WORD DENSITY ANALYSIS

What is the problem?

"Given any URL, be able to classify the web page, and return a list of relevant topics for the web page."

What is the input?

- Any URL

What is the output required?

- List of common keywords that best describe the contents of that page

How did I arrive at the solution?

First, I analysed the given sample URL's HTML page structure. Next, I analysed other URL's that is different from these given websites. I realized that most of these websites are either categorized into News, Blogs or Shopping sites which had their content written within either the body element or article element. But, my requirement is to retrieve the text within the HTML elements so I followed a basic algorithm described below.

Basic Algorithm

1) Fetch the contents of the HTML page as a text

2) Store the results in a List and normalize the List elements one by one.

3) Read the stop words from a file and remove all the stop words from the result List. Because, not all words fetched from the webpage are important to us.

4) Store the non-stop words in a Map with its key and value. Key being the word itself and the value being the "Frequency".

5) Loop through the non-stop words and perform the word density calculation based on a Threshold value.

6) Print all the most common keywords.

To implement this, I mainly searched for an external library that can parse webpages quickly and get me the results what I want. I found two of them namely: JSoup and HtmlUnit. Both of them were java web parsers. Since, my first instruction in the algorithm is to retrieve the contents of an html page I went on with *"JSOUP".*

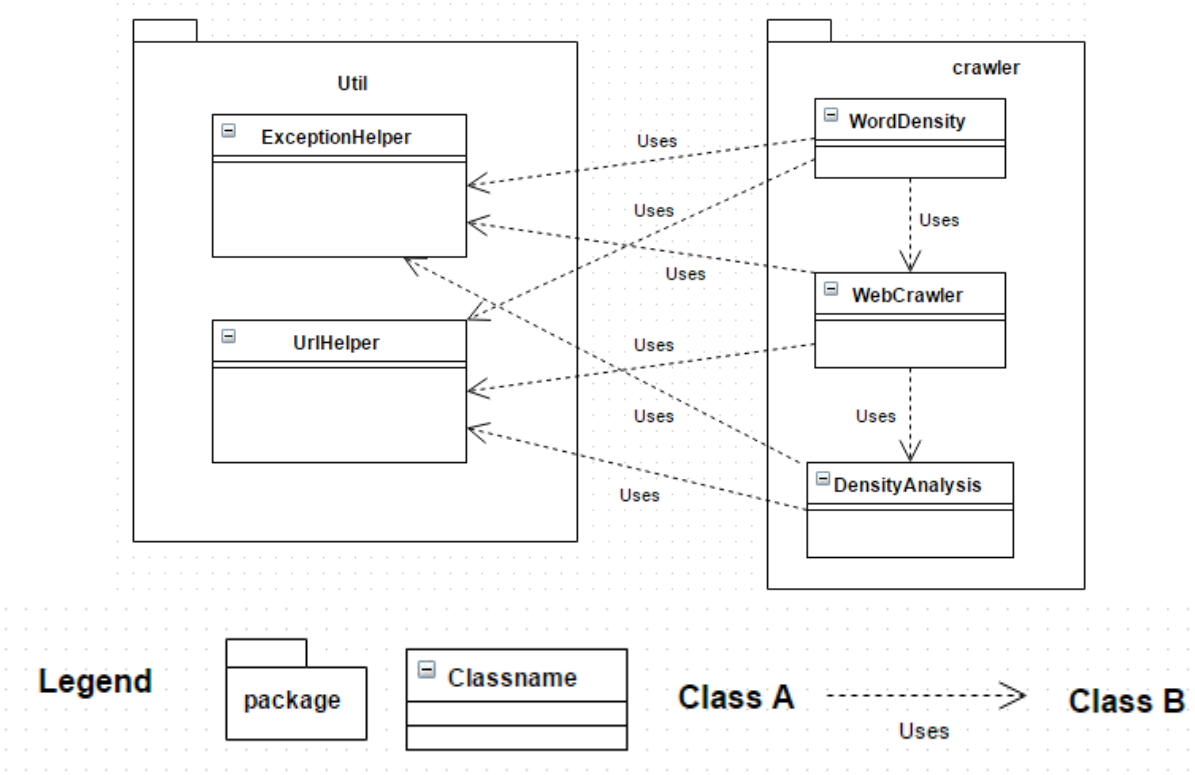
To ensure that my algorithm is implemented in the right way, I followed a step-by-step process which is mentioned below.

1) Design a class diagram for the application
2) Get the required libraries
3) Create a Stop-Word text file containing all stop-words collected from various external resources.
4) Develop the code
5) Sample Input and Output using the jar file

These are explained in detail in the upcoming section.

Solution Overview

- *Class Diagram* - The class diagram for this assignment is shown diagrammatically below. It was designed using the UML standards. The architecture follows a layered pattern and has three layers: UI, Filter and Util. By following this pattern, we can easily modify the source code with less effort. The responsibilities of these classes are described below. The responsibilities of each class is described in the *"ReadMe"* file present in the source folder.



