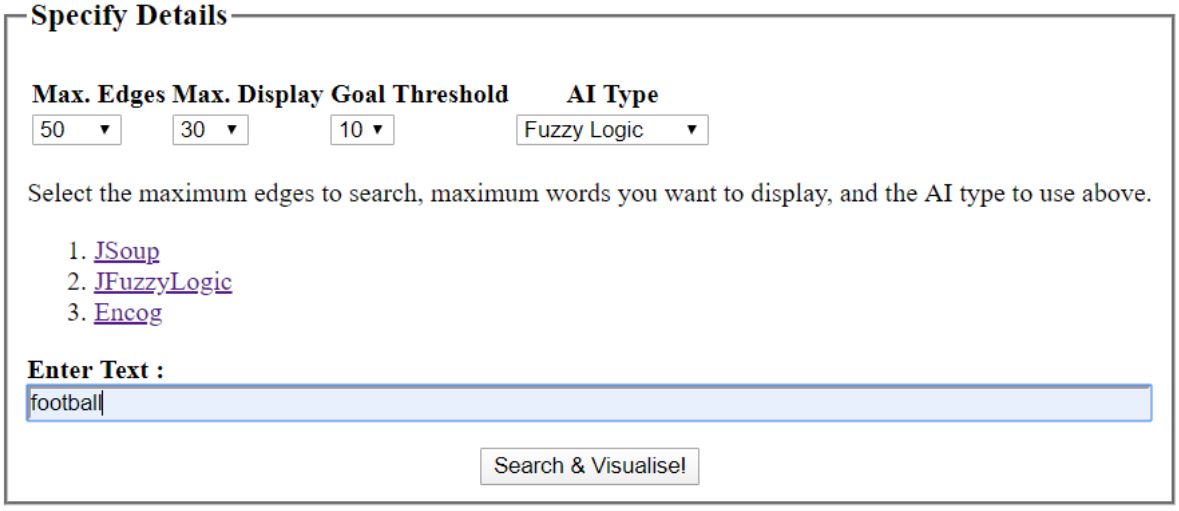
A Web Opinion Visualiser

Name: Mark NDIPENOCH
ID: G00352031

Overview

A user launches the application from Tomcat and is presented with a window as seen below.
Users can freely decide how they want the search to be.
Users can choose how many numbers or links or edges they will like to search through per result from DuckDuckGo [1], the maximum number of the most popular words they will like to see, the goal threshold or the maximum number of result they will like to go through from the DuckDuckGo [1] search.
Also, users can choose the Artificial Intelligence (AI) search type, between Fuzzy Logic[3] and Neural Network[4].
Then the user will enter the search team. The result is sent to the backend. The term is searched in DuckDuckGo [1].
For a given goal threshold each link gotten from DuckDuckGo [1] is then passed to an AI search algorithm. Each of the pages for the links is then scored by a Heuristic rule, So as to get the best pages only.
After the best pages are determined, the most popular words or most frequent words which are not in the ignore word list are arranged in descending order with the most frequent words at the top.
The top most frequent words are then displayed to the user for the given maximum display number selected by the user.
Also, some statistics like the maximum depth search and the branching factor, accuracy for a given search are displayed to the user.
PS: This application was done and compile with Java 11.4. To run this application on Tomcat you will need a Java JDK version equals or higher than 11.4.



Design and Technique

- **Design**

  The application is designed to be user freely, so user can personalize their search. This will allow the user to either increase or decrease the speed of the application depending on how the set up the search.

- **Depth-first search (DFS)**[6]

  The AI search use is DFS. This allows the search to go as far as possible down the tree. The advantage being that once the search gets to the bottom of the tree of the first results, it is most likely to have search all the possible links related to that search team.
  So, since every visited link is stored in a set list, you don't have to re-visit the links already searched. So you are most likely to get 90 percent of the search from the first result return from DuckDuckGo [1] depending on how far you want to go.
  Also, another advantage of this is that the user have a hand on this, So, they can control how far they want to go and how many links from the DuckDuckGo [1] result they will like to search on.

