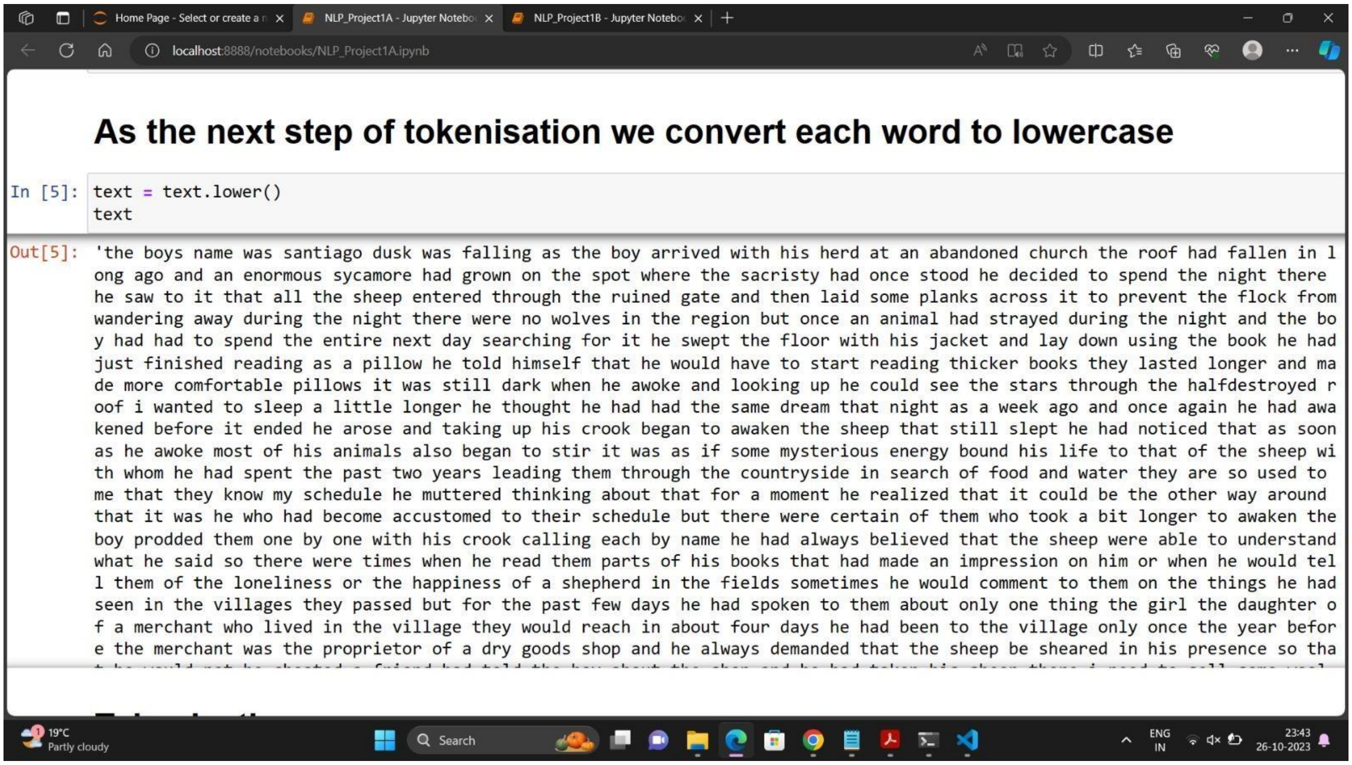
```
Out[3]: 'THE ALCHEMIST PAULO COELHO PART ONE THE BOY’S NAME WAS SANTIAGO. DUSK WAS FALLING AS the boy arrived with his herd at an aba
        ndoned church. The roof had fallen in long ago, and an enormous sycamore had grown on the spot where the sacristy had once st
        ood. He decided to spend the night there. He saw to it that all the sheep entered through the ruined gate, and then laid some
        planks across it to prevent the flock from wandering away during the night. There were no wolves in the region, but once an a
        nimal had strayed during the night, and the boy had had to spend the entire next day searching for it. He swept the floor wit
        h his jacket and lay down, using the book he had just finished reading as a pillow. He told himself that he would have to sta
        rt reading thicker books: they lasted longer, and made more comfortable pillows. It was still dark when he awoke, and, lookin
        g up, he could see the stars through the half-destroyed roof. I wanted to sleep a little longer, he thought. He had had the s
        ame dream that night as a week ago, and once again he had awakened before it ended. He arose and, taking up his crook, began
        to awaken the sheep that still slept. He had noticed that, as soon as he awoke, most of his animals also began to stir. It wa
        s as if some mysterious energy bound his life to that of the sheep, with whom he had spent the past two years, leading them t
        hrough the countryside in search of food and water. “They are so used to me that they know my schedule,” he muttered. Thinkin
        g about that for a moment, he realized that it could be the other way around: that it was he who had become accustomed to the
        ir schedule. But there were certain of them who took a bit longer to awaken. The boy prodded them, one by one, with his croo
        k, calling each by name. He had always believed that the sheep were able to understand what he said. So there were times when
        he read them parts of his books that had made an impression on him, or when he would tell them of the loneliness or the happi
        ness of a shepherd in the fields. Sometimes he would comment to them on the things he had seen in the villages they passed. B
        ut for the past few days he had spoken to them about only one thing: the girl, the daughter of a merchant who lived in the vi
        llage they would reach in about four days. He had been to the village only once, the year before. The merchant was the propri
```

Unstructured Data



```
THE ALCHEMIST


PAULO COELHO


PART ONE

THE BOY’S NAME WAS SANTIAGO. DUSK WAS FALLING AS the boy arrived
with his herd at an abandoned church. The roof had fallen in long
ago, and an enormous sycamore had grown on the spot where the
sacristy had once stood.
He decided to spend the night there. He saw to it that all the
sheep entered through the ruined gate, and then laid some planks
across it to prevent the flock from wandering away during the night.

There were no wolves in the region, but once an animal had strayed
during the night, and the boy had had to spend the entire next day
searching for it.
He swept the floor with his jacket and lay down, using the book
he had just finished reading as a pillow. He told himself that he
would have to start reading thicker books: they lasted longer, and
made more comfortable pillows.
It was still dark when he awoke, and, looking up, he could see
the stars through the half-destroyed roof.
I wanted to sleep a little longer, he thought. He had had the same
dream that night as a week ago, and once again he had awakened
```

Output:


Home Page - Select or create a n
NLP_Project1A - Jupyter Notebo
NLP_Project1B - Jupyter Notebo
localhost:8888/notebooks/NLP_Project1A.ipynb
19°C
Partly cloudy
Search
ENG
IN
23:43
26-10-2023

As the next step of tokenisation we convert each word to lowercase

```
In [5]: text = text.lower()
        text
```

```
Out[5]: 'the boys name was santiago dusk was falling as the boy arrived with his herd at an abandoned church the roof had fallen in l
        ong ago and an enormous sycamore had grown on the spot where the sacristy had once stood he decided to spend the night there
        he saw to it that all the sheep entered through the ruined gate and then laid some planks across it to prevent the flock from
        wandering away during the night there were no wolves in the region but once an animal had strayed during the night and the bo
        y had had to spend the entire next day searching for it he swept the floor with his jacket and lay down using the book he had
        just finished reading as a pillow he told himself that he would have to start reading thicker books they lasted longer and ma
        de more comfortable pillows it was still dark when he awoke and looking up he could see the stars through the halfdestroyed r
        oof i wanted to sleep a little longer he thought he had had the same dream that night as a week ago and once again he had awa
        kened before it ended he arose and taking up his crook began to awaken the sheep that still slept he had noticed that as soon
        as he awoke most of his animals also began to stir it was as if some mysterious energy bound his life to that of the sheep wi
        th whom he had spent the past two years leading them through the countryside in search of food and water they are so used to
        me that they know my schedule he muttered thinking about that for a moment he realized that it could be the other way around
        that it was he who had become accustomed to their schedule but there were certain of them who took a bit longer to awaken the
        boy prodded them one by one with his crook calling each by name he had always believed that the sheep were able to understand
        what he said so there were times when he read them parts of his books that had made an impression on him or when he would tel
        l them of the loneliness or the happiness of a shepherd in the fields sometimes he would comment to them on the things he had
        seen in the villages they passed but for the past few days he had spoken to them about only one thing the girl the daughter o
        f a merchant who lived in the village they would reach in about four days he had been to the village only once the year befor
        e the merchant was the proprietor of a dry goods shop and he always demanded that the sheep be sheared in his presence so tha
```

The changed text lacks numbers, special characters (as seen by the fact that the entire text is on the same line), watermarks, and so on.

This concludes the project's data pre-processing phase. The following part depicts the data preparation procedure, which includes text tokenization and stop word removal.

Data Preparation

This consists of two steps:

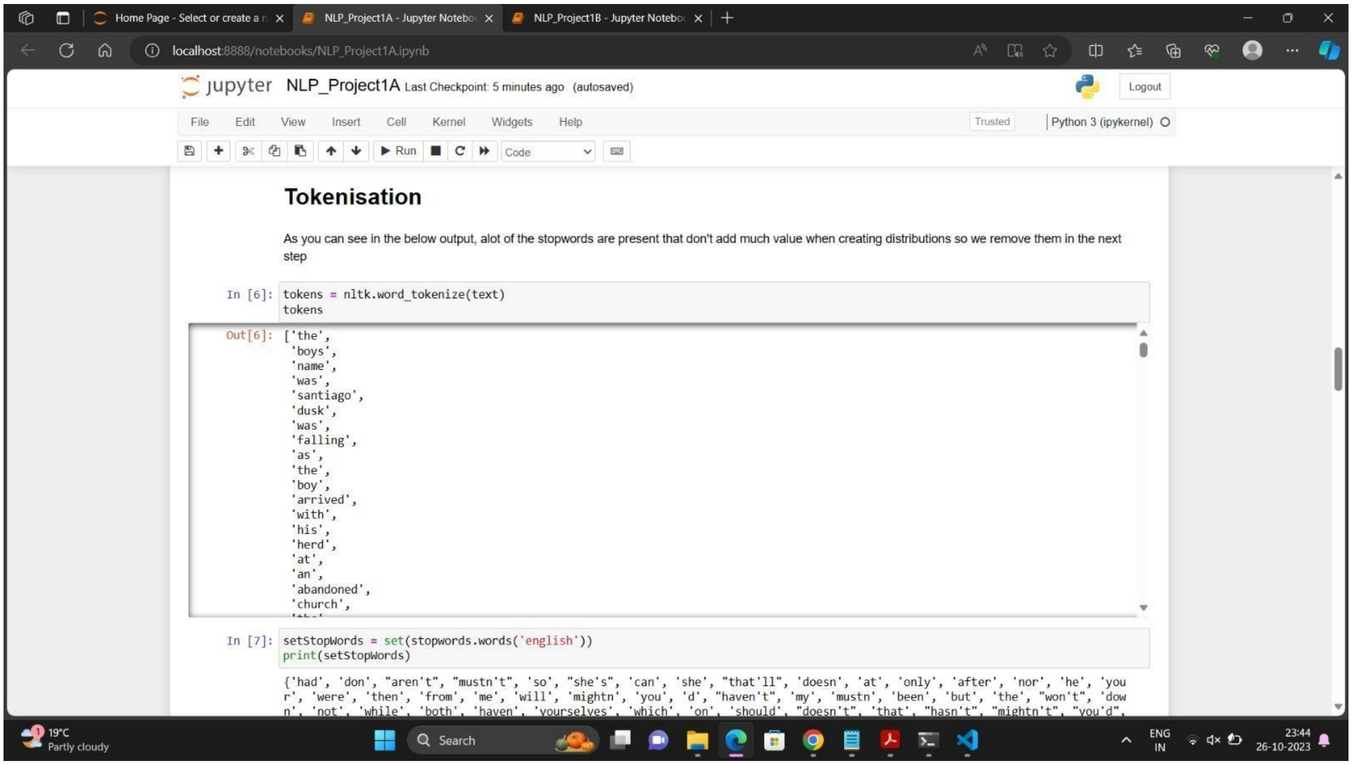
Tokenization

Removal of stop words

These steps are done using the built-in functions in the NLTK library in Python.

