2016

ARTIFICIAL INTELLIGENCE: FREE WORD ASSOCIATION

GROUP PROJECT, CMP2089M: GRP 13

A. PARKER, D. SMITH, G. ONOUFRIOU, J. TRIGG, J. LIOU, T. SUN,

CONTENTS

ABSTRACT

Artificial Intelligence.

What is Artificial Intelligence, and how do we distinguish it from traditional, organic, intelligence. This, at heart, is our core topic of research in this project. We, six members, decided to tackle this problem, using an old psychology technique. Free word association. (Jung, 1910)

We were tasked, with passing the Turing test, or a type thereof. How we chose to tackle this problem, and creating its necessary criteria, was left up to us.

In this report, we critically assess our methods, throughout, detailed in each relevant section, summarising them, later.

INTRODUCTION

IN OUR PROPOSAL

We proposed, that, the most applicable way, for us to pass the Turing test, was to create a, free word association, AI. Our reasoning can be summarized in the following:

- Since our AI, represents an entity pretending to be a human, it needs to be able to interact with as many people as possible. Free word association, is a simple concept that most can understand, abbreviated to, answer as fast as possible, the first word that, occurs in thought. (Jung, 1910)
- We have limited experience in creating AIs. And we are tasked with passing the Turing test. We believed we would have greater success, with a simpler version of natural language processing, limited to one word. Free word association however, still allows us to portray distinct thought patterns. It also allows us to, more easily implement personas.
- Personas, were a key feature of our proposal. The University of Reading (2014), had many personified AIs. We chose to follow the example of Eugene Goostman, who passed the Turing test, against 30 experts. According to their documentation, the persona played a large role in the believability of the AI, instead of just being a generic human. The AI had its own, unique quirks. Since we had a limited time period, to implement such advanced features, we would need to, use a more simplified version of the Turing test. Free word association was perfect for this role.
- Free word association, has been used extensively in the past, for patients, to project their thoughts and emotions, in a manner that, can be easily extracted. (Jung, 1910)

We examined the literary evidence to provide us with the base we could then build upon. We then set out a time plan, using Gantt charts, of how long we expected each of the aspects of our AI were going to take.

IN THE MAKING

As our project developed, we learnt that some areas, took longer and others, too shorter amounts of time, than expected. We did assess in our risk matrix that this was a likely scenario, as we had at the time of starting, not had experience in creating such systems, so the timelines were best estimates. This will impact us in the future, as we now better understand, the complexities of both AIs and of working in teams, on (relatively), large projects. This means we will be able to, more accurately estimate, time spans, which are required to do certain tasks. Even with this difference of expectation, we created an AI system, which can meet many of our initial, and further developed criteria.

CRITERIA

In our project proposal, our initial fulfilment criteria (subsequently goals) were:

1. Model AI composition (Hanheide, 2016)
    a. Knowledge Base
    b. Inference Engine
    c. Interface
2. "Full set" of English words
3. Relations between words
    a. Weighted
4. Persona Capabilities
5. Convincing >30% of users
    a. Believable response
    b. Delays
    c. Mistakes

As our project developed we tweaked the above criteria of success. Primarily the "softer" criterion (2, 4 & 5). The main causes, reasons, and results, are summarised as follows:

- Criterion 2; "Full set" of English words. We discussed in length about getting the information necessary for a "full set" of English words. We felt that to represent every, single, English word that could possibly be written, and its connections to every other word, was both unreasonable and unnecessary. Therefore needing to be changed, from our initial plans.
    - The subsequent adjustments:
        - Getting a smaller, reliable base data set. We reasoned that creating a system that could get any word from the internet, and interpret it, in a meaningful and reliable manner was unreasonable. We decided to go with a set, tested database which we could use as our reliable starting point.
    - The results:
        - What we found as a result astounded not only ourselves, but all manners of people who learned of this amazing resource. The database we discovered had several different facets, appendixes, ranging from advanced statistical testing of word relations, to idiosyncratic responses. (Nelson et al. 1998)
        - We had more than enough data to satisfy our second criterion, this data also advanced our third criterion to success. The relations between words, so explicitly tested that we as a result of this small change to mind set, and a stroke of luck to discover this ancient gem, completed two criterion in one.
- Criterion 4; we reasoned that simply creating a persona capable AI was not representative enough of what we wanted to achieve. We felt that not only does the AI need to be believable in terms of being human but arguably, just as importantly, it had to have a convincing personality. Just like Eugene Goostman. (Reading, 2014)
    - The subsequent adjustments:
        - Added more factors into persona word choices; literacy, daily experience relation chains. We would use our custom database management system, with several calls, to find if any words from what the user said and what the persona would be most likely to relate, overlap. This means we could give higher likelihood to words expected for this persona.
    - The results:

        - This could have been the most inventive, and crowning believability factor adjustment. But at the time of writing this report only a basic version has been implemented due to time constraints. But we look forwards to the final implementation, ready for the demonstration.
        - This basic adjustment, has made a large difference for some words likelihood, impressing many passing users, who have tried our system. The more advanced version will hopefully improve on the amazing work that has already been done to implement this feature. Serving as its groundwork.
- Criterion 5; we found through testing, sometimes, for more recent words that we would occasionally not have the directly relevant data. This hampered the believability of the program, as the best solution to the problem, in the original state, was to use the most generic words in the database. We deemed this to be ineffective, if used repeatedly. We needed to add convincing responses, which change over time, to more successfully convince users.
    - The subsequent adjustments:
        - A clever use of our AI's memory. If for example the user responds to the AI with a word it has not seen before, it knows that, the user's word is somehow related to its last word. It uses its memory, and instead looks up its own last word. Then uses that last word as its association, while removing the word the user just gave, from its possible responses.
    - The results:
        - This gave us a now complete and bullet proof system. We now had the dynamic where, if the first word the user gave was unknown, that it could use generic terms. But as the game progresses the AI learns from both its past and the user's responses to give the most, believable responses possible. Speeding up, just as we humans did, as the game is underway. Based off of our relatively small, but in depth data set.

SUMMARY OF CRITERIA JUSTIFICATION

Throughout the process, of creating our AI system and algorithms, we critically assessed our methods. Improving our system together. Our criteria were not perfect upon proposal. But now with the adjustments of our criteria, and assumptions, we have a better understanding of both, what our aims are, and how we have achieved those aims. We would have liked to have undertaken, fully fledged, user testing. This was however, likely an unrealistic expectation, due to everyone's individual commitments, but would have been plausible, if considered early on in the project.

WHERE WE SUCCEEDED IN MEETING OUR CRITERIA:

- (Criterion: 1) We have undoubtedly, succeeded in creating a model AI system.
- (Criterion: 2) We have a "full set" of English words, taking into consideration that, a full set may not necessarily be, a (full) dictionary of words, but instead, enough words to allow our AI to be, believable. Just as humans do not know every word, and its meaning.
- (Criterion: 3) We have a highly effective, sophisticated, dataset. We have manipulated our base data, and abstracted, the useful information. We have successfully created, a clever weighting system, which combined with our randomness factors, leads to realistic and sometimes quirky responses.
- (Criterion: 4) We have created, a persona based system, that changes according to several distinct factors. Things such as literary ability is taken into consideration.

