# MapMyNotes

Final Report for CS39440 Major Project

*Author:* Ryan Gouldsmith (ryg1@aber.ac.uk)
*Supervisor:* Dr. Hannah Dee (hmd1@aber.ac.uk)

4th March 2016
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in
Computer Science (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

# Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name .....

Date .....

# Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name .....

Date .....

# Acknowledgements

I am grateful to...

I’d like to thank...

# Abstract

Include an abstract for your project. This should be no more than 300 words.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

# Background & Objectives

## 1.1 Background

Handwriting notes is still considered to be an important aspect of note taking. Smoker et al [7] conducted a study comparing handwritten text against digital text for memory retention and out of 61 adults 72.1% prefered to take notes using pen and paper, rather than on a computer. Smoker et al concluded that recall rates for handwritten text was greater than that of typed text proving that handwritten notes are better for a user's memory retention.

However technology has advanced and we're moving into an era where we track and view everything digitally, from email to calendar entries. Therefore, there is a need to transfer the productivity from handwritten notes to digital notes - so they can be located easily.

When a user creates a note they will often come in differing forms: some are semi-structured and some are back of the evenvelope kind of notes. When thinking about an application to analyse notes, first there has to be consideration for what a note will consist of. A taxonomy, by definition, is a biological term for a classification of similar sections, showing how things are linked together.

Notes can be throught of as a collection of similar classifications, whether this is the pure textual descriptions of a note or whether this is purely pictoral form or a mixture of both. However, the notes are normally split into three distinct categories:

1. Textual descriptions
2. Diagrams
3. Graphs

In figure 1.1, it shows a taxonomy of the different aspects which may comprise to make a note. Textural descriptions form the core part of a note, this is essentially the important content that a note-taker is trying to remember and write down. Different note-takers form their notes in different ways, for example the headings may be underlined or hashed - if they were adopting a mark-down style approach. These sections helps to show that there's a break in the content, and should be sub-sectioned. Textural points that are short, but important, are often characterised by a colon or a bullet point; these are the most common form of concise note building, in the classisifcation. Coloured highlighting if often used for a variety of reasons: it stands out on the page and for some

