CARDIFF UNIVERSITY SCHOOL OF COMPUTER SCIENCE AND INFORMATICS

INITIAL PLAN

Game With A Purpose

WILLIAM COOTER
C1535277

SUPERVISOR
Irena Spasic

MODERATOR
Angelika Kimmig

CM3203 One Semester Individual Project
3rd of February 2018

This project is to develop a human-based computation system to perform a task that computers cannot by packaging the task as a game. The task at hand is finding synonyms, antonyms and hypernyms for words retrieved from https://wordnet.princeton.edu/.

To play the game, a player will log in to the system, and then be matched with a random partner whose identity will remain a secret. Once matched, both players will both be shown the same word and will enter possible answers until a common answer has been entered by both players. The game will last for two and a half minutes, with a maximum of 15 words being shown. If a player wishes to skip a word, they can do. The other player is notified, and also given the option to pass, at which point they must accept to move on to the next word. A single player version of the game will also be developed using the relationships stored in WordNet to test someone's knowledge of the English Language.

The project can be broken up into 7 main parts. These are the login system, the matchmaking system, choosing the 15 words for the game, the two players agreeing on an answer, the processing of that answer, the ability to pass a word, and finally the development of a single player version of the game. Below are these in more detail.

The first task would be to build a login system. This would require players to successfully sign-up or sign-in before they can play the game. A player should then stay logged in until they log out or until their browser session ends. Basic functionality such as password hashing and session IDs are essential. Other features such as confirmation emails, forgotten password emails, "not a robot" checks, etc. may be out of my current ability, but are worth considering if there is time. This will allow other features to be developed much further down the line such as a progress tracker, an experience based matchmaking system, leaderboards, profiles, etc. I plan to store user credentials on an Amazon Web Services database. The sign-up page must check that an email address is valid, have criteria for a strong password, and normalise/sanitize inputs into string fields such as username.

The second task will be to create a matchmaking system, where players who select multiplayer will be added to a queue, and then randomly matched with another player in the queue. This will likely require the use of a server, of which I have no experience, so this may be hard to do. I might use a Raspberry Pi to act as a server and hence a game coordinator. This will not be appropriate for actual deployment as it will not be able to handle large amounts of traffic. However, it should be practical for development. The baseline features for this matchmaking system should be that there are only 2 players per game, if a player leaves the game then the game is ended, and players can only queue for and be in one game at a time. Other features that could be implemented further down the line include predicted waiting times, number of other players in the queue, number of other players online, and a matchmaking system that doesn't match a player with their previous opponent unless there are no other players online.

The third task will be selecting 15 words from https://wordnet.princeton.edu/ for a game. If the words are chosen randomly then this will be simple. However, if the "difficulty" of the word were to be considered then it could take more time. Word difficulty could be judge on the word's length, the number of times that word has been skipped in previous games, or the average number of answers that are given for this word before a matching answer is found. These will be questions I ask the client in a meeting at some point in the next couple of weeks.

The fourth task is taking inputted answers from both players and detecting a matching answer when both players have given the same synonym, antonym or hypernym. This will also likely require the user of a server to receive the answers and process them. Due to the words being manually inputted, there may be need of a spell checker to suggest a word similar to one typed if it is not in the dictionary (via a “did you mean…?”). There will also have to be normalisation and sanitization of inputs. The game itself could have several different play styles, depending on the type of answer required:

- Three seperate game modes for synonyms, antonyms and hypernyms.
- One game mode where a player can give any of the three associations.
- One game mode where the system chooses which association the players are finding per round?

There could also be a “definition” button to show the meaning of a word, though this would have to be carefully designed as to not give away any answers.

The fifth task is processing the agreed upon answer to create a link between a word from wordnet and it’s synonyms/antonyms/hypernyms. If the link is to be binary (there is or there is not an association), then it is simple. However, if a degree of association is required, then I will have to develop an algorithm to implement. A few examples of these are:

- Frequency analysis - how often does that answer come up for that word?
- Pass analysis - how often is a word passed?
- Attempts analysis - how many other answers tried before a match?
- Time analysis - how long did it take to find matching answers?