WHAT WE WOULD NEED TO IMPROVE, TO BETTER MEET OUR CRITERIA:

- (Criterion: 5) User studies to better quantify our results, especially with passing 30% success mark.

DESIGN & REASONING

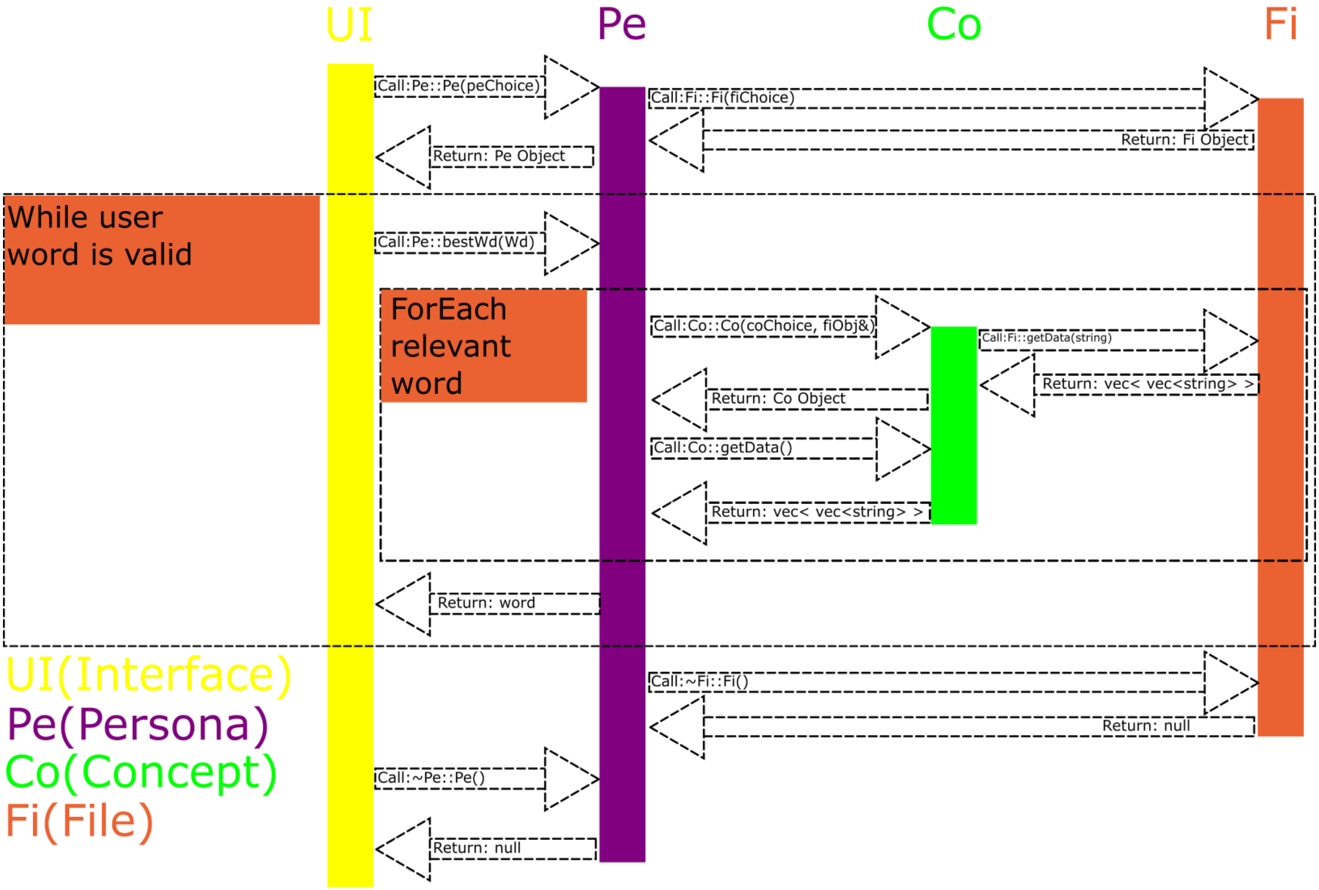


Figure 1: Initial Database Architecture

ARCHITECTURE

Since, the group decided to use Object Orientated Programming (OOP) languages, we abstracted the system into objects. Through abstraction we found the main components of the system needed to be:

- User Interface - Needed interface for the user, to better assess the AI
- Persona (human factors) - Needed to personify responses and chain (as proposal)
- Concept (individual concepts) - Needed to frame individual ideas for use (OOP)
- File (files) - Needed to manage file data to be more accessible

Based, off of the above necessary components, we linked them together in a manner, which would not require processor intensive, reconstruction of the objects, regularly. The exception is Concept, as multiple instances of Concept would be necessary. Examples of multiple Concepts are, when given a user word that a second word/ concept is pulled to compare. This includes comparing against Persona relations; Bob, is a persona. Bob is a builder. The database is searched for “builder” the concept. The results are compared against the results from the user's word.

ABSTRACT DATA TYPE SPECIFICATIONS

We initially intended to use ADTs, but we found problems in that, not all members were familiar with them. We did begin solving this problem, by helping members understand ADTs, but instead we opted for class diagrams, as they were something the group was already comfortable with. This did also allow for the individual programmers to be creative, and subsequently be more enthusiastic about each section, as the specifics of function semantics is left up to the programmers. This lack of a formal definition, did have its own drawbacks.

THE MAIN DRAWBACKS OF NOT USING ADTS ARE:

- Others are less familiar with the intricacies of each individual function, and what each one is meant to do. This can lead to some confusion.
- More difficult to debug and test, as you do not have the specification to refer to, if something seems out of place.
- More difficult to enforce, good programming practices, such as minimal, global scoping, since there were less restrictions, without formal definitions.

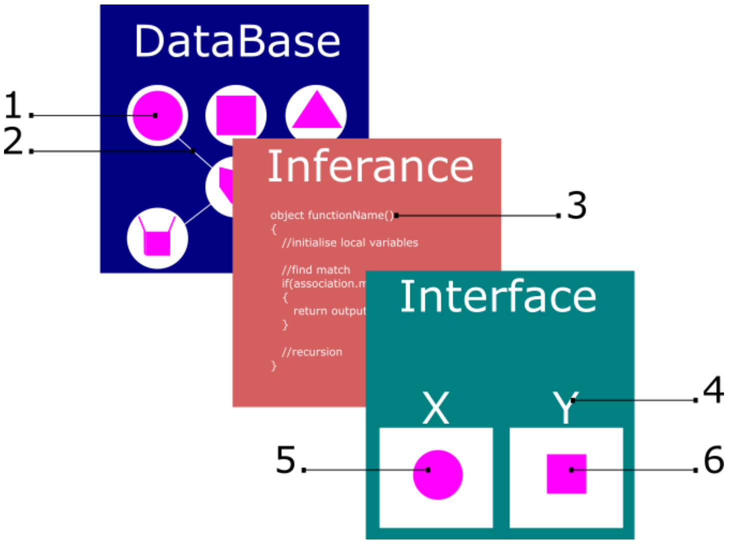
THE MAIN ADVANTAGES WE GAINED FROM USING DIAGRAMS:

- Not as temporarily expensive, getting members to define ADTs, and ensuring their definitions are correct.
- Feeling of freedom, and achievement of creating something, rather than implementing someone else's code.
- No need to spend time ensuring, members understand ADT specifications.
- Quickly, go into the productive stage, to produce the artefact that we have a restricted time to make, and write about.