1) Tokenization: It is the process of breaking a stream of textual data into words, terms, sentences, symbols or other meaningful elements called tokens.

2) Removal of stopwords: These are the most common words in any natural language. Since they do not add much value to the document they must be removed before creating a word cloud or a frequency distribution table.


Home Page - Select or create a n
NLP_Project1A - Jupyter Notebo
NLP_Project1B - Jupyter Notebo
localhost:8888/notebooks/NLP_Project1A.ipynb
jupyter NLP_Project1A Last Checkpoint: 5 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
19°C
Partly cloudy
Search
ENG
IN
23:44
26-10-2023

Tokenisation

As you can see in the below output, alot of the stopwords are present that don't add much value when creating distributions so we remove them in the next step

```
In [6]: tokens = nltk.word_tokenize(text)
        tokens
```

```
Out[6]: ['the',
         'boys',
         'name',
         'was',
         'santiago',
         'dusk',
         'was',
         'falling',
         'as',
         'the',
         'boy',
         'arrived',
         'with',
         'his',
         'herd',
         'at',
         'an',
         'abandoned',
         'church',
```

```
In [7]: setStopWords = set(stopwords.words('english'))
        print(setStopWords)
```

```
{'had', 'don', "aren't", "mustn't", 'so', "she's", 'can', 'she', "that'll", 'doesn', 'at', 'only', 'after', 'nor', 'he', 'you
r', 'were', 'then', 'from', 'me', 'will', 'mightn', 'you', 'd', "haven't", 'my', 'mustn', 'been', 'but', 'the', "won't", 'dow
n', 'not', 'while', 'both', 'haven', 'yourselves', 'which', 'on', 'should', "doesn't", 'that', "hasn't", "mightn't", "you'd",
```

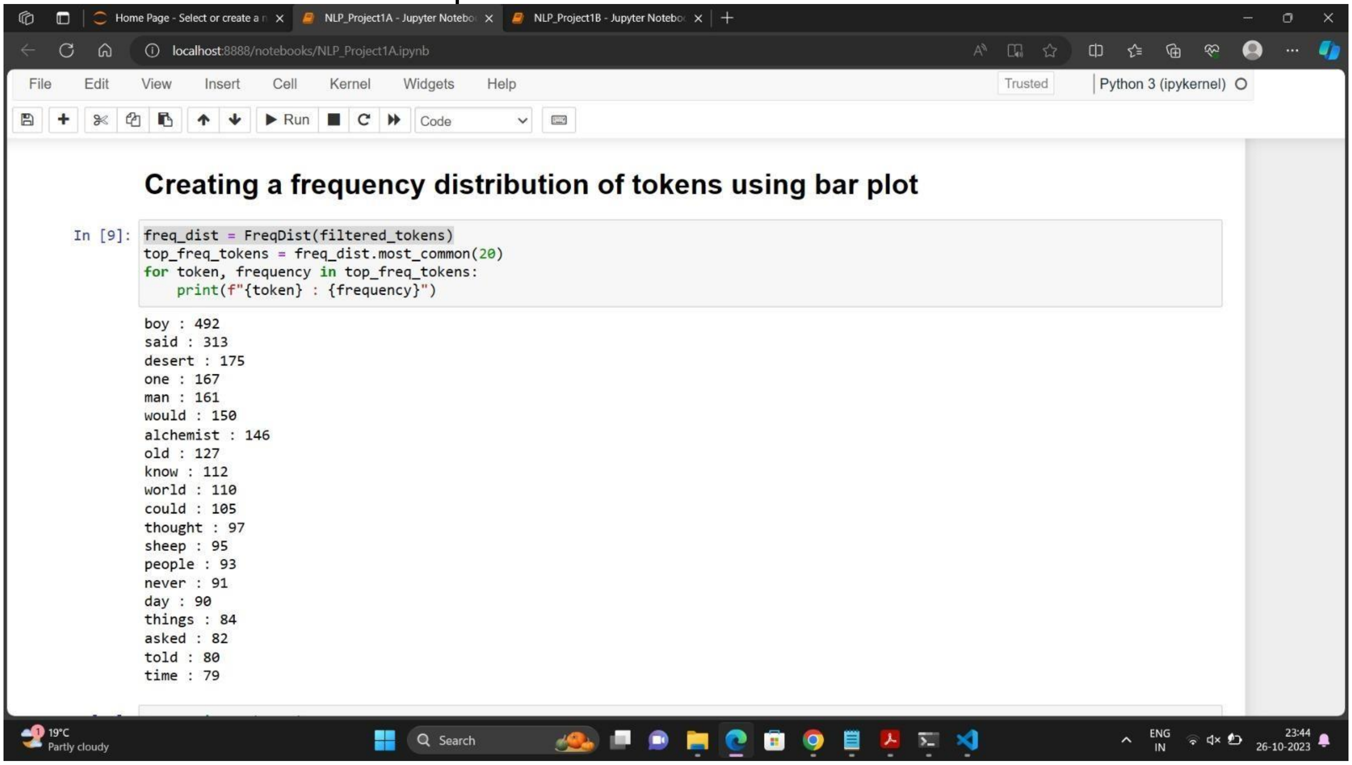
It can be observed that the stopwords like ‘is’,’are’ and so on are removed from the textual data. Now we can create the frequency distribution table.

Frequency Distribution Table

This table gives the frequency of each of the tokens generated from the textual data. After tokenization, we create a frequency distribution table directly using a built-in function called FreqDist(tokens) in the NLTK library.

We will only store the 20 most common words (tokens), that is, the first20 tokens when the frequencies of the tokens are arranged in descending order.

Some tokens and their frequencies:



Home Page - Select or create a n
NLP_Project1A - Jupyter Notebo
NLP_Project1B - Jupyter Notebo
localhost:8888/notebooks/NLP_Project1A.ipynb
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code
19°C
Partly cloudy
Search
ENG
IN
23:44
26-10-2023

Creating a frequency distribution of tokens using bar plot

In [9]:

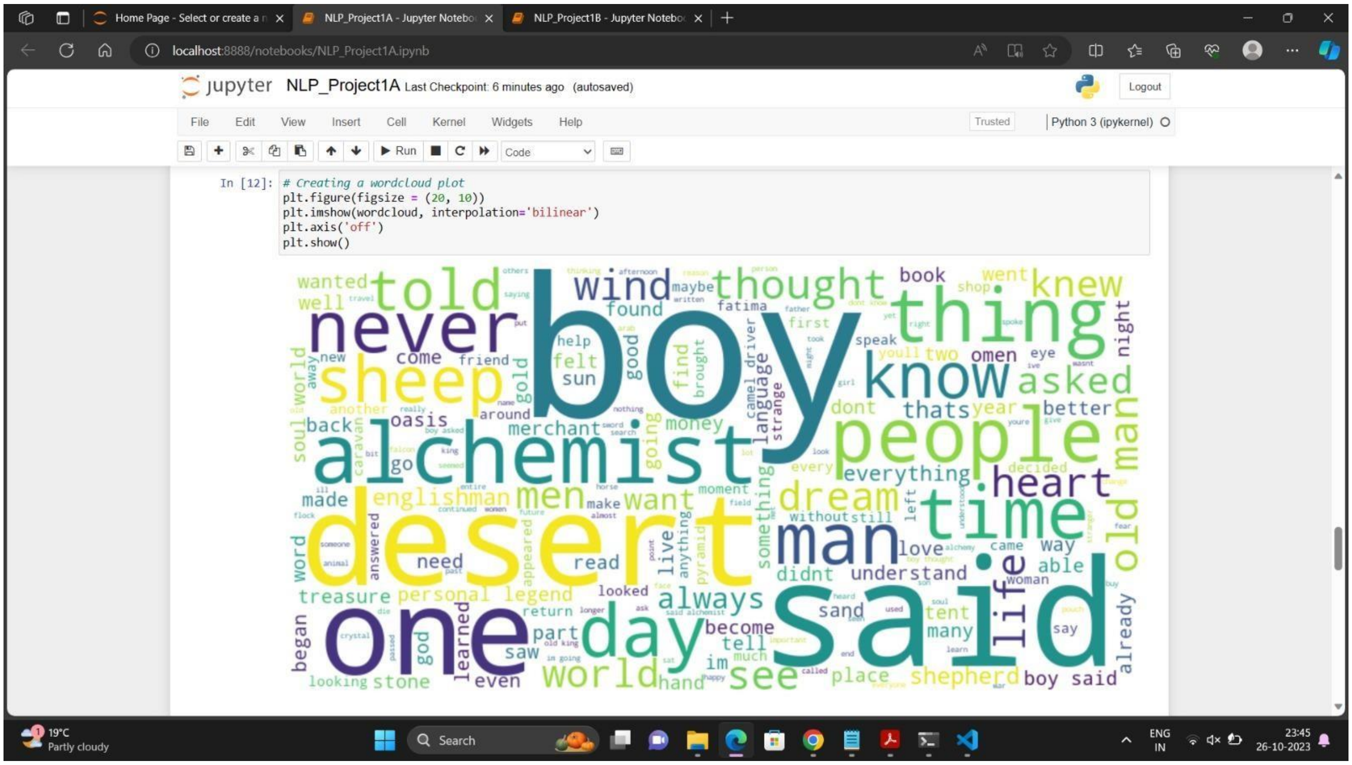
```
freq_dist = FreqDist(filtered_tokens)
top_freq_tokens = freq_dist.most_common(20)
for token, frequency in top_freq_tokens:
    print(f"{token} : {frequency}")
```

```
boy : 492
said : 313
desert : 175
one : 167
man : 161
would : 150
alchemist : 146
old : 127
know : 112
world : 110
could : 105
thought : 97
sheep : 95
people : 93
never : 91
day : 90
things : 84
asked : 82
told : 80
time : 79
```

The token ‘boy’ has the highest frequency which is logical because he is the protagonist of the book.