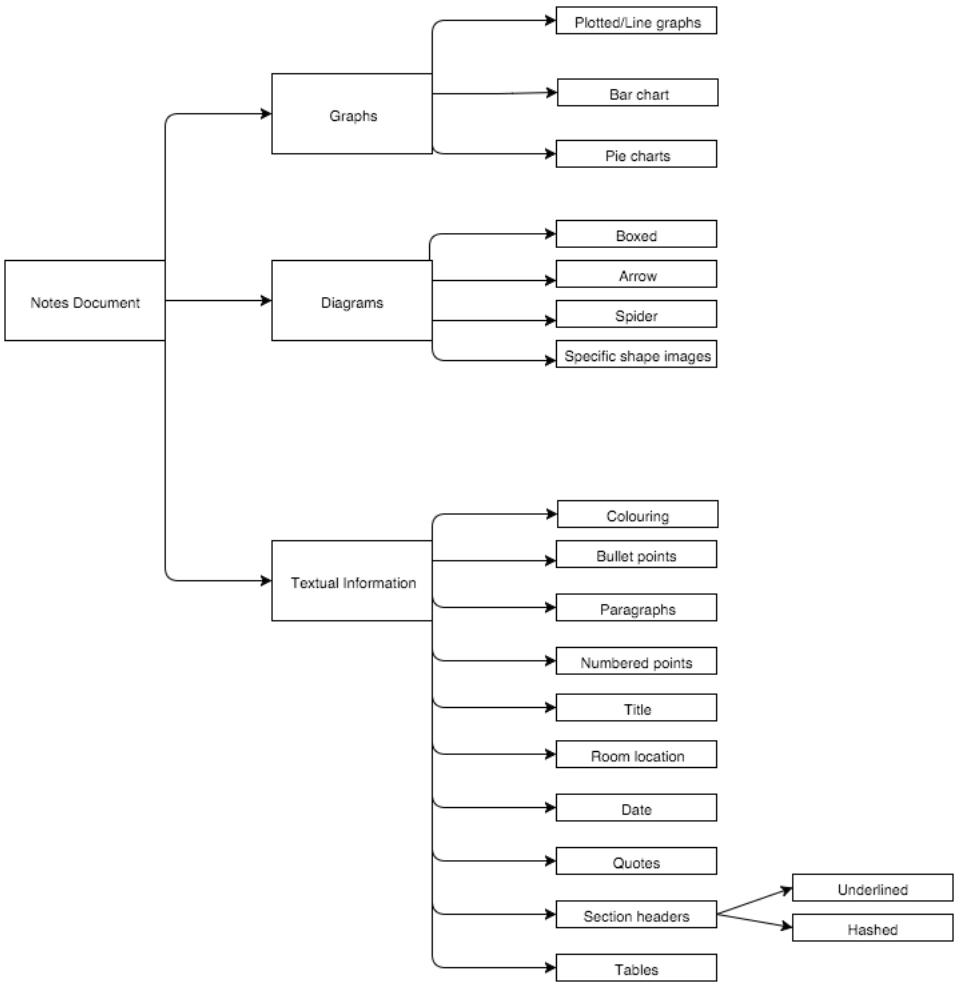


Figure 1.1: A taxonomy showing the structure and classification of different types of notes and what is contained in a note.

people colour retention is better. Finally, tables help to represent textual content in tabular form. This is often good in notes for comparisons.

Graphs are great visual tools for user's in helping to convery textual information in a graphical form. Naturally, they have their limitations such as they come in different shapes and sizes, such as a line-graph, pie chart or a bar chart. Coupled with graphs, notes often consist of diagram drawings. In figure 1.1, there are different sections and classsifications of a diagram: boxed, arrow etc. Each one has its own purpose and arrowed and box can overlap; UML diagrams are a case of this. Spider diagrams are probably the hardest to represent, due to the varying sizes and whether the user draws circles or clouds. Furthermore, specific shape diagrams are conceptutally hard to think about as it depends on the domain in which the user is drawing the note. For example, a person in biology may draw a stick person, whereas someone in Computer Science may draw a PC.

Overall, identifying the taxonomy of notes is important as it gives context to which the application can potentially parse and identify these different classifications. Additionally, it acknowledges where there may need to be constraints due to the variations in notes and how it can deal

with semi-structured documents.

### 1.1.1 Similar Systems

There are a number of systems which offer similar, and sometimes the same functionality that MapMyNotesApplication would use. The systems are

- OneNote
- EverNote

Each of these tools offer something slightly different and it would be good to produce a system that would encapse these different aspects.

#### 1.1.1.1 OneNote

OneNote is a note-taking and organisational app made by Microsoft [cite].

#### 1.1.1.2 EverNote

EverNote is a note-taking and note organisation app, it is both supported on the web and in app form. EverNote is widely used and would provide a lot of the functionality a user would need to upload their note. They have realeased development articles [CITE-2013 one] stating that they are able to do OCR recognition on images. This would allow the user to upload an image and it would give a list of words, which it thinks is the word identified. This differs from MapMyNotes where the author aims to develop an application which would give a 1-1 word comparision, rather than a list of words.

### 1.1.2 Why this project?

The author's motivation for this project is to provide a tool, which initially would be tailored to lecturer based note content. Often notes are written up, but discarded into a big pile and later they're hard to find. As a user, it would be good if the notes were all in one place that they could photograph in and it would automatically create meta-data for it and save it to their calendar. Often the modules that are undertaken are stored in my calendar as a reminder that there's a lecturer on that day - so providing a link to the event in the calendar would be a nice way to tie all these dispersed information together.

## 1.2 Analysis

By analysing the problem a taxonomy of notes was collated. This helped to breakdown exactly what was in a set of notes. This taxonomy gave a comprehensive analysis of what information would be available to parse from the test, as well as giving the information parsed semantic meaning. It was quickly brought up that I should potentially create a specific layout for the note so that the data can be parsed sensibly, to reduce the complexity.

After identifying that handwriting recognition would be a part of the application then research began investigating what OCR (optical character recognition) tools are available. It was suggested that the open source tool Tesseract [cite] would be an appropriate tool. Instead of using an OCR tool then I could have focussed my dissertation on analysing notes and extracting user's handwriting from it - however, that is not within the aim of the project. As it is more of a tool to use, then using an OCR eases the complexity.

The main objective of the tool would allow a user to upload a photo of their handwritten notes. It will use very basic OCR analysis of meta-data such as the date, lecturer, module code, lecture title and location from the user's handwriting. For the very basic application the handwriting would analyse the author's handwriting. The application will be able to add and edit meta-data relating to a note they upload to the web application. The user will then be able to search for a given module code, they would then be expected to view their notes that they have made. Futhermore, they would be expected to be able to view all the notes that they have entered into the application.

