Mining Online Food Recipes

Model Development for Ingredient Based Recipe Rating Prediction

Final Report for CS39440 Major Project

Author: Charlotte Taylor (cht26@aber.ac.uk)

Supervisor: Dr. Chuan Lu (cul@aber.ac.uk)

4th May 2016

Version: 1.0 (Release)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (inc Integrated Industrial and Professional Training) (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 Charlotte Taylor

Date 3rd May 2016

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 Charlotte Taylor

Date 3rd May 2016

Acknowledgements

I am grateful to Dr. Chuan Lu for her support, inspiration and guidance throughout this project.

I'd like to thank my parents and John Colgrave for their support and encouragement whilst work was carried out on this project. They helped me dedicate a lot of time to the development and research that was required and without them that would not have been possible.

Abstract

Big Data is rapidly becoming one of the biggest sectors within all businesses [27]. Companies are realizing the potential of analysing data to give them an edge on competition through more informed decisions in the business world. Harnessing the power of this data, making it useful and being able to predict the outcome has become a hot commodity and research has become an extremely valuable resource. Incorporating machine learning into the decision process, allowing a machine to look over past trends and predict the outcome is extremely important for the development of Big Data technologies.

Over the past few years online recipes and being able to access quick, tasty and interesting dishes and then make them yourself has become the trend. Just looking at FaceBook there are hundreds of pages: "Proper Tasty" [18] a recipe in under a minute cooked in front of you, and "BuzzFeed Food" a site where they list recipes from all websites being just a couple of sites available. As food is an integral part of people's lives, finding new and interesting ways to enjoy it will always be in high demand.

Using machine learning on recipe data, which is easily accessible via any website, the project discussed in this document will aim to be able to predict whether a suggested recipe will be rated highly or poorly. This will be based solely on the ingredients and past experience with combinations of these. Python and the Natural Language Toolkit (NLTK) is used to gather and process the data in order to make accurate predictions.

The results show that when using the Bayesian Bernoulli algorithm, the expected accuracy of guessing the rating is 77.8%. With further development of the pre-processing of the data and a look at alternative machine learning algorithms this could be improved and become a valid prediction model.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background

1.1 Big Data & Machine Learning

Big Data has already taken the industry by storm, and is on the rise as one of the most integral parts of any business. Businesses are beginning to take the approach of: to understand your customer you need to track their every move. "The hiring market for analytics professionals and data scientists has gone into overdrive" [12] and machine learning is becoming a part of the package within these roles.

Machine Learning is a valuable insight that is created through data collection and analysis. The data is processed, used, trained and tested upon in order to create predictive models. With more research and processing, more accurate predictive models can be created. Real-world practices can be as simple as giving a company an edge in the business world to groundbreaking research into better diagnosis of patients and predictive treatment plans [28]. Machine Learning has become invaluable throughout the industry and is beginning to really take hold within all sectors of business such as: autonomous cars, fraud detection, speed recognition, facial recognition and recommendations to name a few [11].

Before this project a year in industry took place within one of the largest independent games companies in the UK (Jagex). The internship involved being a data analyst and a part of the Business Intelligence (BI) team. This team consisted of data analysts, data scientists and data warehouse administrators. Within this company alone 2TB of game data is collected every day. This data is then used to analyse every aspect of a player, from finding out where they live and what sort of income they are likely to have to finding out their play style. Using this allowed the company to target promotions specifically at these different players, make more money and also keep the players happy. Through tracking customers and finding out what they want to do before they know they want to do it via machine learning practices and data analytics, a business can become extremely successful. This project will hopefully mirror and improve upon the ideology found at Jagex.

With the field being relatively new to the spotlight, and memory being so cheap, research and data collection has boomed, not only within Jagex but across the globe. The amount of data being collected is growing exponentially, with at least 2.5 quintillion bytes of data produced every day (as of March 2015) [23], this is growing at over four times the rate of the worlds economy [32].