- **Fuzzy Logic**

  I used Fuzzy Logic to score the pages of the links. Since this is an Internet search, we will get so many results. Some of the results are good and some are bad. We are only interested in the good ones. So, I used Fuzzy Logic to score each page to determine if it is bad, good or excellent and I am only interested in the good and excellent ones.
  The inputs for the Fuzzy Logic are title, heading and paragraph. Title is the most important, followed by the heading and then the paragraph. The output is a score which is either bad, good or excellent. The inputs are scored base on rules I set.

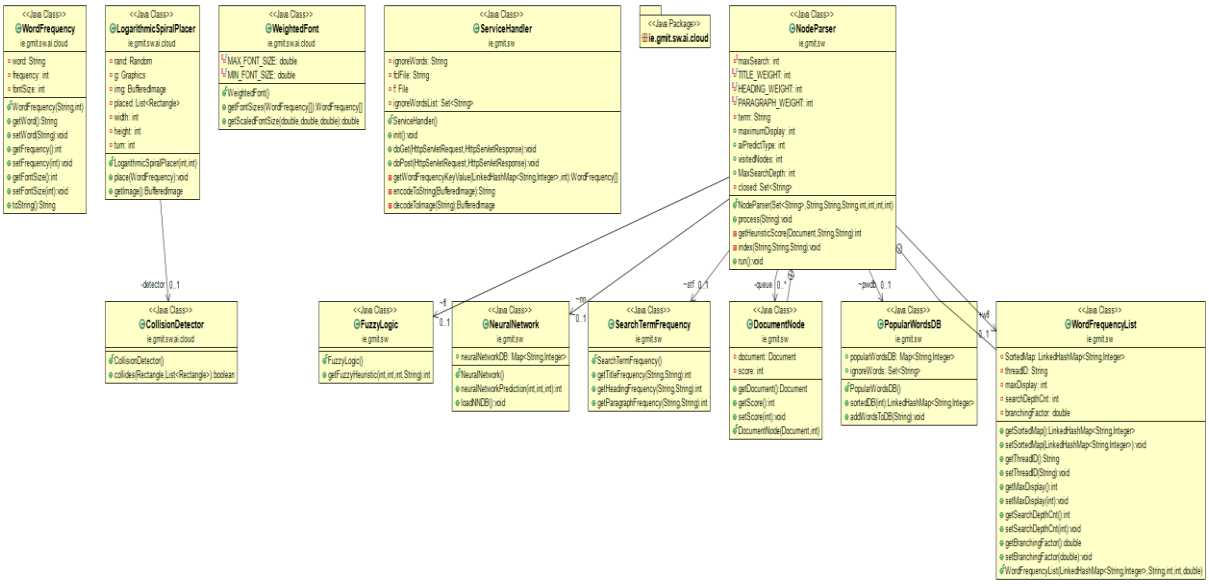
- **Heuristics**

  The search term is given a weighted value to determined how important is the webpage.
  A search term found in the title is given the highest weight of 50 per occurrence, 40 if found in the heading, and 10 if found in the body or paragraph. This is because if a search team is mentioned in the title of the page, the page is most likely about the search term than a search term mention in the headings or even less the body.

- **Adapted Neural Network**

  I have included in the application an AI logic that is between Fuzzy Logic and Neural Network. I researched on both and decided to do something original for this project. At the start I wanted to include a Neural Network to the application, but I release my input and output for this particular application is finite and not that many. The inputs are title, headings and paragraph and the output is score. I release if you combine 3 different things(title, heading and paragraphs) in 3 different ways(bad, good and excellent) in which the order matters and there are no repetitions you will get 21 outcomes. I wrote it down manually and even came up with a formula but I haven't tested the formula in all situation to see if it works. The formula is $C = W*(!n)+n$, where C, is all the possible outcomes, W is the number of ways and n is the number of things. !n = n factorial.
  Since the input and output are finite, I decided to create a database(Hash Map) with all possible combinations and results. I thought for this particular application there's no need to train a model as we already know all the possible inputs and outputs and there are finite. Also, we are guarantee of always having an accuracy rate of 100 percent since all the possible possibilities and results are already guarantee. The disadvantage of this logic is that it is not suitable for application which you don't know or have all the possibilities at hand or an application with infinite possibilities and results.