With the web application, it is expected that the user will be able to integrate with their calendar. Google Calendar has been chosen for the calendar the application will interact with. This would have to interact via oAuth. The calendar would be integrated when the user views the homepage to see all their notes, and when they save the note it will update a calendar event of the given day - adding the note URL.

Ideally, there would be more features available in the release but due to the time constraints then certain functionality was not in the primary goals. Analysing the whole note and converting it all to text and displaying the appropriate colour from the note would have been a nice feature to have and would have been really useful. However, this is far too complex within the timeframe that was available. Instead the parsing of the meta-data would be considered enough instead of this additional feature.

During the first few meetings my supervisor, Dr Hannah Dee, it was discussed what was actually plausable in the time frame. Acknowledging that the project was large and could be expanded we opted to rein in the features and get a minimal project. Dr Hannah Dee suggested that the just the application without the Tesseract parsing would be enough for a minimal product, however I wanted the ability to try and recognise handwriting recognition. So we comprised but mentioned that the handwriting recognition would be a background process, and not the main aim.

- Looked into different ways to binarise an image accordingly.
- Created a list of user stories after the discussion with hannah to decompose the problems.
- I chose to do OCR recognition for singular pages and hopefully getting th etop 3 lines as meta-data than normal notes because I needed structure.

## 1.3 Process

Due to the flexibility and the unknown nature of certain aspects of the project, such as how long handwriting training and the calendar integration would take, then a normal plan-driven approach like the waterfall approach would not be suitable. Therefore, due to the changing nature of the project it was agreed that an Agile Metholodoly approach was to be followed.

The chosen Agile Methodolgy was Scrum. This was coupled with principles from Extreme Programming, such as: merciless refactoring and continuous integration. In a Scrum approach work is split into sprints, which define what should be done that week. These normally consist of user stories such as: *As a user I want to be able to search for a given module, so that I can find all notes for that module*. This story is used as a reminder for some feature value. The story is then split into tasks which act as acceptance criteria for the story - this acceptance criteria is used to evaluate the story once it's completed to make sure that it's fully completed.

Once the acceptance critera has been specified I estimate the complexity of that task, in comparision to a "goldilock" task. For my planning I adpated the planning poker technique - where the task goes from 1, 2, 3, 5, 8 and so on. Normally, if I got to the 15 mark then I felt the story was too large and needed to be reviewed and broke down, and the process would start again.

At the end of the sprint a review and restrospected was conducted, this was in the form of a blog post instead of a group, and the primary aims of this was to analyse performance and how I could improve over the next sprint. The retrospectives were key to highlighting issues as well as reflecting on what went well, so I can improve the process and the work produced.

Choosing story points per sprint was not just pot luck or what the customer, in this case the supervisor, wanted the most. At the end of every sprint a tally was kept of how many story points were completed, this would then be brought forward to the next sprint as how many story points should be completed that week. For example, if you completed 20 story points one week your next weeks estimation would be 20 story points worth.

- Taiga.io

In addition to Scrum the process has been modified to incorporate Extreme Programming principles. They involved test-driven-development, so the tests were written first to give me a better understanding of the underlying design. This coupled with the constant mericless refactoring allowed me to have confidence in the system I was building would still pass the appropriate tests. Furthermore, continuous integration tools were used on the project. TravisCI was used [CITE] and although it's normally for teams, it was good for the build automation and to ensure code was being plaed in the repository. Finally, CRC were used appropriately when thinking about how the classes interacted with one another - this was another way to help and think about the design of the application.

# Chapter 2

# Image pre-processing for Tesseract

## 2.1 ImageMagick

In order to produce the best possible results with Tesseract, there had to be a pre-processing stage beforehand. Initially, the image was converted to grayscale using ImageMagick[CITE] but this showed that training was difficult and was not yielding the success rates which was needed.

Pursuing using ImageMagick for the pre-processing stage the image was converted to Monochrome. This managed to be able to pick up more characters in the image, but in doing so picked up too much noise around the image. This distorted the characters and made it hard to recognise characters with the correct confidence that Tesseract required.

Dr Hannah Dee suggested that maybe OpenCV[cite] image manipulation would be useful for the pre-processing step, instead of using ImageMagick.

## 2.2 OpenCV

OpenCV is an open source image processing library. There was a choice of using the C++ or the Python bindings. Due to the applications core infrastructure it was agreed that Python would be easier to implement.