1.2 Online Food Recipes

With technology forever being a part of day to day life people have begun to use devices to find, learn and prepare recipes. This is a significant change from recipe books, magazine articles and pieces of paper scrawled with your Grandmas favourite recipe. This evolved from the improved recommendation, search and customization of recipes that initially became fashionable from Netflix and Amazon [15]. Through additional research with Network Analysis, recommendations became a better functioning system and additional features such as sharing, bookmarking and healthy alternatives, made online recipes become what it is today.

People over the age of 35 are more likely to print out a recipe but still find it online, and 59% of 25 to 34 year olds cook by referring to an electronic device [13]. Online food recipes are an integral part of YouTube, Facebook, Tumblr, Pinterest, Twitter and any major social networking site. People want to try new recipes, and when they appear on a timeline or a newsfeed they get inspired and want to make the item and share it with their friends. The digital age even allows people to Instagram their food before they eat it. There are peaks in searching for recipes based on holidays within the year as shown within figure 1.1. Therefore, it is essential for supermarkets to track these eating trends, ensuring their stock will supply these seasonal items in demand.

Figure 1.1: Recipe search trends on Google over the course of a year, averaged indexed search query volumes from the United States between January 2010 & October 2014 [26]

This industry is going to continue to grow and people will be increasingly interested in the development of technological computing behind the food industry. Food is something everyone has in common. People share their own recipes on websites in the hope to find others recipes and get kudos for sharing their amazing home-cooked recipe. More and more people are becoming connected via the internet, including third-world countries having the technology to connect and share.

This means data and a lot of data! With lots of data comes exciting prospects. This can be seen via competitions that have been created that are dedicated to computational cooking [14], this really is a new and exciting field to be a part of.

1.3 Related Work

1.3.1 Ingredient Space & Generative Probabilistic Models

Similar research projects have been carried out in the past relating to recipes. One project was undertaken by Vladimir Nedovic that used machine learning to research ingredient networks to find good ingredient combinations; the title of his work: "Learning Recipe Ingredient Space using Generative Probabilistic Models" [19]. This project mainly focused upon the way in which to process ingredient lists and recipes, particularly digital recipes. Analysing each ingredient at 'word' level, and treating ingredients individually allowed for the correlations of ingredients and recipe similarities to be revealed which in turn allowed for some sort of ingredient substitution or recipe combination. Nedovic's approach focused mainly on the initial experiments to look at the feasibility of recipe improvisation via computational methods.

The approach taken by this looks at general probabilistic models, including Latent Dirichlet Allocation (LDA) and Deep Learning.

LDA is a more general descendant of the Probabilistic Latent Semantic Analysis (PLSA) model [22]. Using one of these methods assumes parameters are drawn from prior Dirichlet distribution allowing for some reduction in overfitting; LDA uses additional smoothing parameters. It is known for its work with large volumes of data especially text via the bag-of-words methodology. Both LDA and PLSA use iterative processes through the corpus, allowing for the separation of: documents, words and topics in order to find a distribution of vocabulary over topics. Using this method assumes that the topics within a certain document, in this case recipes, will be at the very least related.

Deep Learning is coupled with LDA in the form of Deep Belief Networks (DBN). DBN uses features of Restricted Boltzmann Machines (RBM) that stack and train in layers in order to fine tune their own parameters [5]. This is a very *greedy* processing method due to the multiple training layers on each individual piece of data.

The paper shows the LDA and Deep Learning techniques on 56,000 recipes containing 361 unique ingredients taken from allrecipes.com, epicurious.com and menupan.com (two American sites and one Korean for a wider culture base). Ingredients are the main priority and only the presence or lack of presence is considered and mapped. Once ingredients are mapped with their labeled cuisine type it appears obvious that distinct groupings are formed within cuisines, seen via clustering within the mapping. The mapping, however, is biased due to three quarters of the data being from North America. After the mapping has occurred a 3-Layer DBN is used with progressively more hidden variables to adapt the network shape. This filters some of the more exotic combinations and provides a probability ingredient presence. The preliminary works show it working within ingredient combinations such asS: cakes mixed with fruits, meats and seafood mixed with herbs and vegetables. All of these have constraints that can be altered in order to investigate the more common cuisines.