The frequency distribution table is stored in a non-graphical format and hence must be converted into a more visualizable format. One option is the Word Cloud. It can be created by using WordCloud function using the wordcloud and matplotlib library.

Word Cloud:

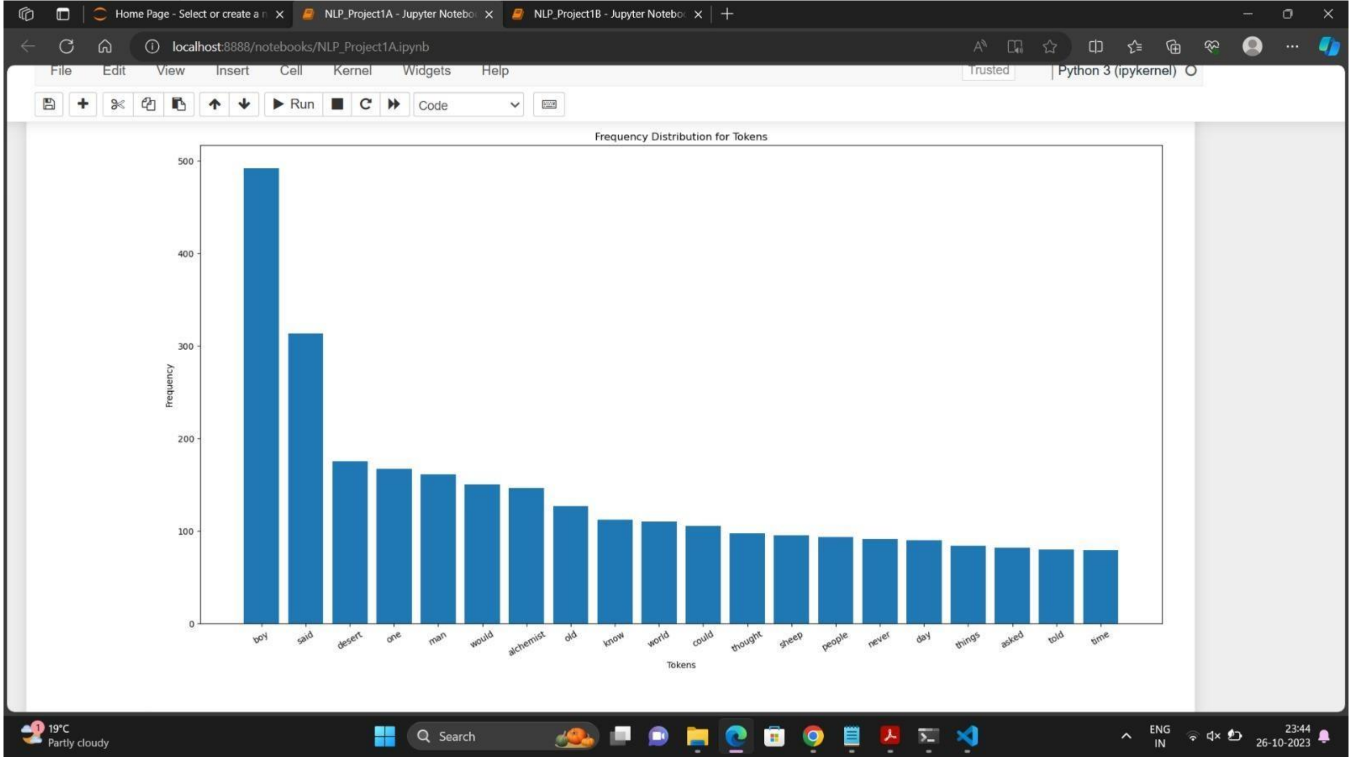

Home Page - Select or create a n
NLP_Project1A - Jupyter Notebo
NLP_Project1B - Jupyter Notebo
localhost:8888/notebooks/NLP_Project1A.ipynb
jupyter NLP_Project1A Last Checkpoint: 6 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
boy
said
desert
alchemist
one
people
time
thing
never
sheep
know
man
told
wind
thought
knew
day
life
dream
heart
world
see
19°C
Partly cloudy
Search
ENG
IN
23:45
26-10-2023

```
In [12]: # Creating a wordcloud plot
         plt.figure(figsize = (20, 10))
         plt.imshow(wordcloud, interpolation='bilinear')
         plt.axis('off')
         plt.show()
```

The words with the larger size correspond to larger frequency in the frequency distribution table.

Another option is to create a plot between the words and their frequencies. In this project, we will create a histogram using the ‘matplotlib’ library.

The histogram:

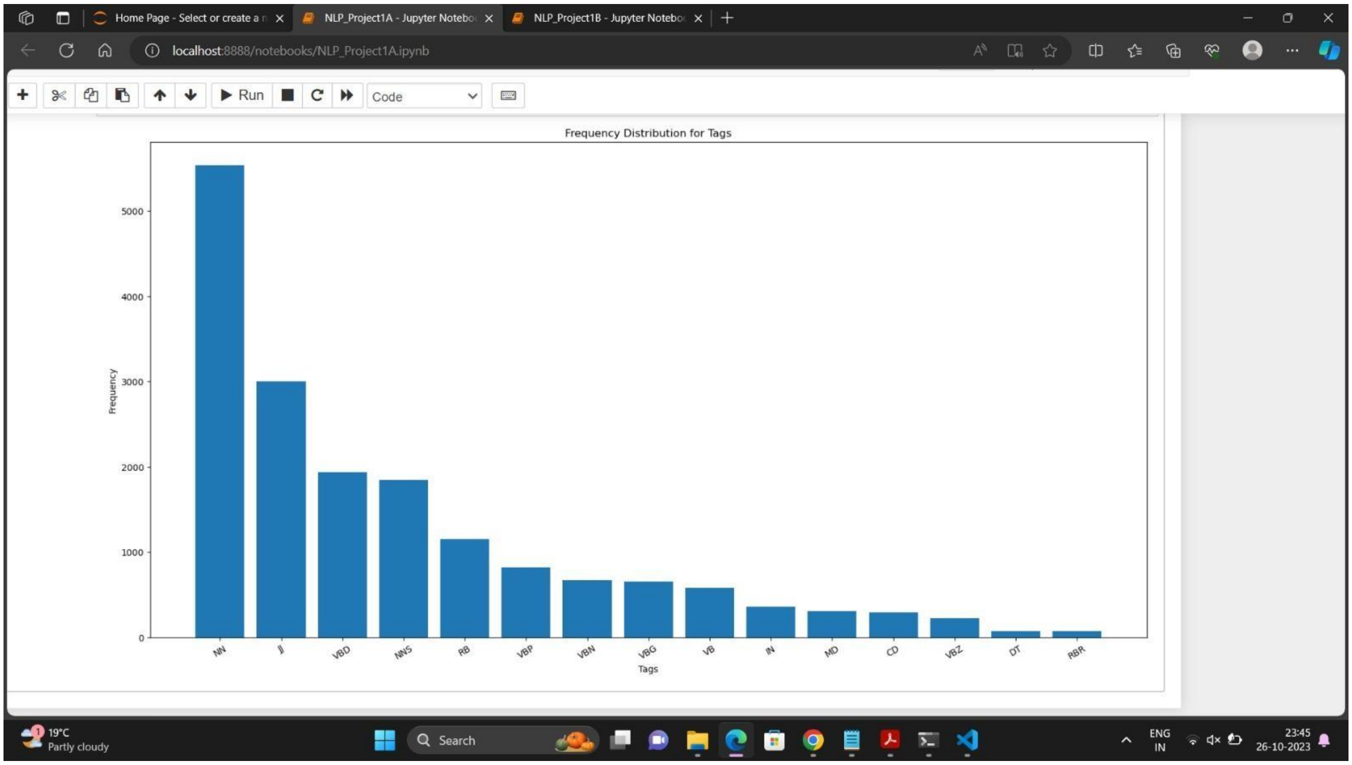


Part of Speech Tagging

It is a process of assigning each word in the textual data to its corresponding equivalent in part of speech. This includes Nouns, Verbs, Adjectives etc.

This is done using a tag set. In this project, we will use the Penn Treebank Corpus which has 36 PoS tags.

Distribution of some of the tags:


Home Page - Select or create a n
NLP_Project1A - Jupyter Notebo
NLP_Project1B - Jupyter Notebo
localhost:8888/notebooks/NLP_Project1A.ipynb
Run
Code
Frequency Distribution for Tags
Frequency
0
1000
2000
3000
4000
5000
NN
JJ
VBD
NNS
RB
VBP
VBN
VBG
VB
IN
MD
CD
VBZ
DT
RBR
Tags
19°C
Partly cloudy
Search
ENG
IN
23:45
26-10-2023