- **DuckDuckGo**
  I used DuckDuckGo as the search browser for the search term. I searched through DuckDuckGo and got a result back. Then loop over the result and for each link in the result I go as deep as indicated by the user.
- **JSoup**
  Jsoup[7], is a Java library for working with real-world HTML. It provides a very convenient API for fetching URLs and extracting and manipulating data, using the best of HTML5 DOM methods and CSS selectors. I use Jsoup API to connect and fetch data from the DuckDuckGo result and other web pages.
- **Robust**
  The application is stable and can go as far as the user wants it to go. Try and catch method are use everywhere, whether I am reading data from a file or connect to an API. In case the service is not available or the result is invalid. I make sure this is wrapped in a try and catch block, so as not to interrupt the service.
  Also, the words are converted to lower cases to maintain uniformity across .
- **Thread**
  The application is multi-threaded, the thread starts at the Service Handle class. Each search is a thread and each thread has a unique id which is the current date, time, minute, seconds and a concatenation of a random generated number between 0-100.
  After the search is done the result is store in a Linked Hash Map database. In the Service handler there is another process that keeps checking continuously if there is anything in the database if there is it pulls and check if the thread id matches what it is looking for.
  If it is what it is looking for, it pulls the Map and put it into an inner queue, the information is processed, pass to the outer queue and it is deleted from the inner queue.
  Also, to avoid multiple threads from access the same resources like databases at the same time, concurrency is used like ConcurrentSkipListSet. The thread is done at the level of the class to minimize the number of resources involved in the threaded process.
- **UML**



Conclusion

I learnt and enjoyed so much from the project. The project is interesting and directly connected to industrial problems like data mining and web search. I really enjoyed it especially having to search on the internet and get some results back in real time.

- **Problems:**
  Developing this application I came across several problems and resolved them. The biggest one was that I ran compile my code on Eclipse EE with Java JDK 11.4 and it was working fine on the Appache Tomcat server. But when I first deployed the war file on Tomcat I had an error as seen below, because I had set JAVA HOME JDK to a version lower than that 11. 4. So when I launch the application and click the "Search & Visualization" button it gave me the below error.
  I read the online chat forum on team and it was the same issues David. G had. I spent hour trying to fix it and didn't realize it was the wrong Java JDK that was set up in Tomcat. I figured that an the end and upgrade the Java JDK to 11.7 on Tomcat and it works fine.

PS: This application was done and compile with Java 11.4. To run this application on Tomcat you will need a Java JDK version equals or higher than 11.4.

- **What I will like to do?**
  I wanted to use another AI search for the application but didn't have time, so that is something I have in mind to do in the future.
  Also, I will like to develop an application that different people can use by applying the knowledge I have learnt and gained from this module.

Research

1- DuckDuckGo – Official website https://duckduckgo.com/
2- Artificial Intelligence – Paper by Peter C Cheeseman
3- https://www.researchgate.net/profile/Peter_Cheeseman2/publication/24154867_Introduction_to_artificial_intelligence/links/56eae3ae08aec6b500166171.pdf
4- Fuzzy Logic – Paper by Lofti A. Zadeh : http://www.geocities.ws/hhvillav/00000053.pdf
   a. And https://learnonline.gmit.ie/pluginfile.php/154754/mod_resource/content/1/fuzzyExpertSystemsChapter.pdf
5- Neural Network – Paper by Robert Hecht-Nielsen http://www.andrew.cmu.edu/user/nwolfe/esr/pdf/backprop.pdf
   a. And https://learnonline.gmit.ie/pluginfile.php/154755/mod_resource/content/1/neuralNetworksChapter.pdf
6- Depth First Search - Paper by Tarjan, Robert https://users.cs.duke.edu/~reif/paper/dfs.ptime.pdf
7- Jsoup – Official website https://jsoup.org/