USER INTERFACE

PLANS

We initially, planned in an abstract manner. Which we developed into a fit for purpose, user interface.



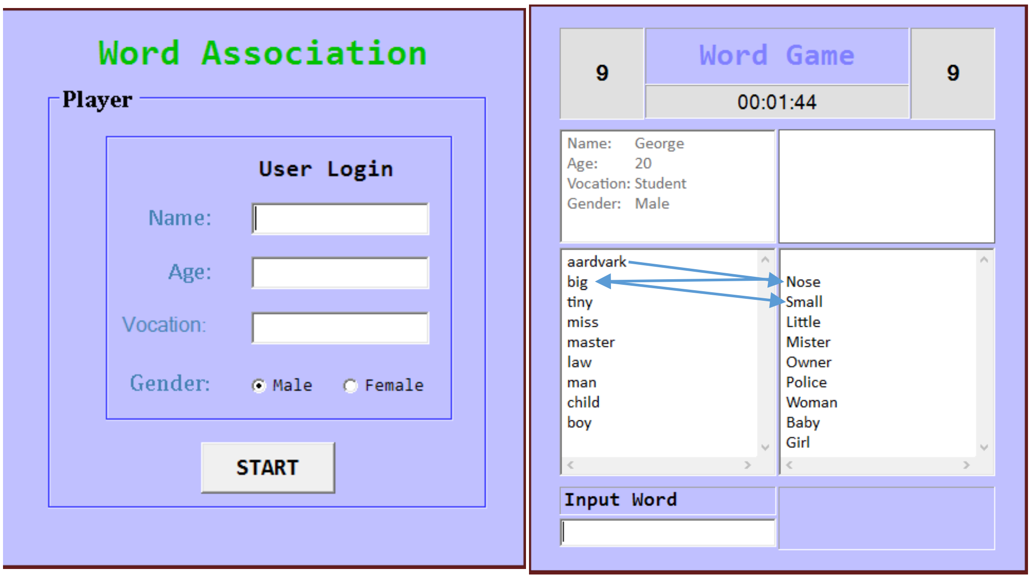
```
object functionName()
{
  //initialise local variables

  //find match
  if(association.m
  {
    return output
  }

  //recursion
}
```

Figure 2: Initial abstract plans, of the AI system.

Figure 2 shows how we abstracted the components. 4 being the two personas X (Player), and Y (AI). 5 & 6 represent responses, such as square when given circle. This functioned as the base, which we then developed.

Since we had already planned in Figure 1, to link the Interface using set functions, we had no need to factor this in to this preliminary design of the interface.

CURRENT



Word Association

Player

User Login

Name:

Age:

Vocation:

Gender: ● Male ○ Female

START

9 Word Game 9

00:01:44

Name: George
Age: 20
Vocation: Student
Gender: Male

aardvark
big
tiny
miss
master
law
man
child
boy

Nose
Small
Little
Mister
Owner
Police
Woman
Baby
Girl

Input Word

Figure 3: Current User Interface.

The team developed the initial abstract design, into Figure 3 above. Please note that, this is it in the current state. But we do have plans to develop this further before the demonstration date. Especially in regards to the second persona displaying, and adding certain game conditions for draws.

Figure 3 shows the prominent persona layout, to be at the very head, so as to be instantly seen. We have a cascading reply system, so it zig zags between responses. Flowing better on the page.

SUCCESSES

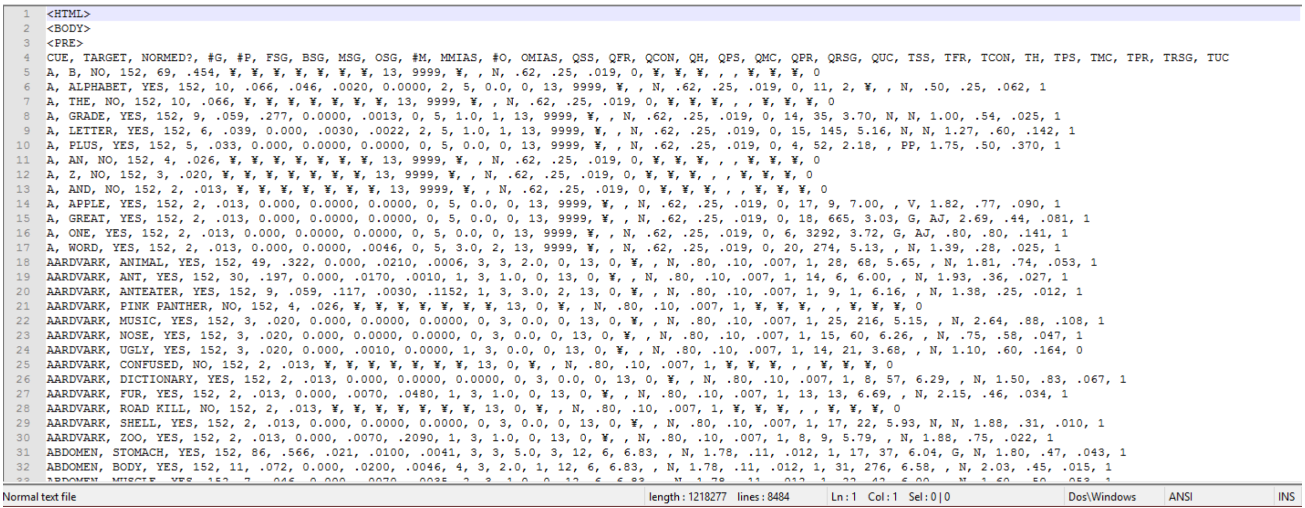
- The user interface, contains all the necessary information, to begin to pressure users. The timer, functions to keep people replying, as fast as possible. Improving the overall game, as less time is given to input esoteric terminology.
- The user interface, sanitises user input, especially on the log in page, preventing important details from being missing, and as correct as possible. This ensures that the user can compare, their attributes, to those of the persona they are playing against, after the final UI changes before the demonstration. Although please note Figure 3, does not how opponents persona currently.
- Lists are adjustable, so as to allow the game to carry on, in particularly quick situations.
- Errorless UI system, which will always elegantly exit the application.
- Relative response delays, to make the responses believable, harder responses taking longer.