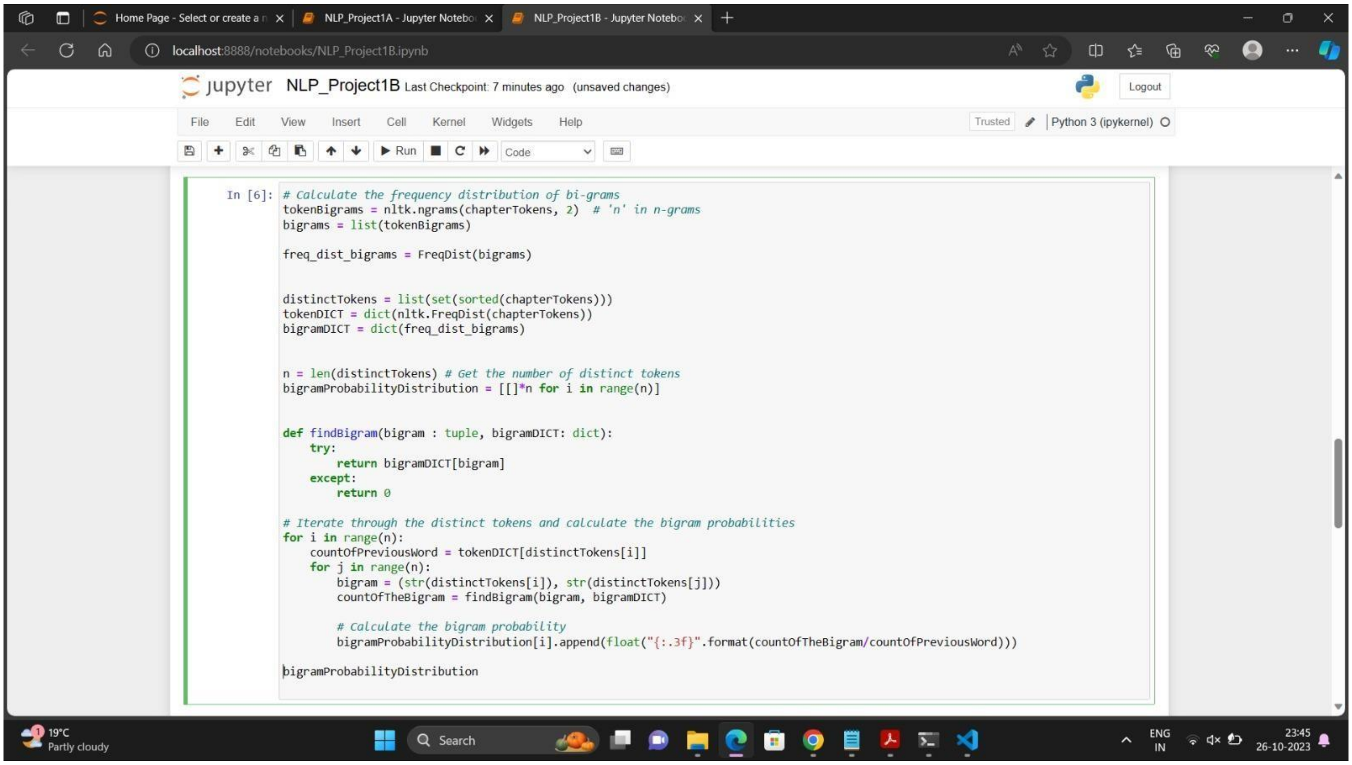
It can be observed that most tokens are tagged as nouns (NN) followed by adjectives (JJ) and so on.

Bi-gram Model

In this project, we will calculate the bigram probabilities of the words in the largest chapter of the book. The bigram model assumes that the probability of a word appearing in a sentence depends only upon the previous word.
To create the bi-gram table, we will first import the largest chapter, pre-process the textual data, tokenize it, and then create the bigram table.fFirst

we Import the largest chapter.

Pre-Processing the data and Tokenizing it:



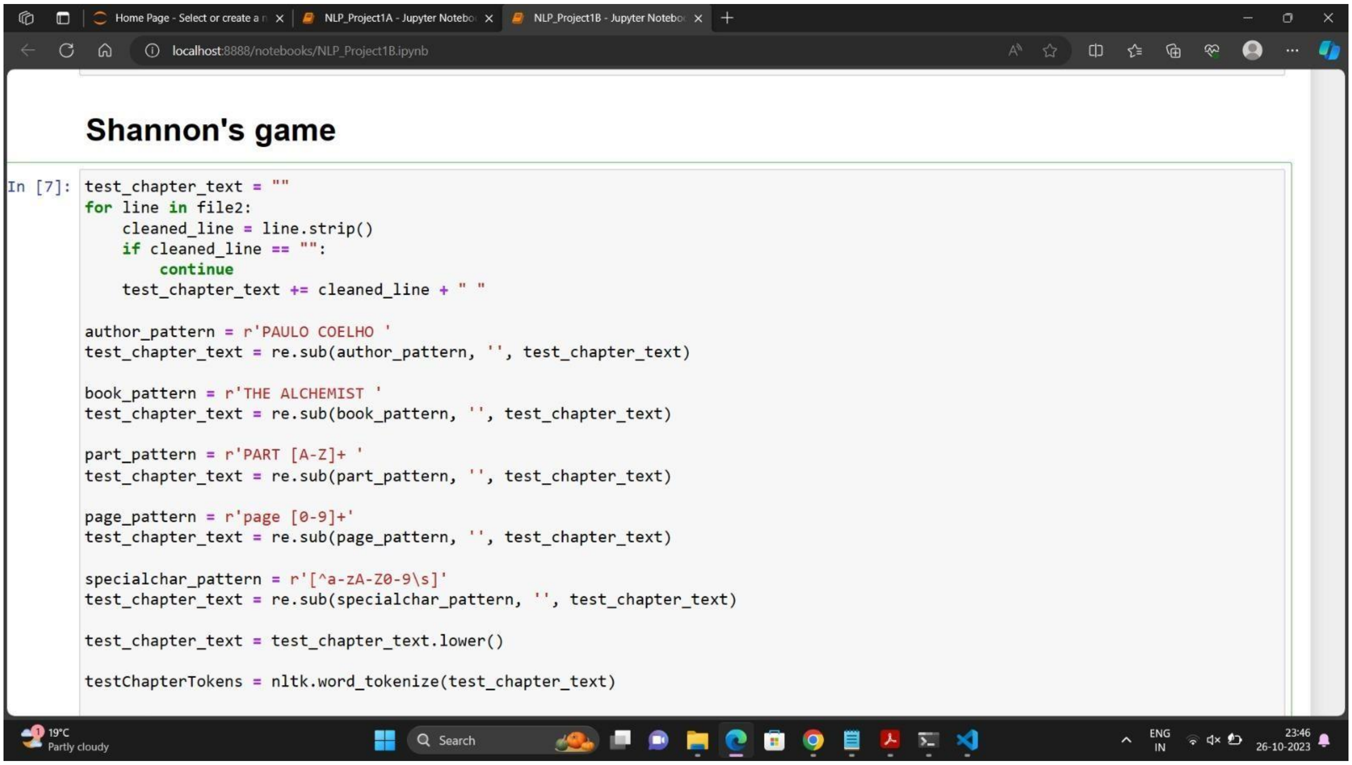
Home Page - Select or create a n
NLP_Project1A - Jupyter Notebo
NLP_Project1B - Jupyter Notebo
localhost:8888/notebooks/NLP_Project1B.ipynb
jupyter NLP_Project1B Last Checkpoint: 7 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
19°C
Partly cloudy
Search
ENG
IN
23:45
26-10-2023

```
In [6]: # Calculate the frequency distribution of bi-grams
tokenBigrams = nltk.ngrams(chapterTokens, 2)  # 'n' in n-grams
bigrams = list(tokenBigrams)

freq_dist_bigrams = FreqDist(bigrams)


distinctTokens = list(set(sorted(chapterTokens)))
tokenDICT = dict(nltk.FreqDist(chapterTokens))
bigramDICT = dict(freq_dist_bigrams)


n = len(distinctTokens) # Get the number of distinct tokens
bigramProbabilityDistribution = [[]*n for i in range(n)]


def findBigram(bigram : tuple, bigramDICT: dict):
    try:
        return bigramDICT[bigram]
    except:
        return 0

# Iterate through the distinct tokens and calculate the bigram probabilities
for i in range(n):
    countOfPreviousWord = tokenDICT[distinctTokens[i]]
    for j in range(n):
        bigram = (str(distinctTokens[i]), str(distinctTokens[j]))
        countOfTheBigram = findBigram(bigram, bigramDICT)

        # Calculate the bigram probability
        bigramProbabilityDistribution[i].append(float("{:.3f}".format(countOfTheBigram/countOfPreviousWord)))

bigramProbabilityDistribution
```

Shannon Game

This game is used to test the accuracy of a word guessing algorithm. Here, using the bi-gram model based upon the largest chapter in the book, we fill in the blanks in another chapter chosen and then calculate the accuracy by comparing the guessed word with the actual word in the chapter.

We first perform the basic steps (importing, pre-processing and tokenization) on a chosen chapter of the book.


Home Page - Select or create a n
NLP_Project1A - Jupyter Notebo
NLP_Project1B - Jupyter Notebo
localhost:8888/notebooks/NLP_Project1B.ipynb
19°C
Partly cloudy
Search
ENG
IN
23:46
26-10-2023

Shannon's game

In [7]:

```
test_chapter_text = ""
for line in file2:
    cleaned_line = line.strip()
    if cleaned_line == "":
        continue
    test_chapter_text += cleaned_line + " "

author_pattern = r'PAULO COELHO '
test_chapter_text = re.sub(author_pattern, '', test_chapter_text)

book_pattern = r'THE ALCHEMIST '
test_chapter_text = re.sub(book_pattern, '', test_chapter_text)

part_pattern = r'PART [A-Z]+ '
test_chapter_text = re.sub(part_pattern, '', test_chapter_text)

page_pattern = r'page [0-9]+'
test_chapter_text = re.sub(page_pattern, '', test_chapter_text)

specialchar_pattern = r'[^a-zA-Z0-9\s]'
test_chapter_text = re.sub(specialchar_pattern, '', test_chapter_text)

test_chapter_text = test_chapter_text.lower()

testChapterTokens = nltk.word_tokenize(test_chapter_text)
```

Now we create ‘n’ blanks in the text by replacing the tokens created (in the original text) with blanks. Here, we will take n=300. We will then play the Shanon game.

The blanks are created and then they are filled using the bi-gram probabilities table created using the largest chapter in the book.

Now, we will calculate the accuracy of our model.

The accuracy of the model comes out to be 11.66.

In the project's second phase, various Natural Language Processing techniques will be employed to analyze the book chosen for this project (The Alchemist). The subsequent step involves executing 'Entity Recognition' utilizing Python's NLP frameworks. The code is uploaded on GitHub and the screenshots are from the code on Jupyter Notebook and GitHub.