Although successful preliminary results were found with the testing of this data, the biased dataset that was used to manufacture the learning algorithms was a major factor. There were also issues with approximations having to be made due to the inexact maximum likelihood approach to learning. The success, however, is encouraging with fairly accurate results; even having used LDA rather than the more accurate and better known Rigid LDA. For this to be used outside of the practices within this paper has potential.

1.3.2 Cultural Flavour Networks & Food Pairing

Another valuable project used for research within this field is "Flavor Network and the Principles of Food Pairing" [9]. The paper focuses on the diversity between dishes from different cultures, looking more in depth at the breakdown between regional recipes. It looks for patterns within combinations of ingredients and principles via in-depth flavour compound analysis.

This method is very different to the probabilistic type model, within section 1.3.1, due to its more biological approach. It explores the idea that mankind is "exploiting but a tiny fraction of the potential combinations [of ingredients]", and follows the idea of finding a way in which to reproduce the choices made for good/bad recipes. Colour, texture, sound and temperature are recognised to be important but ignored for this project due to taste being classified as the most important aspect of a meal. Certain ingredients can be used not for their flavour but as stability, textural or even for colour purposes. These ingredients are filtered out due to the large number of recipes tested upon (56,498); this leaves flavour as the main focus.

A widely known hypothesis that both chefs and food scientists agree on is that those ingredients that share certain flavour compounds are more likely to work together within a recipe. This is known as food pairing, which can lead to and can see unusual ingredient combinations; such as caviar and white chocolate. The article then goes on to create networks of flavour compounds over ingredient combination. Each ingredient averages 51 flavour compounds, these are then used within a bipartite network consisting of nodes (ingredients and compounds known to contribute to flavouring the ingredient). If nodes connect they are then weighted based on the number of shared flavour compounds. These are then extracted via a backbone extraction method to identify the statistically significant links.

Backbone extraction is an abstraction of a complex network in order to help people understand networked systems in a simplified form [24]. The extraction looks at the distribution of weighting, expectation and filtering to retrieve the normalized weight. Filtering methods start by defining statistical features within nodes and edges to determine whether it can be discarded or preserved. This allows only the integral weighted networks that carry relevant information to the entire model/network to remain.

The paper then continues by testing the links on two American websites (allrecipes.com, epicurious.com) and one Korean website (menupan.com). The recipes were then split into geographical cuisines and the popularity of ingredients were measured. It considered the individual ingredient within the recipe and calculates:

- Authenticity: the number of times an ingredient is used within all recipes of a particular cuisine
- Contribution: the degree in which the ingredient changes the shared compounds factor
- Shared Compounds: the mean number of shared compounds within a recipe, based on real and randomly constructed recipes

The results showed vast differences between location and their tendency to either retain (most Northern American recipes) or avoid (most Eastern Asian recipes) similar flavour compounds. The results showed a successful and robust model even on biased/incomplete data. This approach would be great to implement outside of this project; allowing for additional breakdown of ingredients to ensure they are analysed correctly. The backbone extraction isn't overly efficient however,

due to the need for the whole model to be analysed multiple times to find the important aspects. An alternative may be useful due to the need for an understanding on what is/isn't a significant feature within the recipe.

1.3.3 Recipe Recommendation Systems

A slightly different take on recipes, big data and machine learning is found within "Recipe Recommendation Using Ingredient Networks" [29]. This, similar to section 1.3.2, looks at networking the ingredients. Instead of the focus being on combinations within different regions, it looks at the ability to add or drop ingredients from an existing recipe and the ability to predict the rating of the recipe. A useful ability for those that can't find specific ingredients and need to either swap or ignore one or two to complete the recipe; then determining how good it will be without. Another large difference with this paper is the data parsing that occurs. Rather than looking at just the recipes themselves like those described within sections 1.3.1 and 1.3.2, it also looks at the user reviews and comments on the recipes themselves.