IMPROVEMENTS

- Finding ways to make the page less, cognitively expensive. To emphasize the important details, such as time remaining, to further user pressure, and speed.
- Backdoor to access free word association game immediately, during development.
- Pressing enter on log in page, should be bound to clicking the start menu button for user ease.

DATA

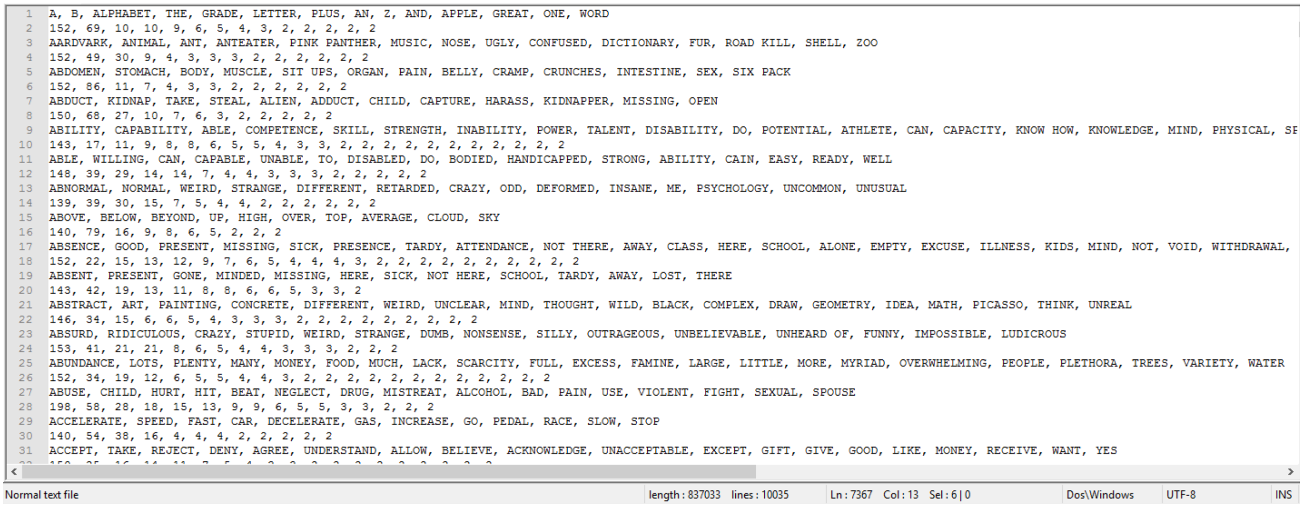
We began with a very convoluted and inconsistent data format. The data that was within was invaluable to us, but the formatting and consistency of it was a draw back. To solve this problem we developed our own data set, which was far more concise and applicable for our needs.



```
<HTML>
<BODY>
<PRE>
CUE, TARGET, NORMED?, #G, #P, FSG, BSG, MSG, OSG, #M, MMIAS, #O, OMIAS, QSS, QFR, QCON, QH, QPS, QMC, QPR, QRSG, QUC, TSS, TFR, TCON, TH, TPS, TMC, TPR, TRSG, TUC
A, B, NO, 152, 69, .454, ¥, ¥, ¥, ¥, ¥, ¥, ¥, 13, 9999, ¥, , N, .62, .25, .019, 0, ¥, ¥, ¥, , , ¥, ¥, ¥, 0
A, ALPHABET, YES, 152, 10, .066, .046, .0020, 0.0000, 2, 5, 0.0, 0, 13, 9999, ¥, , N, .62, .25, .019, 0, 11, 2, ¥, , N, .50, .25, .062, 1
A, THE, NO, 152, 10, .066, ¥, ¥, ¥, ¥, ¥, ¥, ¥, 13, 9999, ¥, , N, .62, .25, .019, 0, ¥, ¥, ¥, , , ¥, ¥, ¥, 0
A, GRADE, YES, 152, 9, .059, .277, 0.0000, .0013, 0, 5, 1.0, 1, 13, 9999, ¥, , N, .62, .25, .019, 0, 14, 35, 3.70, N, N, 1.00, .54, .025, 1
A, LETTER, YES, 152, 6, .039, 0.000, .0030, .0022, 2, 5, 1.0, 1, 13, 9999, ¥, , N, .62, .25, .019, 0, 15, 145, 5.16, N, N, 1.27, .60, .142, 1
A, PLUS, YES, 152, 5, .033, 0.000, 0.0000, 0.0000, 0, 5, 0.0, 0, 13, 9999, ¥, , N, .62, .25, .019, 0, 4, 52, 2.18, , PP, 1.75, .50, .370, 1
A, AN, NO, 152, 4, .026, ¥, ¥, ¥, ¥, ¥, ¥, ¥, 13, 9999, ¥, , N, .62, .25, .019, 0, ¥, ¥, ¥, , , ¥, ¥, ¥, 0
A, Z, NO, 152, 3, .020, ¥, ¥, ¥, ¥, ¥, ¥, ¥, 13, 9999, ¥, , N, .62, .25, .019, 0, ¥, ¥, ¥, , , ¥, ¥, ¥, 0
A, AND, NO, 152, 2, .013, ¥, ¥, ¥, ¥, ¥, ¥, ¥, 13, 9999, ¥, , N, .62, .25, .019, 0, ¥, ¥, ¥, , , ¥, ¥, ¥, 0
A, APPLE, YES, 152, 2, .013, 0.000, 0.0000, 0.0000, 0, 5, 0.0, 0, 13, 9999, ¥, , N, .62, .25, .019, 0, 17, 9, 7.00, , V, 1.82, .77, .090, 1
A, GREAT, YES, 152, 2, .013, 0.000, 0.0000, 0.0000, 0, 5, 0.0, 0, 13, 9999, ¥, , N, .62, .25, .019, 0, 18, 665, 3.03, G, AJ, 2.69, .44, .081, 1
A, ONE, YES, 152, 2, .013, 0.000, 0.0000, 0.0000, 0, 5, 0.0, 0, 13, 9999, ¥, , N, .62, .25, .019, 0, 6, 3292, 3.72, G, AJ, .80, .80, .141, 1
A, WORD, YES, 152, 2, .013, 0.000, 0.0000, .0046, 0, 5, 3.0, 2, 13, 9999, ¥, , N, .62, .25, .019, 0, 20, 274, 5.13, , N, 1.39, .28, .025, 1
AARDVARK, ANIMAL, YES, 152, 49, .322, 0.000, .0210, .0006, 3, 3, 2.0, 0, 13, 0, ¥, , N, .80, .10, .007, 1, 28, 68, 5.65, , N, 1.81, .74, .053, 1
AARDVARK, ANT, YES, 152, 30, .197, 0.000, .0170, .0010, 1, 3, 1.0, 0, 13, 0, ¥, , N, .80, .10, .007, 1, 14, 6, 6.00, , N, 1.93, .36, .027, 1
AARDVARK, ANTEATER, YES, 152, 9, .059, .117, .0030, .1152, 1, 3, 3.0, 2, 13, 0, ¥, , N, .80, .10, .007, 1, 9, 1, 6.16, , N, 1.38, .25, .012, 1
AARDVARK, PINK PANTHER, NO, 152, 4, .026, ¥, ¥, ¥, ¥, ¥, ¥, ¥, 13, 0, ¥, , N, .80, .10, .007, 1, ¥, ¥, ¥, , , ¥, ¥, ¥, 0
AARDVARK, MUSIC, YES, 152, 3, .020, 0.000, 0.0000, 0.0000, 0, 3, 0.0, 0, 13, 0, ¥, , N, .80, .10, .007, 1, 25, 216, 5.15, , N, 2.64, .88, .108, 1
AARDVARK, NOSE, YES, 152, 3, .020, 0.000, 0.0000, 0.0000, 0, 3, 0.0, 0, 13, 0, ¥, , N, .80, .10, .007, 1, 15, 60, 6.26, , N, .75, .58, .047, 1
AARDVARK, UGLY, YES, 152, 3, .020, 0.000, .0010, 0.0000, 1, 3, 0.0, 0, 13, 0, ¥, , N, .80, .10, .007, 1, 14, 21, 3.68, , N, 1.10, .60, .164, 0
AARDVARK, CONFUSED, NO, 152, 2, .013, ¥, ¥, ¥, ¥, ¥, ¥, ¥, 13, 0, ¥, , N, .80, .10, .007, 1, ¥, ¥, ¥, , , ¥, ¥, ¥, 0
AARDVARK, DICTIONARY, YES, 152, 2, .013, 0.000, 0.0000, 0.0000, 0, 3, 0.0, 0, 13, 0, ¥, , N, .80, .10, .007, 1, 8, 57, 6.29, , N, 1.50, .83, .067, 1
AARDVARK, FUR, YES, 152, 2, .013, 0.000, .0070, .0480, 1, 3, 1.0, 0, 13, 0, ¥, , N, .80, .10, .007, 1, 13, 13, 6.69, , N, 2.15, .46, .034, 1
AARDVARK, ROAD KILL, NO, 152, 2, .013, ¥, ¥, ¥, ¥, ¥, ¥, ¥, 13, 0, ¥, , N, .80, .10, .007, 1, ¥, ¥, ¥, , , ¥, ¥, ¥, 0
AARDVARK, SHELL, YES, 152, 2, .013, 0.000, 0.0000, 0.0000, 0, 3, 0.0, 0, 13, 0, ¥, , N, .80, .10, .007, 1, 17, 22, 5.93, N, N, 1.88, .31, .010, 1
AARDVARK, ZOO, YES, 152, 2, .013, 0.000, .0070, .2090, 1, 3, 1.0, 0, 13, 0, ¥, , N, .80, .10, .007, 1, 8, 9, 5.79, , N, 1.88, .75, .022, 1
ABDOMEN, STOMACH, YES, 152, 86, .566, .021, .0100, .0041, 3, 3, 5.0, 3, 12, 6, 6.83, , N, 1.78, .11, .012, 1, 17, 37, 6.04, G, N, 1.80, .47, .043, 1
ABDOMEN, BODY, YES, 152, 11, .072, 0.000, .0200, .0046, 4, 3, 2.0, 1, 12, 6, 6.83, , N, 1.78, .11, .012, 1, 31, 276, 6.58, , N, 2.03, .45, .015, 1
```