Problem Statement

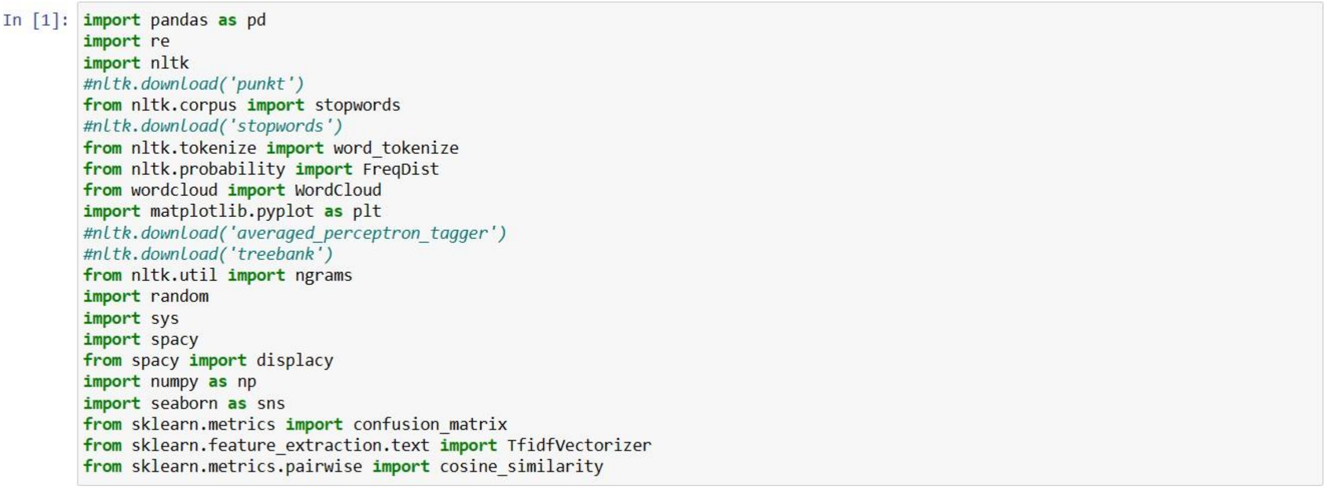
We aim to perform the following tasks:

1. Recognize all entities in the book and recognize all entity types.
2. Use different performance measures to evaluate the method used. We will take random passages from the book for this.
3. Repeat the above step three times.
4. Generate a TF-IDF vector for all the chapters in the book.
5. Find similarity between the chapters using similarity measure.
6. Create a gradient table.

Libraries Used

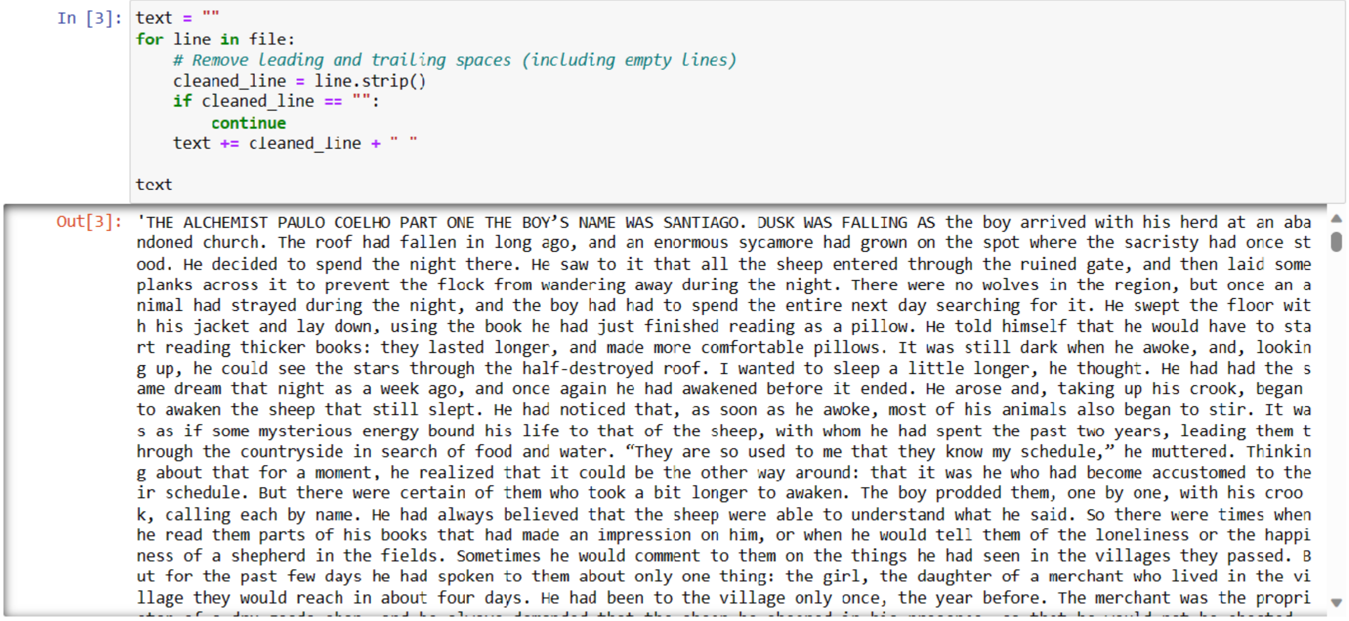
In this round, we will primarily use the ‘spaCy’ library in Python. spaCy is a free open-source library for Natural Language Processing in Python. It features NER, POS tagging, dependency parsing, word vectors and makes information extraction and general-purpose natural language processing easier.

Installing spaCy Library



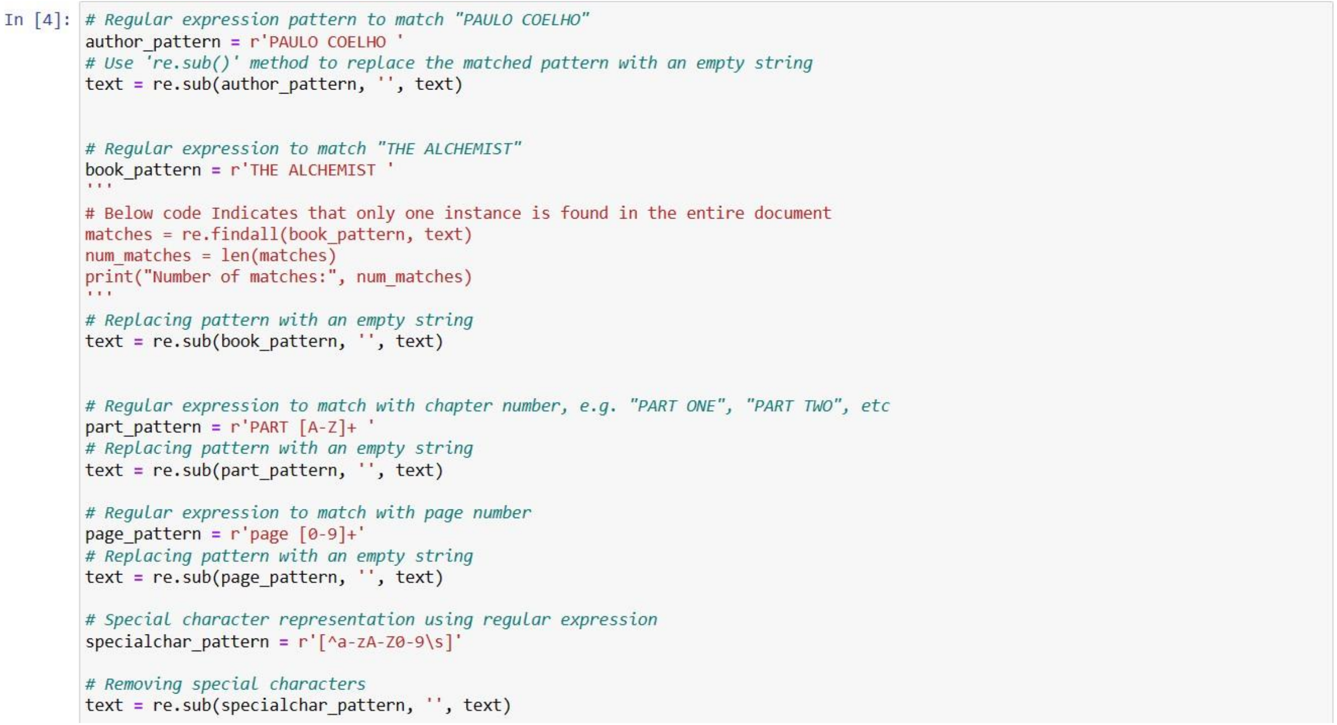
```
In [1]: import pandas as pd
        import re
        import nltk
        #nltk.download('punkt')
        from nltk.corpus import stopwords
        #nltk.download('stopwords')
        from nltk.tokenize import word_tokenize
        from nltk.probability import FreqDist
        from wordcloud import WordCloud
        import matplotlib.pyplot as plt
        #nltk.download('averaged_perceptron_tagger')
        #nltk.download('treebank')
        from nltk.util import ngrams
        import random
        import sys
        import spacy
        from spacy import displacy
        import numpy as np
        import seaborn as sns
        from sklearn.metrics import confusion_matrix
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.metrics.pairwise import cosine_similarity
```

Data Preprocessing

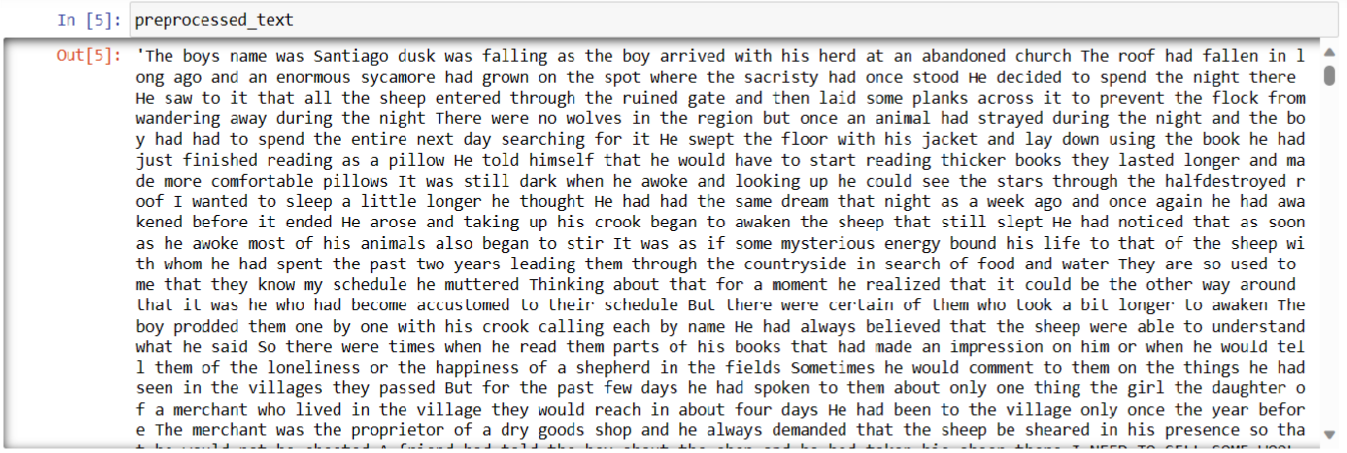


```
In [3]: text = ""
        for line in file:
            # Remove leading and trailing spaces (including empty lines)
            cleaned_line = line.strip()
            if cleaned_line == "":
                continue
            text += cleaned_line + " "

        text
```

```
Out[3]: 'THE ALCHEMIST PAULO COELHO PART ONE THE BOY’S NAME WAS SANTIAGO. DUSK WAS FALLING AS the boy arrived with his herd at an aba
        ndoned church. The roof had fallen in long ago, and an enormous sycamore had grown on the spot where the sacristy had once st
        ood. He decided to spend the night there. He saw to it that all the sheep entered through the ruined gate, and then laid some
        planks across it to prevent the flock from wandering away during the night. There were no wolves in the region, but once an a
        nimal had strayed during the night, and the boy had had to spend the entire next day searching for it. He swept the floor wit
        h his jacket and lay down, using the book he had just finished reading as a pillow. He told himself that he would have to sta
        rt reading thicker books: they lasted longer, and made more comfortable pillows. It was still dark when he awoke, and, lookin
        g up, he could see the stars through the half-destroyed roof. I wanted to sleep a little longer, he thought. He had had the s
        ame dream that night as a week ago, and once again he had awakened before it ended. He arose and, taking up his crook, began
        to awaken the sheep that still slept. He had noticed that, as soon as he awoke, most of his animals also began to stir. It wa
        s as if some mysterious energy bound his life to that of the sheep, with whom he had spent the past two years, leading them t
        hrough the countryside in search of food and water. “They are so used to me that they know my schedule,” he muttered. Thinkin
        g about that for a moment, he realized that it could be the other way around: that it was he who had become accustomed to the
        ir schedule. But there were certain of them who took a bit longer to awaken. The boy prodded them, one by one, with his croo
        k, calling each by name. He had always believed that the sheep were able to understand what he said. So there were times when
        he read them parts of his books that had made an impression on him, or when he would tell them of the loneliness or the happi
        ness of a shepherd in the fields. Sometimes he would comment to them on the things he had seen in the villages they passed. B
        ut for the past few days he had spoken to them about only one thing: the girl, the daughter of a merchant who lived in the vi
        llage they would reach in about four days. He had been to the village only once, the year before. The merchant was the propri
```



