Strategy:

I have used JSoup Library to parse the HTML documents. It does the job of eliminating the tags and HTML entities which are not required to analyze the data. One hash Map is globally maintained to store words vs frequency. The logic checks whether the word is encountered for the first time or already visited. Its frequency is evaluated accordingly. One map is created for each document when parsed to record the details of its words. In this iteration for each document, global map is also updated as per visiting words. Further, for each document, its corresponding property file is also created in which its tokens and frequency is stored.

Handling Punctuations:

I made sure that all non-relevant characters are removed before evaluating frequency. Firstly, all digits are replaced by empty string because the focus of assignment is to evaluate word frequency. Then, I am using customized regex to split words on the basis of certain special characters.

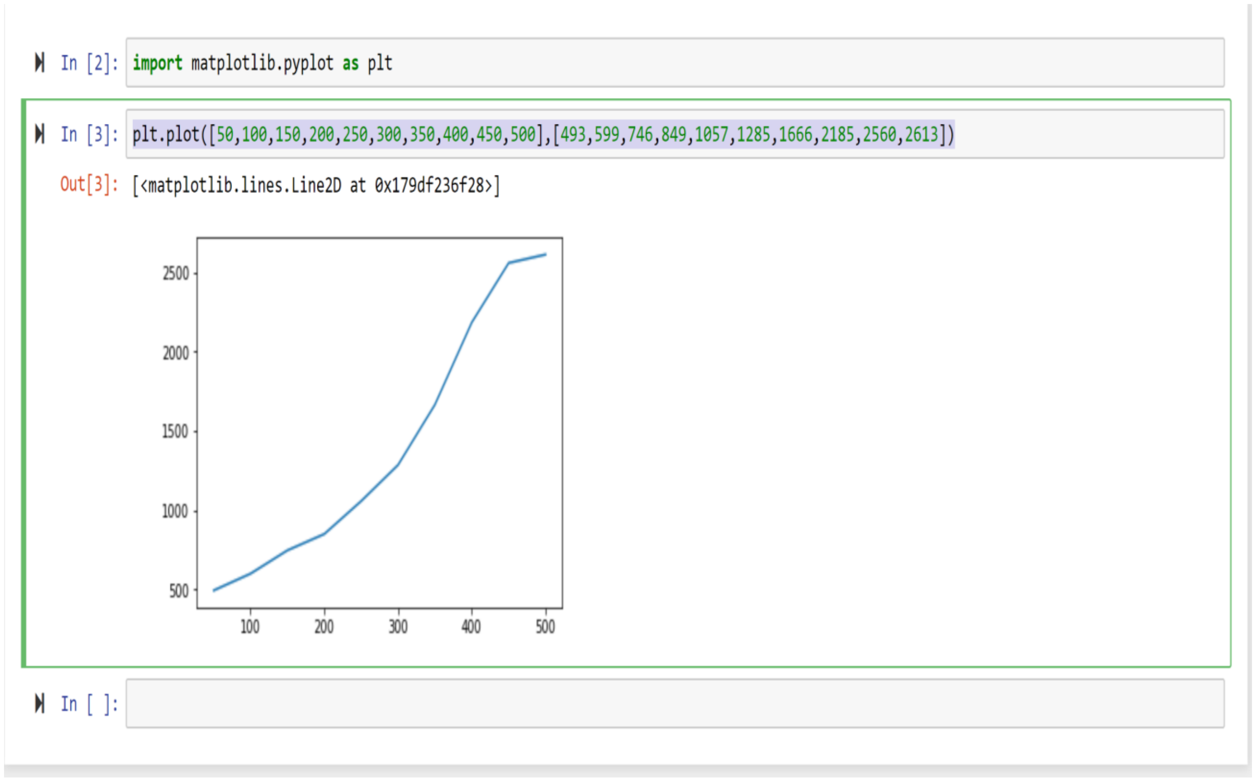
Regex: (" \t\n\r\f,.:;?![]{}()|%#$/<>@\\'*_+-\"=&`")

These characters like # and : does not present any valuable information, but their surrounding words may be important. Same argument applies for other mentioned special characters. I have not used a simple regex like (^a-zA-Z), as this also eliminates certain words like félix. But these words may be important when we are dealing with search queries. This is the reason; a complex regex is created to such capture important details.

Performance Table:

Number of Input Documents	Time Taken (ms)
50	493
100	599
150	746
200	849
250	1057
300	1285
150	1666
400	2185
450	2560
500	2613

Graph-Time Vs #Input Files



```
In [2]: import matplotlib.pyplot as plt
```

```
In [3]: plt.plot([50,100,150,200,250,300,350,400,450,500],[493,599,746,849,1057,1285,1666,2185,2560,2613])
Out[3]: [<matplotlib.lines.Line2D at 0x179df236f28>]
```

Installation Instructions:

I used Apache Maven as a building tool. Pom.xml can be found in the code directory.

Following command can be used to create a executable jar:

mvn clean install

Entry point class is WordFrequencyCount.java. It accepts two arguments- input-fil-dir and output-file-dir.

To execute jar, following command can be used:

Java -jar assignment1-0.0.1-SNAPSHOT <inputFileDir> <outputFileDir>

Sometimes, maven may not update Main-class parameter in the Manifest of jar, so it should be updated manually as:

Main-Class: informationRetrieval.assignment1.WordFrequencyCount

Program can also be executed using javac and java commands.

javac informationRetrieval.assignment1.WordFrequencyCount.java (to compile)

java informationRetrieval.assignment1.WordFrequencyCount.java <inputFileDir> <outputDir> (to run)

Comparisn :

I compared my program with Shaunak's program. Execution time wise, my program was 10 times faster. Main reason was that I used java and Shaunak used Python. He used Beautiful Soup python package to parse HTML. I used JSoup, python package performed better in terms of meaningful parsing. He used not-so-complex complex regex, and python tokenizer automatically eliminated certain non-required Unicode entities. In my case, certain (less frequent) words appeared containing Unicode entities