- I then downloaded the "*Jsoup 1.8.1*" library into the source folder and tested a sample application by executing the following line:

  Document doc = Jsoup.connect(URL).timeout(1000).get();
  Scanner sc = new Scanner(doc.text());

  Once, this was a success then I proceed with my development process.

- I created a text file called "StopWords.txt" within the jar file which contains all the stop-words that are required to filter the keywords retrieved from the HTML page using the JSoup library. These stop-words are fetched from external resources such as Wikipedia, seobook.com, Ranks.nl and lektek.com.
- I collect all the words retrieved from the HTML page using JSoup as a Text in a Scanner object and then iterate over the scanner object to retrieve one by one and store them in a list. I pass this list to another class which normalizes these words into pure keywords including the stop-words. In that class (WebCrawler), I use some special symbols, punctuations and other patterns to normalize them and return to the list as keywords. Then, I pass the list to another method which takes care of removing the stop-words from the list. I iterate through the collection of words present in the list

and I check against the collected "Stop Words" list and consider them as **relevant keyword** only when that word is not present in list of stop-words. I then check whether the non-stop word is present in the Map or not. If found then I increment the key value or add the new key along with 1 in the Map. Finally, the Map will contain all non-stop words along with their frequency.

- The word density is then calculated for every word in Map using this formula: (frequency of word/total number of words) * 100. Once, the density is calculated I immediately check with the Threshold values both Min and Max which are provided in the UrlHelper file. If the calculated density of a word falls within the Threshold values then I have found the most common keyword and print it in the output screen. These Threshold values can be modified to print n number of common keywords. The Threshold was assigned based on the concept: linked and non-linked words. It means that duplicated words in the Map are called "*Linked words"* and unique ones are called "*Non-Linked words"*. To differentiate between these two I set a value of 1 and 10 respectively for Min and Max Threshold.

Sample Output

1. I tested the application by entering the following URL:

   http://www.amazon.com/Cuisinart-CPT-122-Compact-2-Slice-Toaster/dp/B009GQ034C/ref=sr_1_1?s=kitchen&ie=UTF8&qid=1431620315&sr=1-1&keywords=toaster

   The output is shown in the below screen.

```
C:\Users\vinod\Desktop>java -jar Assignment.jar http://www.amazon.com/Cuisinart-
CPT-122-Compact-2-Slice-Toaster/dp/B009GQ034C/ref=sr_1_1?s=kitchen&ie=UTF8&qid=1
431620315&sr=1-1&keywords=toaster
Connection is established to the website...

Started Crawling into WebSite...

Fetching Common Keywords...

There are 3357 words in this page.
There are 921 non-stop words in this page.
Of those 921 words 307 words are linked ones.

Most Common Keywords based on the Threshold value are...

Add
toast
Compact
Toaster
2-Slice
helpful
stars
Customer
Size
Published
Shipping
bread
toaster
Kitchen
Prime
feedback
inches
review
Cuisinart
Amazon
Cart
Amazon.com

Finished fetching common keywords...
```

Figure 1 – Amazon URL

2. To verify the application again, I entered another URL:

   http://www.cnn.com/2013/06/10/politics/edward-snowden-profile/

   The output is shown below:

```
C:\Users\vinod\Desktop>java -jar Assignment.jar http://www.cnn.com/2013/06/10/po
litics/edward-snowden-profile/
Connection is established to the website...

Started Crawling into WebSite...

Fetching Common Keywords...

There are 1574 words in this page.
There are 502 non-stop words in this page.
Of those 502 words 146 words are linked ones.

Most Common Keywords based on the Threshold value are...

WATCH
WATCHED
Edward
surveillance
intelligence
government
Replay
worked
programs
U.S.
Obama
NSA
Snowden
contractor
leaks
Videos

Finished fetching common keywords...
```

Figure 2 – CNN URL

Future Work

There can be many improvements in the current application to make it a robust one in the future.

Some of them I have highlighted below:

1. Use of better data structure such as "*Trie*" instead of HashMap for storing words as Key and Value pair. Trie data structure is used to store words with root and along with it the number of vertices and word count which makes it easier to retrieve words.
2. Some kind of thought can be given towards the relevancy of keywords fetched towards the article title or page content. Based on relevancy, we can perform filtering of words rather than looping thru a set of words list and calculating the word density for each word and finally print the set of words.
3. To make the filter finer and accurate, we can include more number of stop-words from other external resources apart from the resources which I have mentioned above. This helps to fine tune the word list and makes it easier to sort the common keywords easily.
4. We can even perform the same application to search for relevant code snippets rather than depending on search engines like google, stackoverflow, kruggle and blackduck, etc. to return just the snippets based on search query without any relevancy between the search query and the user's development language.