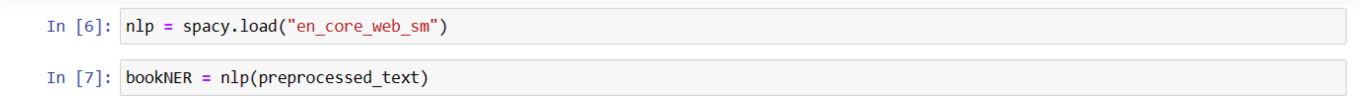
```
In [4]: # Regular expression pattern to match "PAULO COELHO"
        author_pattern = r'PAULO COELHO '
        # Use 're.sub()' method to replace the matched pattern with an empty string
        text = re.sub(author_pattern, '', text)


        # Regular expression to match "THE ALCHEMIST"
        book_pattern = r'THE ALCHEMIST '
        '''
        # Below code Indicates that only one instance is found in the entire document
        matches = re.findall(book_pattern, text)
        num_matches = len(matches)
        print("Number of matches:", num_matches)
        '''
        # Replacing pattern with an empty string
        text = re.sub(book_pattern, '', text)


        # Regular expression to match with chapter number, e.g. "PART ONE", "PART TWO", etc
        part_pattern = r'PART [A-Z]+ '
        # Replacing pattern with an empty string
        text = re.sub(part_pattern, '', text)

        # Regular expression to match with page number
        page_pattern = r'page [0-9]+'
        # Replacing pattern with an empty string
        text = re.sub(page_pattern, '', text)

        # Special character representation using regular expression
        specialchar_pattern = r'[^a-zA-Z0-9\s]'

        # Removing special characters
        text = re.sub(specialchar_pattern, '', text)
```



```
In [5]: preprocessed_text

Out[5]: 'The boys name was Santiago dusk was falling as the boy arrived with his herd at an abandoned church The roof had fallen in l
        ong ago and an enormous sycamore had grown on the spot where the sacristy had once stood He decided to spend the night there
        He saw to it that all the sheep entered through the ruined gate and then laid some planks across it to prevent the flock from
        wandering away during the night There were no wolves in the region but once an animal had strayed during the night and the bo
        y had had to spend the entire next day searching for it He swept the floor with his jacket and lay down using the book he had
        just finished reading as a pillow He told himself that he would have to start reading thicker books they lasted longer and ma
        de more comfortable pillows It was still dark when he awoke and looking up he could see the stars through the halfdestroyed r
        oof I wanted to sleep a little longer he thought He had had the same dream that night as a week ago and once again he had awa
        kened before it ended He arose and taking up his crook began to awaken the sheep that still slept He had noticed that as soon
        as he awoke most of his animals also began to stir It was as if some mysterious energy bound his life to that of the sheep wi
        th whom he had spent the past two years leading them through the countryside in search of food and water They are so used to
        me that they know my schedule he muttered Thinking about that for a moment he realized that it could be the other way around
        that it was he who had become accustomed to their schedule But there were certain of them who took a bit longer to awaken The
        boy prodded them one by one with his crook calling each by name He had always believed that the sheep were able to understand
        what he said So there were times when he read them parts of his books that had made an impression on him or when he would tel
        l them of the loneliness or the happiness of a shepherd in the fields Sometimes he would comment to them on the things he had
        seen in the villages they passed But for the past few days he had spoken to them about only one thing the girl the daughter o
        f a merchant who lived in the village they would reach in about four days He had been to the village only once the year befor
        e The merchant was the proprietor of a dry goods shop and he always demanded that the sheep be sheared in his presence so tha
```

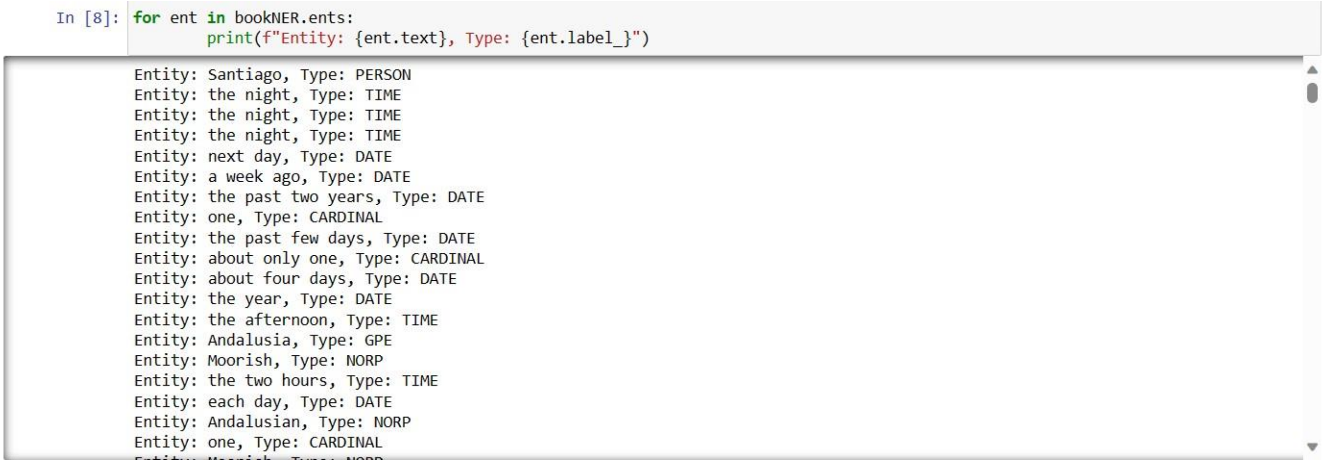
Entity Recognition

To generate all the entities, we will first import the "spaCy" library and use it to process the text.



```
In [6]: nlp = spacy.load("en_core_web_sm")

In [7]: bookNER = nlp(preprocessed_text)
```

Now iterating through all the entities in the book using the library.

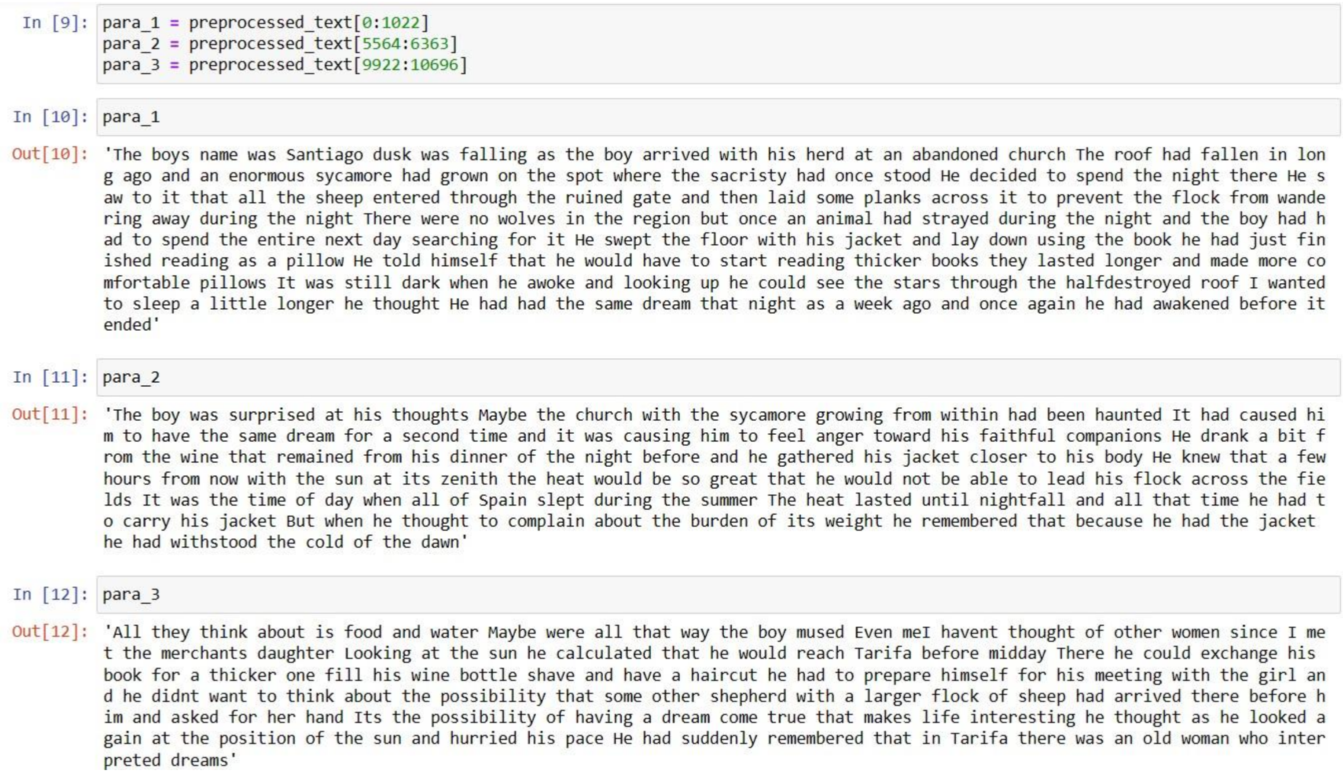


```
In [8]: for ent in bookNER.ents:
                print(f"Entity: {ent.text}, Type: {ent.label_}")
```

```
Entity: Santiago, Type: PERSON
Entity: the night, Type: TIME
Entity: the night, Type: TIME
Entity: the night, Type: TIME
Entity: next day, Type: DATE
Entity: a week ago, Type: DATE
Entity: the past two years, Type: DATE
Entity: one, Type: CARDINAL
Entity: the past few days, Type: DATE
Entity: about only one, Type: CARDINAL
Entity: about four days, Type: DATE
Entity: the year, Type: DATE
Entity: the afternoon, Type: TIME
Entity: Andalusia, Type: GPE
Entity: Moorish, Type: NORP
Entity: the two hours, Type: TIME
Entity: each day, Type: DATE
Entity: Andalusian, Type: NORP
Entity: one, Type: CARDINAL
Entity: Moorish, Type: NORP
```