Figure 4: Original Data Set. (Nelson Et Al, 1998)

Figure 4 shows how the initial data set, had peculiarities, and anomalies that would later need to be removed, to ensure the data remained useful to us.



```
A, B, ALPHABET, THE, GRADE, LETTER, PLUS, AN, Z, AND, APPLE, GREAT, ONE, WORD
152, 69, 10, 10, 9, 6, 5, 4, 3, 2, 2, 2, 2, 2
AARDVARK, ANIMAL, ANT, ANTEATER, PINK PANTHER, MUSIC, NOSE, UGLY, CONFUSED, DICTIONARY, FUR, ROAD KILL, SHELL, ZOO
152, 49, 30, 9, 4, 3, 3, 3, 2, 2, 2, 2, 2, 2
ABDOMEN, STOMACH, BODY, MUSCLE, SIT UPS, ORGAN, PAIN, BELLY, CRAMP, CRUNCHES, INTESTINE, SEX, SIX PACK
152, 86, 11, 7, 4, 3, 3, 2, 2, 2, 2, 2, 2
ABDUCT, KIDNAP, TAKE, STEAL, ALIEN, ADDUCT, CHILD, CAPTURE, HARASS, KIDNAPPER, MISSING, OPEN
150, 68, 27, 10, 7, 6, 3, 2, 2, 2, 2, 2
ABILITY, CAPABILITY, ABLE, COMPETENCE, SKILL, STRENGTH, INABILITY, POWER, TALENT, DISABILITY, DO, POTENTIAL, ATHLETE, CAN, CAPACITY, KNOW HOW, KNOWLEDGE, MIND, PHYSICAL, SP
143, 17, 11, 9, 8, 8, 6, 5, 5, 4, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
ABLE, WILLING, CAN, CAPABLE, UNABLE, TO, DISABLED, DO, BODIED, HANDICAPPED, STRONG, ABILITY, CAIN, EASY, READY, WELL
148, 39, 29, 14, 14, 7, 4, 4, 3, 3, 3, 2, 2, 2, 2, 2
ABNORMAL, NORMAL, WEIRD, STRANGE, DIFFERENT, RETARDED, CRAZY, ODD, DEFORMED, INSANE, ME, PSYCHOLOGY, UNCOMMON, UNUSUAL
139, 39, 30, 15, 7, 5, 4, 4, 2, 2, 2, 2, 2, 2
ABOVE, BELOW, BEYOND, UP, HIGH, OVER, TOP, AVERAGE, CLOUD, SKY
140, 79, 16, 9, 8, 6, 5, 2, 2, 2
ABSENCE, GOOD, PRESENT, MISSING, SICK, PRESENCE, TARDY, ATTENDANCE, NOT THERE, AWAY, CLASS, HERE, SCHOOL, ALONE, EMPTY, EXCUSE, ILLNESS, KIDS, MIND, NOT, VOID, WITHDRAWAL,
152, 22, 15, 13, 12, 9, 7, 6, 5, 4, 4, 4, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
ABSENT, PRESENT, GONE, MINDED, MISSING, HERE, SICK, NOT HERE, SCHOOL, TARDY, AWAY, LOST, THERE
143, 42, 19, 13, 11, 8, 8, 6, 6, 5, 3, 3, 2
ABSTRACT, ART, PAINTING, CONCRETE, DIFFERENT, WEIRD, UNCLEAR, MIND, THOUGHT, WILD, BLACK, COMPLEX, DRAW, GEOMETRY, IDEA, MATH, PICASSO, THINK, UNREAL
146, 34, 15, 6, 6, 5, 4, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2
ABSURD, RIDICULOUS, CRAZY, STUPID, WEIRD, STRANGE, DUMB, NONSENSE, SILLY, OUTRAGEOUS, UNBELIEVABLE, UNHEARD OF, FUNNY, IMPOSSIBLE, LUDICROUS
153, 41, 21, 21, 8, 6, 5, 4, 4, 3, 3, 3, 2, 2, 2
ABUNDANCE, LOTS, PLENTY, MANY, MONEY, FOOD, MUCH, LACK, SCARCITY, FULL, EXCESS, FAMINE, LARGE, LITTLE, MORE, MYRIAD, OVERWHELMING, PEOPLE, PLETHORA, TREES, VARIETY, WATER
152, 34, 19, 12, 6, 5, 5, 4, 4, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
ABUSE, CHILD, HURT, HIT, BEAT, NEGLECT, DRUG, MISTREAT, ALCOHOL, BAD, PAIN, USE, VIOLENT, FIGHT, SEXUAL, SPOUSE
198, 58, 28, 18, 15, 13, 9, 9, 6, 5, 5, 3, 3, 2, 2, 2
ACCELERATE, SPEED, FAST, CAR, DECELERATE, GAS, INCREASE, GO, PEDAL, RACE, SLOW, STOP
140, 54, 38, 16, 4, 4, 4, 2, 2, 2, 2, 2
ACCEPT, TAKE, REJECT, DENY, AGREE, UNDERSTAND, ALLOW, BELIEVE, ACKNOWLEDGE, UNACCEPTABLE, EXCEPT, GIFT, GIVE, GOOD, LIKE, MONEY, RECEIVE, WANT, YES
```

