GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Department of Computer Science & Applied Physics

B.Sc. Software Development – Artificial Intelligence (2020)
ASSIGNMENT DESCRIPTION & SCHEDULE

A Web Opinion Visualiser

Note: This assignment constitutes 50% of the total marks for this module.

Word-clouds are a mechanism for creating a visual representation of text and are used to display a visual summary of the most prominent words used on a web page, a news forum or a social media web site. A word-cloud is comprised of a set of tags, with each tag representing a single word. The prominence of a word is typically estimated from its occurrence frequency in a text and is visualised using a large font size or different font colour.

Minimum Requirements

You are required to develop a multithreaded AI search application that can generate a word cloud from the top 20 words associated with an internet search term. The application should:

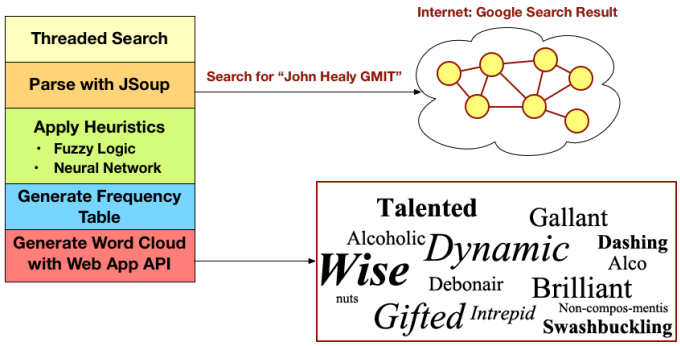
1. Use the **JSoup API** (*http://www.jsoup.org*) to parse the results returned from ***DuckDuckGo*** for a given term and goal threshold and identify a set of candidate child nodes. Consider the HTML page of results as an initial starting node. Web pages should be scored by heuristics specified using the **JFuzzyLogic API** and any other heuristic schema(s) of your choosing. You can also use the **Encog API** to add a neural network to your application. *Your application should ignore any words it encounters that are contained in the list of "ignore words" provided*.

2. Use an **AI search(es)** to follow the hyperlinks contained in the HTML page of search results. Each HTML page should be parsed with *JSoup* and its relevant contents indexed. You are free to use any AI search algorithm(s) or combination/variation. Extra marks will be given for the correct use and exploitation of heuristics. For example, if a search term in found inside a page title of <h1> tag, it is probably more significant than a word found inside a paragraph of text. You could also apply string distance metrics to associate scores with the distance of a string, e.g. Euclidean, Jaccard etc., from the search term or compute the adjacency of a word to the search term.

3. When the search has concluded, your application should use the **Word Cloud generator available in the web application on Moodle** to generate a word cloud from the most frequently occurring words in the index. Your application should allow the user to choose the maximum number of words to display. The word cloud API only requires that the *n* results to display are supplied in an array of *WordFrequency* objects. The class *WordFrequency* is a value class for a String word and its frequency.

4. The application must be **threaded**, i.e. your application should be able to execute *n* parsers in parallel, one for each search algorithm employed. You should consider using a thread pool to control the number of concurrent parsers in your application.

A diagrammatic representation of the application is given below.



The following Java snippet illustrates how to access the search results from **DuckDuckGo** using **JSoup** (do not use the Google API, as it only adds a layer of unnecessary complexity to the problem):

```
String query = "GMIT Software";
Document doc = Jsoup.connect("https://duckduckgo.com/html/?q=" + query).get();

Elements res = doc.getElementById("links").getElementsByClass("results_links");
for(Element r: res){
   Element title = r.getElementsByClass("links_main").first().getElementsByTag("a").first();
   System.out.println("URL:\t" + title.attr("href"));
   System.out.println("Title:\t" + title.text());
   System.out.println("Text:\t" + r.getElementsByClass("result__snippet").first().wholeText());
}
```

As the whole objective of the assignment is to reinforce your understanding of AI search algorithms, their uses and limitations, you are encouraged to modify or use any combination of search approaches that you like. In particular, the use of good heuristics and scoring functions will be richly rewarded. If you decide to use a hash map as an index, you will need think carefully about how to deal with multiple threads updating or searching the map simultaneously. *Note: you are free to asset-strip any of the resources, including labs and source code, available on the Moodle page for this module.*

Deployment and Submission

- ***The project must be submitted by midnight on Friday 17th April 2020*** as a Zip archive ***(not a rar or WinRar file)*** using the Moodle upload utility. You can find the area to upload the project under the *"A Web Opinion Visualiser - (50%) Assignment Upload"* heading in the *Assignment* section of Moodle.
- The name of the Zip archive should be ***{id}*.zip** where *{id}* is your student number.
- You must use the package name **ie.gmit.sw** for the assignment.
- Do not include any additional JAR archives with your submission. The libraries **jsoup-1.12.1.jar**, **jFuzzyLogic.jar** and **encog-core-3.4.jar** are all already packaged correctly in the WEB-INF/lib directory of the web application.
- The Zip archive should have the following structure (do NOT submit the assignment as an Eclipse project):

Marks	Category
wcloud.war	A **Web Application Archive containing** the resources required for a Tomcat Web Application. All environmental variables should be declared in the file WEB-INF/web.xml. You can create the WAR file with the following command from inside the "**wcloud**" folder: **jar –cf wcloud.war ***

src	A directory that contains the unpackaged web application, including JSPs, resources, WEB-INF and **source code**.
README.txt	A text file detailing the main **features** of your application. Marks will only be given for features that are described in the README.

Marking Rubric

Marks for the project will be applied using the following criteria:

Marks	Category
(10%)	**Packaging & distribution.** *All or nothing.* The web application must be structured correctly and execute without any manual intervention after being placed in the Tomcat *webapps* directory.
(10%)	**Threads.** Each search algorithm should run in a different thread and any data structures should handle concurrency correctly.
(30%)	**Heuristic Search (15% for Each of the following):** • Fully commented design in the README and implementation of heuristics for an AI search of the state space. • Implementation of algorithm(s) and integration with JSoup.
(30%)	**Fuzzy Logic (10% for Each of the following):** • Fuzzy sets and membership functions with comments. • Fuzzy rules (all sets should be covered by at least one rule) with comments The fuzzy logic FCL should be placed in a file in the **res** folder of the web application.
(20%)	**Extras** • AI extras only that have been fully described in the README. The extras must auto deploy with the rest of the web application.

Each of the categories above will be scored using the following criteria:

- 0–39% Not delivering on basic expectations
- 40-59% Meeting basic expectations
- 60–79% Tending to exceed expectations
- 80-90% Exceeding expectations
- 90-100% Exemplary