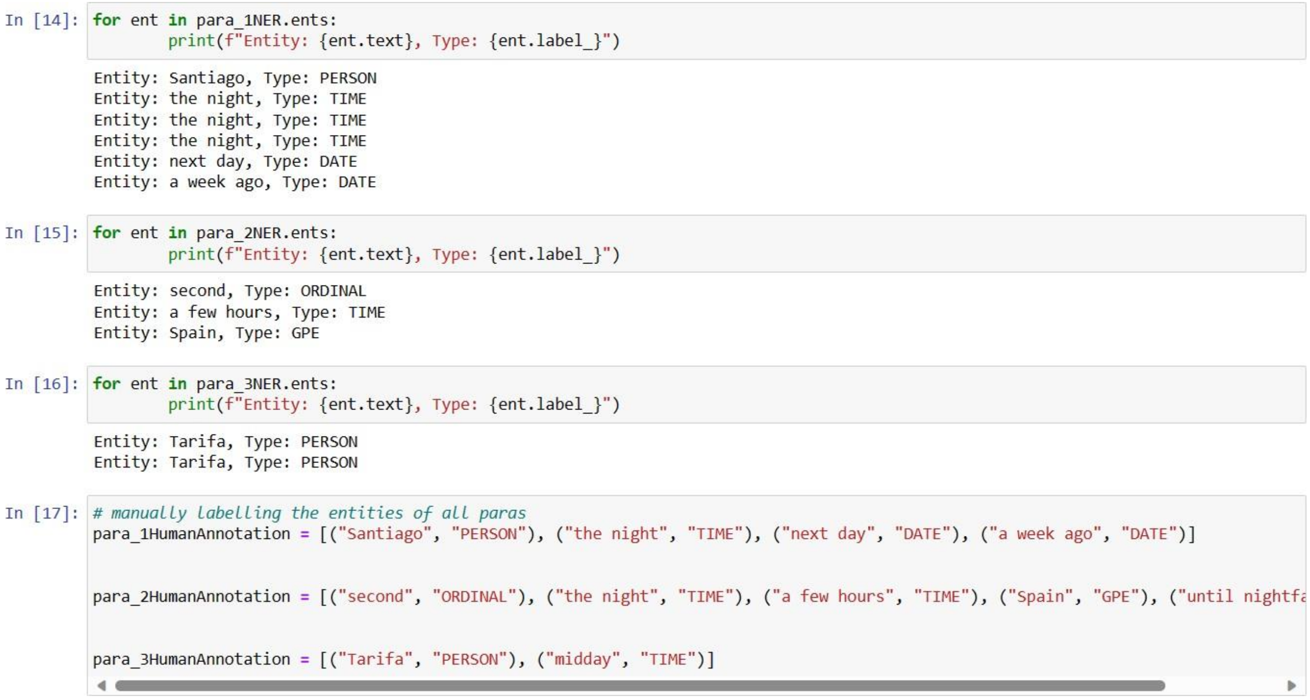
Evaluation using F1 Score

To calculate F1 score, we need to calculate precision and recall. For this, we first need to manually label the paragraphs we have chosen. The paragraphs are manually labelled using the entities used in tagging.

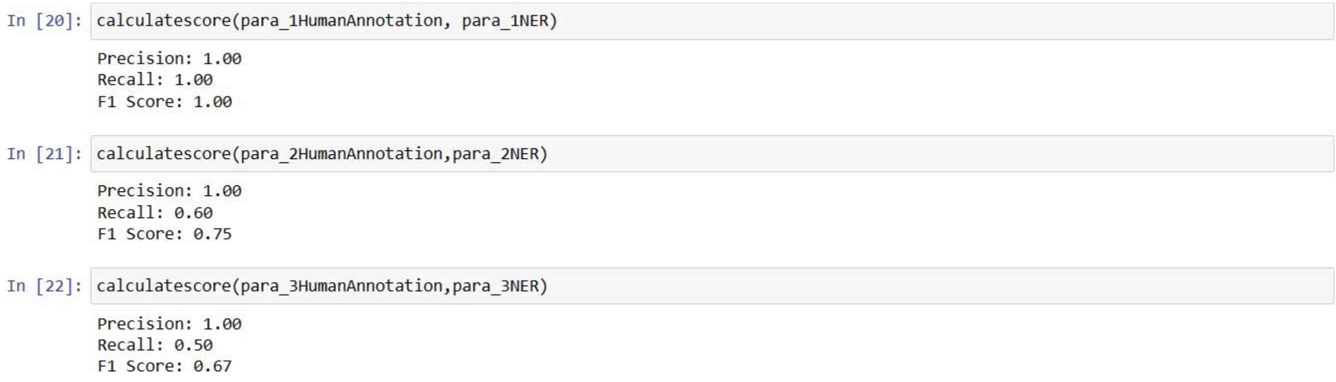


```
In [9]:  para_1 = preprocessed_text[0:1022]
         para_2 = preprocessed_text[5564:6363]
         para_3 = preprocessed_text[9922:10696]

In [10]: para_1

Out[10]: 'The boys name was Santiago dusk was falling as the boy arrived with his herd at an abandoned church The roof had fallen in lon
         g ago and an enormous sycamore had grown on the spot where the sacristy had once stood He decided to spend the night there He s
         aw to it that all the sheep entered through the ruined gate and then laid some planks across it to prevent the flock from wande
         ring away during the night There were no wolves in the region but once an animal had strayed during the night and the boy had h
         ad to spend the entire next day searching for it He swept the floor with his jacket and lay down using the book he had just fin
         ished reading as a pillow He told himself that he would have to start reading thicker books they lasted longer and made more co
         mfortable pillows It was still dark when he awoke and looking up he could see the stars through the halfdestroyed roof I wanted
         to sleep a little longer he thought He had had the same dream that night as a week ago and once again he had awakened before it
         ended'

In [11]: para_2

Out[11]: 'The boy was surprised at his thoughts Maybe the church with the sycamore growing from within had been haunted It had caused hi
         m to have the same dream for a second time and it was causing him to feel anger toward his faithful companions He drank a bit f
         rom the wine that remained from his dinner of the night before and he gathered his jacket closer to his body He knew that a few
         hours from now with the sun at its zenith the heat would be so great that he would not be able to lead his flock across the fie
         lds It was the time of day when all of Spain slept during the summer The heat lasted until nightfall and all that time he had t
         o carry his jacket But when he thought to complain about the burden of its weight he remembered that because he had the jacket
         he had withstood the cold of the dawn'

In [12]: para_3

Out[12]: 'All they think about is food and water Maybe were all that way the boy mused Even meI havent thought of other women since I me
         t the merchants daughter Looking at the sun he calculated that he would reach Tarifa before midday There he could exchange his
         book for a thicker one fill his wine bottle shave and have a haircut he had to prepare himself for his meeting with the girl an
         d he didnt want to think about the possibility that some other shepherd with a larger flock of sheep had arrived there before h
         im and asked for her hand Its the possibility of having a dream come true that makes life interesting he thought as he looked a
         gain at the position of the sun and hurried his pace He had suddenly remembered that in Tarifa there was an old woman who inter
         preted dreams'
```

Calculating F1 score for each paragraph:



```
In [14]: for ent in para_1NER.ents:
                 print(f"Entity: {ent.text}, Type: {ent.label_}")

         Entity: Santiago, Type: PERSON
         Entity: the night, Type: TIME
         Entity: the night, Type: TIME
         Entity: the night, Type: TIME
         Entity: next day, Type: DATE
         Entity: a week ago, Type: DATE

In [15]: for ent in para_2NER.ents:
                 print(f"Entity: {ent.text}, Type: {ent.label_}")

         Entity: second, Type: ORDINAL
         Entity: a few hours, Type: TIME
         Entity: Spain, Type: GPE

In [16]: for ent in para_3NER.ents:
                 print(f"Entity: {ent.text}, Type: {ent.label_}")

         Entity: Tarifa, Type: PERSON
         Entity: Tarifa, Type: PERSON

In [17]: # manually labelling the entities of all paras
         para_1HumanAnnotation = [("Santiago", "PERSON"), ("the night", "TIME"), ("next day", "DATE"), ("a week ago", "DATE")]


         para_2HumanAnnotation = [("second", "ORDINAL"), ("the night", "TIME"), ("a few hours", "TIME"), ("Spain", "GPE"), ("until nightfa


         para_3HumanAnnotation = [("Tarifa", "PERSON"), ("midday", "TIME")]
```



```
In [20]: calculatescore(para_1HumanAnnotation, para_1NER)

         Precision: 1.00
         Recall: 1.00
         F1 Score: 1.00

In [21]: calculatescore(para_2HumanAnnotation,para_2NER)

         Precision: 1.00
         Recall: 0.60
         F1 Score: 0.75

In [22]: calculatescore(para_3HumanAnnotation,para_3NER)

         Precision: 1.00
         Recall: 0.50
         F1 Score: 0.67
```

Paragraph-1:

1. Precision = 1.00
2. Recall = 1.00
3. F1 score = 1.00

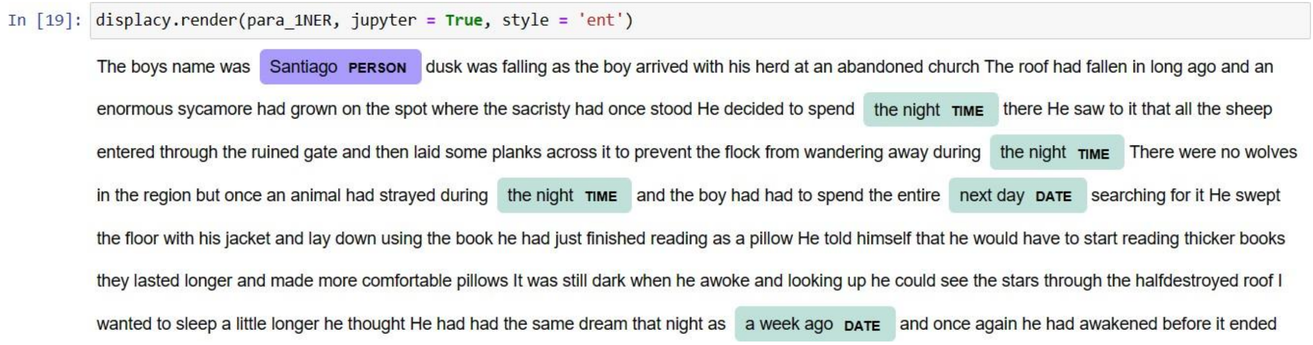
Paragraph-2:

1. Precision = 1.00
2. Recall = 0.60
3. F1 score = 0.75

Paragraph-3:

1. Precision = 1.00
2. Recall = 0.50
3. F1 score = 0.67

Visualization :-



```
In [19]: displacy.render(para_1NER, jupyter = True, style = 'ent')
```

The boys name was Santiago PERSON dusk was falling as the boy arrived with his herd at an abandoned church The roof had fallen in long ago and an enormous sycamore had grown on the spot where the sacristy had once stood He decided to spend the night TIME there He saw to it that all the sheep entered through the ruined gate and then laid some planks across it to prevent the flock from wandering away during the night TIME There were no wolves in the region but once an animal had strayed during the night TIME and the boy had had to spend the entire next day DATE searching for it He swept the floor with his jacket and lay down using the book he had just finished reading as a pillow He told himself that he would have to start reading thicker books they lasted longer and made more comfortable pillows It was still dark when he awoke and looking up he could see the stars through the halfdestroyed roof I wanted to sleep a little longer he thought He had had the same dream that night as a week ago DATE and once again he had awakened before it ended

TF-IDF Vector

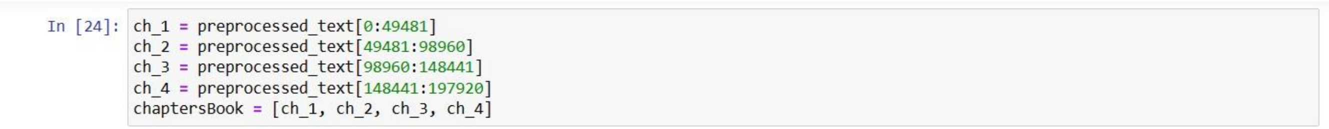
TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical statistic in Natural Language Processing that reflects the importance of a word in a document relative to a collection of documents (corpus). It is commonly used for text mining and information retrieval. The formula for TF-IDF is given by:

TF = (Occurrence of a word in a document) / (Number of words in the document)

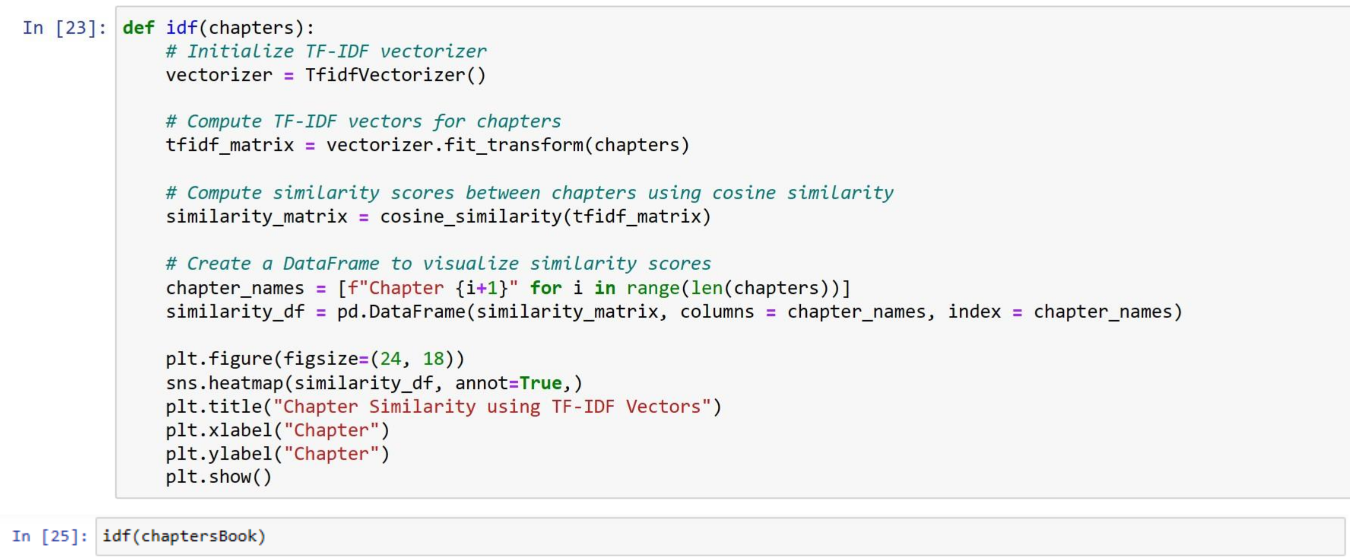
IDF = Log [(Total number of documents) / (Number of documents containing the word + 1)]

TF-IDF = TF * IDF

TF-IDF is finally used for checking similarity between chapters of the book. The following is process for calculating TF-IDF:

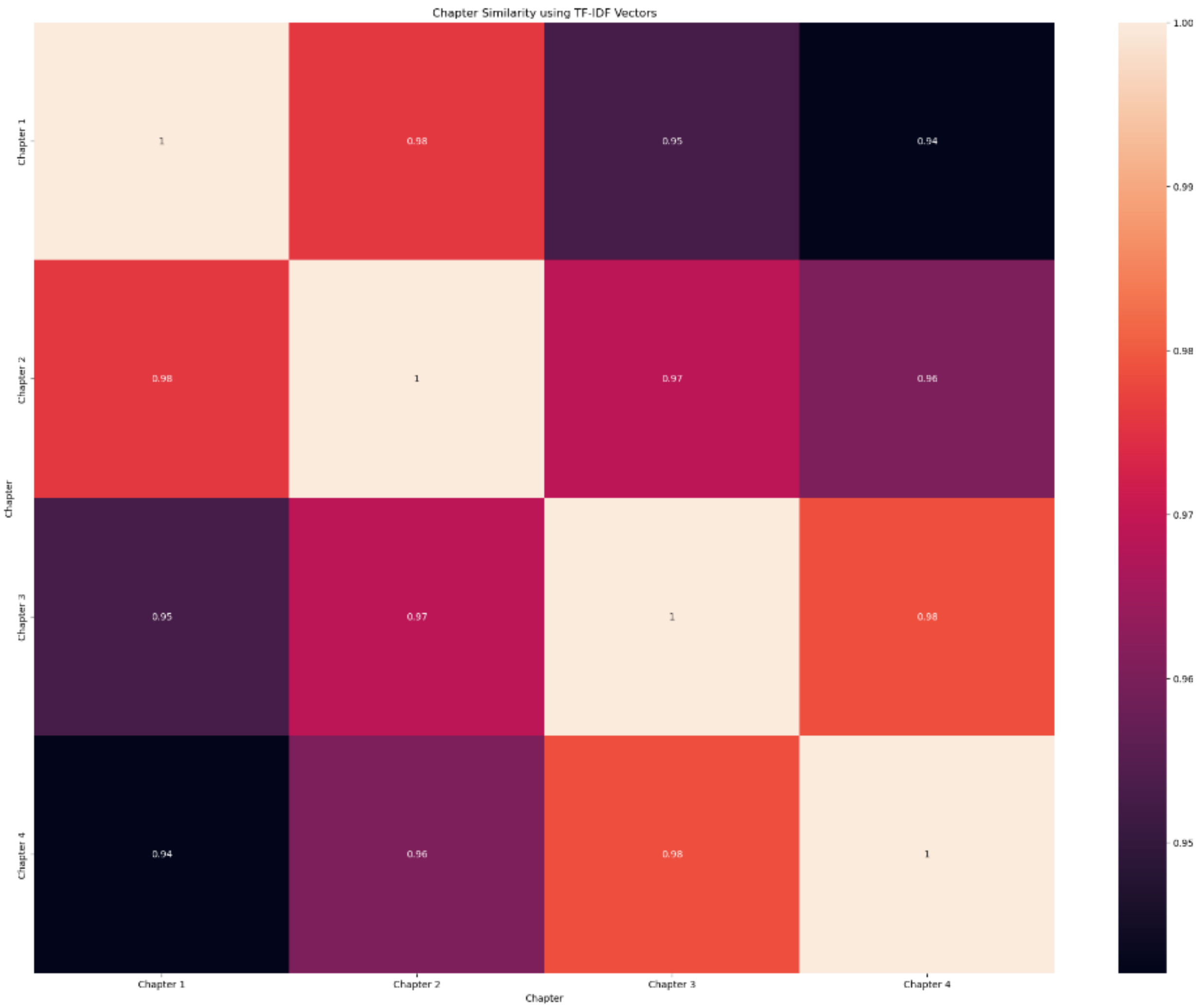


```
In [24]: ch_1 = preprocessed_text[0:49481]
         ch_2 = preprocessed_text[49481:98960]
         ch_3 = preprocessed_text[98960:148441]
         ch_4 = preprocessed_text[148441:197920]
         chaptersBook = [ch_1, ch_2, ch_3, ch_4]
```



```
In [23]: def idf(chapters):
             # Initialize TF-IDF vectorizer
             vectorizer = TfidfVectorizer()

             # Compute TF-IDF vectors for chapters
             tfidf_matrix = vectorizer.fit_transform(chapters)

             # Compute similarity scores between chapters using cosine similarity
             similarity_matrix = cosine_similarity(tfidf_matrix)

             # Create a DataFrame to visualize similarity scores
             chapter_names = [f"Chapter {i+1}" for i in range(len(chapters))]
             similarity_df = pd.DataFrame(similarity_matrix, columns = chapter_names, index = chapter_names)

             plt.figure(figsize=(24, 18))
             sns.heatmap(similarity_df, annot=True,)
             plt.title("Chapter Similarity using TF-IDF Vectors")
             plt.xlabel("Chapter")
             plt.ylabel("Chapter")
             plt.show()
```

```
In [25]: idf(chaptersBook)
```



The table showcases the similarity of each chapter with every other chapter using the TF-IDF vectors generated earlier.

Inferences

The following observations were made after using the NLP frameworks on the document ‘The Alchemist’:

1. From the TF-IDF similarity matrix, we can see that the similarity between same chapters is always 1 which is obvious.
2. The TF-IDF similarity matrix is a symmetric positive definite matrix.