Figure 5: Manipulated Data Set.

Figure 5 shows the resultant data set, modelled after the Google hash code challenge data files. This format is far more consistent and concise than the original dataset. We cut down the data from all 7 files or roughly 1 million+ lines each, into a single file of 837,033 lines. This directly decreased the amount of memory required to hold the same amount of data, and it lead to a more efficient read/ write process. Putting less strain on whatever machine is running the application.

SUMMARY OF DESIGN

Critically looking back, it would have been preferential to have used ADTs. They allow for more coherent design, and easier testing for any bugs that are created.

In future, assuming that most members are comfortable with ADT specifications, we would use them. In our circumstances, due to the lack of familiarity with ADTs at the time, the two choices were either, sit down as a group and go through, making sure everyone's knowledge is up to date. Or we could use class and other such diagrams which the group is already comfortable with. We did chose the latter, but where possible, ADTs are preferable.

The program overall is efficient. It completes every operation in a timely manner and with no errors. It could have been optimised in certain places, including multi-threading, to make the operation smoother and faster. But, for the most part, there were no flaws in the design that needed, amendment.

The User interface functions effectively, portraying all the information that is necessary, in a useful manner. Always exiting elegantly and dynamic enough to handle both large words, and long lists.

The planned, and subsequently executed changes to the data, helped incredibly. It reduced the strain on the host computer massively, and reduced the time taken to retrieve a result, so that more time could be used to manipulate the results in a human manner.

The group worked well, with a well-orchestrated plan to link all the individual sections together using a predefined interface (Figure 1). There were some issues with certain empty returns, for say words that do not exist in the database, but these were dealt with early on. This was solved primarily because of an initially limited G Appendix (main data file), which forced the program to run clean, with empty sequences.

TOOLS AND METHODOLOGIES

From the first meeting, the need for communication was identified as key for a successful project. We decided on the use of the University Email system, as our main line of official communication. With the Emails, the team could easily communicate with each other, as either a whole, or as individuals, enabling the sharing of important updates and issues. This was particularly helpful during the Easter break, with the lack of group meetings due to members of the team returning home; The Email system allowed for clarifications and deadlines for smaller targets rather than relying on a large target set for the end of Easter. Furthermore, these deadlines could easily be altered for any changing circumstances, and unforeseen issues that developed over the break.

The need for a less formal route of communication was also called for; Where Emails would be used to arrange meetings and share key information, we also used WhatsApp for short term reminders and brief updates. This allowed group members to communicate information too brief for an Email with WhatsApp, and too in depth for a message with Email. The format of WhatsApp allowed for a more back-and-forth approach to communication, speeding up smaller discussions like questioning which room a meeting was in, which can be quickly answered through the App.

The development of the Artefact itself began with the creation of a GitHub repository. We decided that using Git for version control would be very beneficial, enabling us to make changes to code, without the fear of creating bugs and risking the long term stability of the project. Any damaging change could be reverted back, and the use of GitHub to store our code enabled the whole group to work on the project, without interfering with the others. Furthermore, the recording of individual commits let the group see exactly what changes had been made, and when features had been added. GitHub notably helped during the development of the Artefact over Easter, enabling the team to view other members' up to date code, and troubleshoot issues with how the classes interacted. Once the classes were mostly complete, GitHub let the team make changes to a file simultaneously, speeding up debugging the code, and the interactions between classes.

GitHub also aided communication, allowing the team to leave comments in each other's code, as well as providing a Wiki for the project. At the beginning, the Wiki was filled with the basic outline for each section of the project, with a specification for each class. As the database was added, the wiki was updated to detail the format of the Appendix Files, and information was added to help the team understand the other members' code.

The Code itself was developed in C++, using Visual Studio 2015. C++ was picked as several members of the team were familiar with the language, as well as being commonly used for projects involving databases. C++ is also an efficient language, making it a good choice for projects dealing with large data sets (Baldwin, 2011). Originally, the UI was being developed in C#, due to the UI team being more familiar with the language. Once they were more this changed to C++ once the team members felt more confident in C++ to avoid needing to link multiple languages. Visual Studio natively supports C++, as well as being available to the whole team in the labs and at home through DreamSpark. VS is also the IDE most familiar with the team, making it ideal for everyone to use for the project.

One issue with using Visual Studio was the amount of generated binary files present in the folders being version controlled by Git. This resulted in some issues at the start with pushing commits to the remote repository on GitHub. As the group became more comfortable with git, we created a git-ignore, removing the frustration of binaries.

Google Drive was also used by the team in order to collaborate writing the report, enabling everyone to work on their sections in the same manner that GitHub allowed simultaneous coding. Drive let everyone access the most up to date version of the report, and also let everyone work on their sections individually, while removing the need to manually compile sections, and enabled everyone to add relevant information.

SUMMARY LIST OF TOOLS

Google drive	(word source control)
Visual Studio	(IDE)
Git	(source control)
GitHub	(online repository/ remote)
Microsoft Office Word	(word processor)
365 Email	(main and official emails)
WhatsApp	(quick communication)

CRITICAL REFLECTION:

HONEST APPRAISAL

*Please refer to the individual sections for more detailed reflections.