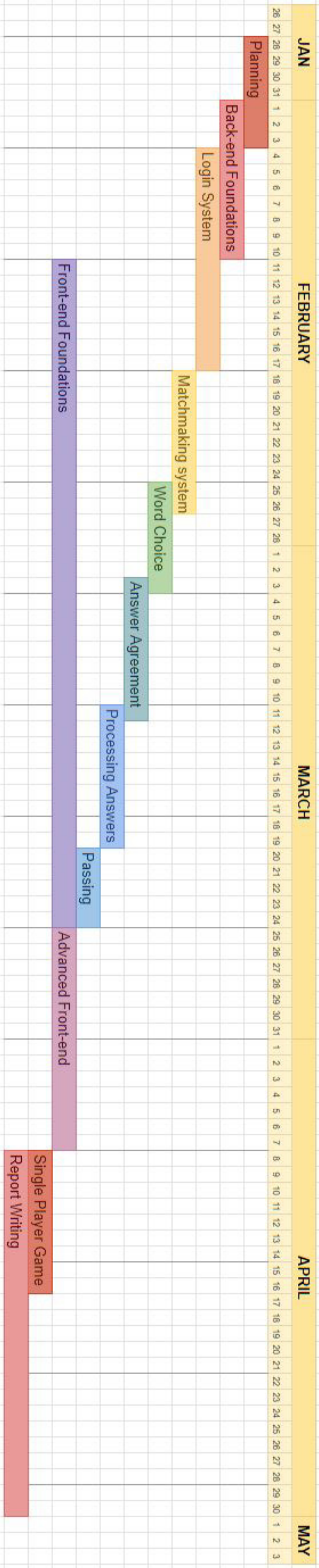
For frequency and pass analysis, the word will have to come up in multiple games, and as there are too many words in the english language to leave this up to chance, I will have to make sure that a word is seen multiple times by different players. Again, these are options that I shall present to my client and allow them to choose which is their prefered method.

The sixth task is allowing players to skip a word. If a player (player 1) skips a word, the other player (player 2) should then be notified that the player 1 has passed, and then player 2 has to also pass to move on to the next word. As the server will already be acting as a game coordinator, this will not be a lot to add. I will have to make sure that a pass is recorded as this could be useful data for determining the degree of association between a word and a synonym/antonym/hypernym.

The final task is to create a single player version of the gamen. Players would see words retrieved from WordNet, and then type in answers until an a valid answer from the associations on WordNet is found. As a lot of the processes from developing the multiplayer version would be repeated in the single player game, it would be hard to measure how far through the development phase I am at any one time. However, despite it being less computationally demanding than the multiplayer version, in the time limit I have for this project I doubt that I will be able to complete a single player version. I therefore plan to make my code for the multiplayer game as reusable and understandable as possible, so that another developer could pick up where I left off.

All 7 sections of the project require front-end and back-end development. I plan to focus primarily on the back-end of each section as I believe that functionality should take precedence over looks. As I do this, I will have to implement some very basic front-end features such as input fields for the login system. Once I have completed the login system and the multiplayer game, I will turn my focus on to developing the front end. If there is time remaining after this, I may begin development of the single player version of the game. However, I do not expect to have time for this.

The Gantt Chart to the right shows my estimated progress throughout the project. I estimate that the back-end will take around 7 weeks to complete, leaving 2 weeks for developing the front-end, and then the remainder of April to focus on my report. I have left the single player version of the game in the Gantt chart just encase I am ahead of schedule and have time to begin developing it.



For this project, I expect to require the following tools:

- A relational database
- A backend server
- Backend libraries and APIs
- Frontend libraries and APIs
- A WordNet API key
- A frontend language (e.g. HTML)
- A backend language (e.g. PHP)
- A styling language (e.g. CSS)

As I will be handling users' data such as email addresses through the log in system, and because the project requires players to play the game (and hence collect data from them), I have completed the Cardiff University Research and Integrity online training program. I may also have to consider terms and conditions to be agreed upon account creation. However, as a computer scientist I would not be expected to know the legalities, so as long as I create the ability to have a terms and conditions agreement on the sign-up page, then I feel that is enough.