An investigation began by looking into the OTSU binarisation method and other versions of the adaptive thresholds. The need for a binarisation method, instead of using the default image, is that it reduced the ambiguity of the light sources on the image and any shadows which may occur during the capturing of the image.

After doing some investigations into different thresholding algorithms it came to light that Tesseract OCR uses OTSU thresholding as its underlying image processing algorithm. This quickly explained why a lot of characters were not being identified from images, when looking at the output of the experiment that were conducted. From this it could be seen that using OTSU as a pre-processing step for Tesseract would not return the best results.

Looking at the experiment of different thresholding techniques it was clear that adaptive mean and adaptive gaussian returned the image clearly binarised. There is no shadow or dark patch over the image and the text on the page is legible.

## 2.3 Line removal

After using Tesseract to train my handwriting on plain paper it showed that some sentences it could parse accurately, but those which are on a slant or slightly skewed, because of the lack of lines, returned a worse recognition rates.

To try and straighten the lines of text normal lined paper was used, but quickly found this was not good when binarising it as it often left unconnected lines which were represented as a series of dots. If the erode function was used on the image, it would lose the quality of the characters correctly identified.

This led to a creation of my own lined paper, one which would have thick lines and one which the lines could be filterable. The premise of using the blue lined paper was to correctly identify the blue lines and remove them from the image, leaving uniform lines of text which could be analysed. This process went through a couple of iterations.

To begin with it was decided to just extract the text from the image which fell between a black to gray threshold colour - excluding the blue lines from the image. This was followed up by the morphological operation, erode, to remove some of the remaining pixels left over by the lines. This worked to an extent and showed the binarised image with some loss of pixels on the characters and some were completely undistinguishable.

Although this worked well, it wasn't the most suitable solution as often line pixels were left in the image and would be picked up by the OCR resulting in an erroneous output from the system. So, again, it went through another iteration.

The next solution would binarise the image more elegantly and would actually remove the lines from the image and clear up the image to just leave the image binarised, with no blue or black lines.

[CITE REFERENCE USED TO DO THIS] Firstly, it would read the image as grayscale and then apply a median blur to the image. This was needed to try and ease the noise and smooth the image. Much like the first iteration adaptive threshold with a gaussian mean was used to binarise the image. Afterwards, a mask was collected containing the horizontal black lines, using the structuring elements "MORPH_RECT". This was eroded and dilated to try and remove the lines as much as possible. Intermediate masks so that black text was passed onto a new canvas - aiming to get as much of the text from the images across as a possible. This naturally carried some of the disjointed lines across, and suffered the same issues the previous iteration had to deal with. However, this iteration includes the use of finding contours.

The contours were considered using because a way was needed to find connected components and filter out the lines. Naturally, if the lines have been dilated and eroded then they will not be connected closely. Therefore, by choosing connected components on the image it will correctly get the characters from the image. Once the connected components of the characters were identified these were transferred to a blank mask, with a final erosion to clear up the image.

This technique has worked well in removing the blue lines from the page, whilst keeping the character quality. It clearly shows a binarised image with the text on the page clearly identified, and it worked on multiple examples. What is more impressing is that because of the adaptive threshold and different morphological operations then it can clearly identify characters in terribly lit photos.

# Chapter 3

# Handwriting training with Tesseract

[CITE THAT GUYS THESIS]

Handwriting recognition is still an active research project and one which is constantly evolving. There are many complexities with handwriting such as whether to analyse cursive or non-cursive text. The project could have been taken in a couple of direction: write my own handwriting recognition system or use an OCR tool. Since the premise of the application is to provide a software tool for a user creating my own handwriting recognition system is not viable. As a result, an OCR tool has been used.

Consideration went into the different OCR tools out there, with commercial vs non-commercial and there was one open source technology which seemed very reliable, Tesseract.[CITE PAPER ON COMPARISON].

Tesseract is an open source C++ library for analysing handwriting, the current stable version at the time of writing the report is 3.04. It is mainly interacted through a command line application, and that has what has been used in this project.

## 3.1 Training

To begin to analyse the characters on a page, there was a decision that the training process would involve analysing non-cursive handwriting. Although this limits the user experience, analysing handwriting is a challenging task in itself, so by making it non-cursive there was a better chance of good recognition.

Secondly, another caveat with training is that the training process would be consisted on the author's handwriting, and not on the general public. This again, limits the application to the user, but that could be expanded in the future.