TEAMWORK

WHAT WE DID WELL

- Team wide, consistent input, methods. Including group discussions, and making sure that in these discussions, every member could and would input their ideas.
- Effective communication arrangements. We had two ways to communicate to everyone, Email and WhatsApp, both for different purposes and response times, for all scenarios.
- Good utilisation of team members' strengths. Every member worked on sections that were most applicable to their strengths. A good example is how the foreign students, were focused on programming things like the user interface, but they were aided by the rest of the team in the more English, text based areas.
- A set schedule/ routine, which members could rely on. We had a set meeting timetable, with only a few voluntary exceptions, that were not attendance marked as a result. Everyone was consistently emailed about the topics in the meetings, and if a quorum was not met, then the meeting would be adjourned.
- Non-restrictive system, that allowed everyone to volunteer for sections, that they preferred. Every member chose the sections they worked in, based on the pool of available sections. This meant that no one felt, pressured into a section they would not like to do, and that there was no quarrelling as a result.

WHAT WE COULD HAVE DONE BETTER

- Stricter internal deadlines, to increase the likelihood of work completion on an individual basis.
- More rigorous GitHub control. To force git, pull requesting, and protecting the master branch, just in case someone did something wrong by accident, like deleting the master branch.
- Enforcing git compiling and erroring. So that git would not allow broken or uncompileable code to enter the remote repository. This way we could have saved time, as there were some delays due to erratic code fragments, on a few occasions.

ARTEFACT

WHAT WE DID WELL

- Extensively tested, errorless code. That always exits as expected.
- Reasonably low strain application, that works as expected.
- Good responses in a timely manner, which can portray personification.
- Achieved most of our criteria, and more. Although more evidence is required to achieve criterion 5 (see criteria).
- Can and will always respond with a word, which is based off of one of our personas.
- Randomly selects a persona for the player to play with.
- Occasionally quirky responses based off of a likelihood mechanism.

WHAT WE COULD HAVE DONE BETTER

- Program backdoor, for development purposes. This would significantly reduce the time needed to test the application, and avoid any frustration as to having to retype the credentials in every time.
- Less cognitively expensive interface, which can be immediately comfortable. Allowing a better, feel of flow, of the program.

BELBIN ROLE UTILISATION

BELBIN RESULTS:

Damon	Coordinator, Shaper*
Jake	Plant, Team Worker*
George	Plant, Specialist
Alex	Monitor Evaluator, Plant*
Jichan	Implementer, Team Worker*
Tao	Monitor Evaluator, Implementer*

*Belbin test taken post team, formation.

TEAM ROLES

*Please refer to Group Members section, for individual details.

UTILISING THE BELBIN RESULTS

Unfortunately, many of the group members did not complete the initial Belbin task, before being allocated into a group. The only known Belbin result was George, as a Primary plant, secondary specialist. This however, did not hinder the team and the roles that individuals filled. We as a group decided to retake the Belbin test, so that we could refer to any findings here.

What we found was that collectively, the members filtered into their preferred roles naturally. Upon closer inspection of the team roles, a clear connection between the members' self-perception (Belbin), and the roles they filled, was found. Despite the lack of possible roles in the team, each member leaned towards, how they perceived themselves.

Instead of any formal mechanism to assign Belbin roles in the group, the group formed roles naturally. No team member was forced into a role they didn't like (see above). There were remarkably few disputes, if any, within the group. We all became aware, and subsequently compensated for each other's strengths and weaknesses.

This suggests that knowing your Belbin results, has little impact on the role you eventually fill. As you will fill the role you perceive to fit in, given the choice.

CLOSING STATEMENT

We worked well as a team, to produce an interesting artefact, which we can all be proud of. Every team has its weaknesses, but we believe, we worked effectively despite these weaknesses. I wish them all the best.

GROUP MEMBERS

GEORGE ONOUFRIOU

Roles:

- Database management Design (Appendix G, File.h & File.cpp)
- Report (Design, Architecture, Criteria & report formatting)
- Programming (File.h & File.cpp)
- Data Gathering (Appendix A-H)
- Co-ordinator

Description:

George, functioned as the plant and co-ordinator for sessions and general organiser of the group. This included making sure that everyone contributed. He was also the architect of the overall system.

Grade: 100% (A), Attendance: 100%. StudentID: ONO14475496 Belbin: Plant, Specialist

JICHAN LIU (KEN)

Roles:

- Programmed UI (C#)
- Finding relevant resources
- Research (Linking C# to C++)
- Code UI in C++(Gaming interface)

Description:

Ken, functioned as one of the two implementers of the team. Critically assessing situations, such as the hard decision to swap to a C++ interface, for easier implementation. He primarily worked on the backwards and forwards, game interface.

Grade: 100% (A), Attendance: 100%. StudentID: LIU13490935 Belbin: Imp, T.Worker

TAO SUN

Roles:

- Design of UI interface
- Programmed UI (C#)
- Research (Linking C# to C++)
- Code UI in C++(Home page)

Description:

Tao, functioned as one of the two implementers of the team. Communicating and pair programming with Ken, to effectively create two interfaces. He primarily worked on the home pages of the UI's and error prevention.

Grade: 100% (A), Attendance: 100%. StudentID: SUN13473462 Belbin: M.Eval, Imp

JAKE TRIGG

Roles:

- Abstracting concepts (Concept.h & Concept.cpp)
- Programming (Concept.h & concept.cpp)
- Report (Tools and Methodologies)
- Updating Wiki (Concept)
- Debugging (concept)

Description:

Jake, functioned as the monitor evaluator, making good judgements for the team, and especially for the program, like changes to make responses more human. Helping all the members of the team make good decisions while completing his own roles.

Grade: 100% (A), Attendance: 90%, StudentID: TRI14468975 Belbin: Plant, T.Worker

ALEX PARKER

Roles:

- Abstracting personas (Persona.h & Persona.cpp)
- Programming (Persona.h & Persona.cpp)
- Report (proof-reading &)
- Debugging (Persona)

Description:

Alex, functioned as the shaper, working well under pressure. Overcoming all the problems thrown at him. He went out of his way to learn the new skills necessary to tackle his initial difficulties with C++.

Grade: 100% (A), Attendance: 60%, StudentID: SMI13458446 Belbin: M.Eval, Plant

DAMON SMITH

Roles:

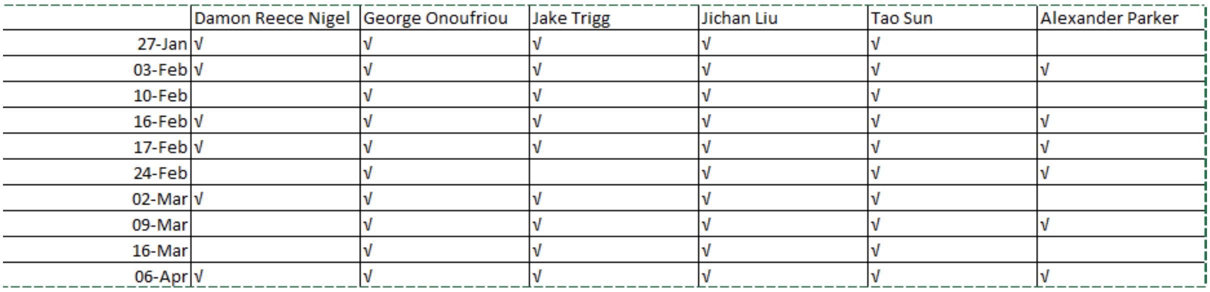
- Report (Drafting; Honest appraisal, Passing the Turing test & Critical Reflection)
- Updating Design of UI interface
- Updating Wiki for UI

Description:

Damon, functioned as the team worker, helping the team to gel. He was especially critical to writing, draft sections of the report. He was also key to getting the foreign students to mix with the team, initially.

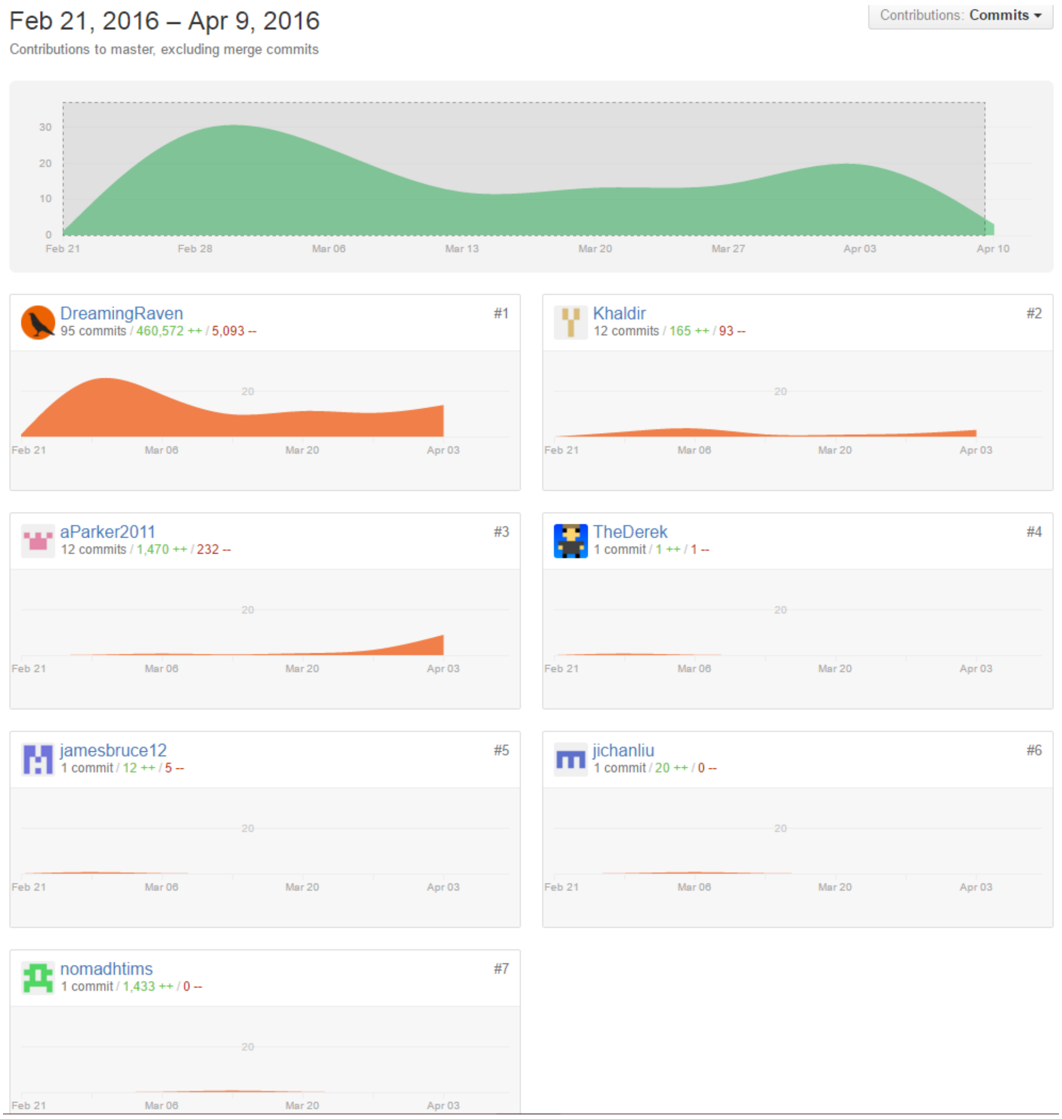
Grade: 100% (A), Attendance: 60%, StudentID: SMI13458446. Belbin: Co-ord, Shaper

MISC RECORDS



	Damon Reece Nigel	George Onoufriou	Jake Trigg	Jichan Liu	Tao Sun	Alexander Parker
27-Jan	√	√	√	√	√	
03-Feb	√	√	√	√	√	√
10-Feb		√	√	√	√	
16-Feb	√	√	√	√	√	√
17-Feb	√	√	√	√	√	√
24-Feb		√		√	√	√
02-Mar	√	√	√	√	√	
09-Mar		√	√	√	√	√
16-Mar		√	√	√	√	
06-Apr	√	√	√	√	√	√

Figure 4 (above): Attendance record, post instructor meeting.



Feb 21, 2016 – Apr 9, 2016

Contributions to master, excluding merge commits

Contributions: Commits

DreamingRaven #1
95 commits / 460,572 ++ / 5,093 --

Khaldir #2
12 commits / 165 ++ / 93 --

aParker2011 #3
12 commits / 1,470 ++ / 232 --

TheDerek #4
1 commit / 1 ++ / 1 --

jamesbruce12 #5
1 commit / 12 ++ / 5 --

jichanliu #6
1 commit / 20 ++ / 0 --

nomadhtims #7
1 commit / 1,433 ++ / 0 --

Figure 5 (above): Commit frequency record. (GitHub 2016)

BIBLIOGRAPHY

Baldwin, B. (2016, April 13). *Why is C++ so fast*. Retrieved from Lingpipe: https://lingpipe-blog.com/2011/07/01/why-is-c-so-fast/

GitHub. (2016, April 13). *Word-Association*. Retrieved from GitHub.com: https://github.com/DreamingRaven/Word-Association

Hanheide, M. (2016, February 13). *Knowledge Representation*. Retrieved from Blackboard: https://blackboard.lincoln.ac.uk/bbcswebdav/pid-1045624-dt-content-rid-2313639_2/courses/CMP2020M-1516/CMP2020M%2002%20Knowledge%20Representation%281%29.pdf

Jung, C. G. (1910). The Association Method. *Journal of Psychology, I*(31), 219-269.

Nelson, D., McEvoy, C., & Schreiber, T. (2016, April 1). *Free Association*. Retrieved from Univeristy of South Florida: http://w3.usf.edu/FreeAssociation/

University of Reading. (2016, February 14). *News and Events*. Retrieved from Univerity of Reading: http://www.reading.ac.uk/news-and-events/releases/PR583836.aspx