In order to successfully, train the handwriting then the pre-processing steps described in chapter 2 are required. Outputted from the image pre-processing is a tiff file - which then used for the Tesseract training.

Firstly, the image has to be in the following format: ¡language¿.¡font¿.exp¡expnumber¿.tiff, for example the following file would be valid: eng.ryan.exp1a.tiff.

# Chapter 4

# Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

## 4.1 Overall Architecture

## 4.2 Some detailed design

### 4.2.1 Even more detail

## 4.3 User Interface

## 4.4 Other relevant sections

# Chapter 5

# Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

# Chapter 6

# Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Have you tested your system on real users? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

## 6.1 Overall Approach to Testing

## 6.2 Automated Testing

### 6.2.1 Unit Tests

### 6.2.2 User Interface Testing

### 6.2.3 Stress Testing

### 6.2.4 Other types of testing

## 6.3 Integration Testing

## 6.4 User Testing

# Chapter 7

# Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

# Appendices

# Appendix A

# Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

Apache POI library The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the clients existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [2]. The library is released using the Apache License [1]. This library was used without modification.

# Appendix B

# Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

# Appendix C

# Code Examples

## 3.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [6].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*-------------------------------------------------*/
  /* Minimum Standard Random Number Generator        */
  /* Taken from Numerical recipies in C              */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity   */
  /* Always call with negative number to initialise  */
  /*-------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
```

```
static long iy=0;
static long iv[NTAB];
double temp;

if (*idum <=0)
{
  if (-(*idum) < 1)
  {
    *idum = 1;
  }else
  {
    *idum = -(*idum);
  }
  idum2=(*idum);
  for (j=NTAB+7;j>=0;j--)
  {
    k = (*idum)/IQ1;
    *idum = IA1 *(*idum-k*IQ1) - IR1*k;
    if (*idum < 0)
    {
      *idum += IM1;
    }
    if (j < NTAB)
    {
      iv[j] = *idum;
    }
  }
  iy = iv[0];
}
k = (*idum)/IQ1;
*idum = IA1*(*idum-k*IQ1) - IR1*k;
if (*idum < 0)
{
  *idum += IM1;
}
k = (idum2)/IQ2;
idum2 = IA2*(idum2-k*IQ2) - IR2*k;
if (idum2 < 0)
{
  idum2 += IM2;
}
j = iy/NDIV;
iy=iv[j] - idum2;
iv[j] = *idum;
if (iy < 1)
{
  iy += IMM1;
}
```

```
   if ((temp=AM*iy) > RNMX)
   {
     return RNMX;
   }else
   {
     return temp;
   }
 }
```

# Annotated Bibliography


[1] Apache Software Foundation, "Apache License, Version 2.0," http://www.apache.org/licenses/LICENSE-2.0, 2004.

This is my annotation. I should add in a description here.

[2] ——, "Apache POI - the Java API for Microsoft Documents," http://poi.apache.org, 2014.

This is my annotation. I should add in a description here.

[3] H. M. Dee and D. C. Hogg, "Navigational strategies in behaviour modelling," *Artificial Intelligence*, vol. 173(2), pp. 329–342, 2009.

This is my annotation.s I should add in a description here.

[4] S. Duckworth, "A picture of a kitten at Hellifield Peel," http://www.geograph.org.uk/photo/640959, 2007, copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

This is my annotation. I should add in a description here.

[5] M. Neal, J. Feyereisl, R. Rascunà, and X. Wang, "Don't touch me, I'm fine: Robot autonomy using an artificial innate immune system," in *Proceedings of the 5th International Conference on Artificial Immune Systems*. Springer, 2006, pp. 349–361.

This paper...

[6] W. Press *et al.*, *Numerical recipes in C*. Cambridge University Press Cambridge, 1992, pp. 349–361.

This is my annotation. I can add in comments that are in **bold** and *italics and then other content.*

[7] T. J. Smoker, C. E. Murphy, and A. K. Rockwell, "Comparing Memory for Handwriting versus Typing," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 53, no. 22, pp. 1744–1747, Oct. 2009. [Online]. Available: http://dx.doi.org/10.1177/154193120905302218

Used to show that there handwriting is still an important part of memory rentention with note taking, compared to digital text

[8] Various, "Fail blog," http://www.failblog.org/, Aug. 2011, accessed August 2011.

This is my annotation. I should add in a description here.