46,337 recipes were downloaded from allrecipes.com with all information about them saved including 1,976,920 reviews from users of the site. Multiple networks (complementary and substitution) were used for this process to work. The complementary community is split between savoury and sweet dishes, while the substitution network was derived from user-generated suggestions for modifications. Ingredients were preprocessed by removing quantities, temperature or consistencies from the ingredient list. The ingredient list was then limited to the top 1,000 (based on frequency), this accounted for 94.9% of all ingredients. The remaining ingredients were removed due to high-specificity, brand names, rarity or not being a food type.

To create the Ingredient Complement Network, the logic behind it looks at the probability that two ingredients occur together against the probability that they occur separately. This sorts itself into sections of sweet vs. savoury dishes due to the central ingredients that are a part of it. A network clustering algorithm is then applied to formulate the clusterings of ingredients.

To create the Substitute Network was more difficult, having to process the text from user reviews and be able to correlate "add" or "remove" or "instead of" to find out what is integral to a recipe. Parsing these reviews creates a network from user knowledge to identify available substitutions. This is used to find high correlations of ingredient preference that can later be used in recipe recommendation systems.

The Recipe Recommendation System looks at its Network positions firstly, to find out whether popular or uncommon ingredients were used within the recipe. This is then used to calculate the positional value based on the centrality of ingredient network. Network communities are created via the positions and the basis of whether an ingredient will likely co-occur with a group of others is established. Once established these can be used to check if ingredients can be substituted and whether it would compliment the recipe. Support Vector Machines (SVM) alongside Stochastic Gradient Boosting Tree models were used in order to consider both the whole model and the set of features at hand at the same time.

The results were disappointing, with prediction accuracy only improving slightly from 71.2% to 74.6% although some variations were found when testing the nutrition or ingredient networks individually. Network structures were found to be the most successful with a 79.2% result set. The networking and deeper look at parsing the data within this paper is shown to be critical, and the networking is again shown to have deep importance.

Chapter 2

Analysis & Relevant Techniques

2.1 Project Aims

The aim of this project is to help a person to predict their level of enjoyment of a recipe from analysing all available data. This will be viable from the creation of a program that will run from start to finish and provide insight into the rating of recipes from a website. This should be able to run from having absolutely no data, to gather the data itself, process the data, allow a user to enter their own recipe and then output the estimated rating.

Previous research within the field of online recipes incorporating machine learning techniques, have been seen to look more at regional suggestions. The purpose of this project is to look at how recipes and their ingredients are perceived from a human point of view. This will tackle what ingredient combinations make for a good recipe, versus those that don't work well together. There has been some insight into this already, however this looked at it in a more biological way by picking out food pairings and the compounds found within ingredients.

The rationale of this project is that some characteristics of food change the way cooks perceive the food; be it the look or the texture alongside the flavour it holds. This is more of a psychological reaction to the food rather than assuming it is all relative to taste. This project will therefore look at the user ratings and the ingredients found within them. In order to do this various approaches will been considered and implemented. This will result in the development of a piece of software that will allow a user to test their own recipes to see what sort of rating they would be likely to receive on social media or recipe websites.

2.2 Technology Overview

2.2.1 Programming Language

There are many languages that can be used within the collection, processing and final analysis of data. The two programming languages most apppropriate for the analysis of data are R and Python. These programming languages have more in-depth and up to date libraries for data management.

R is open-source software [17]. This source-code is readily available, modifiable and has been reviewed by coders from all over the globe for the past 15 years. This has made the accuracy of this

programming language extremely high. The open-source aspect has also created a large community of users that have been a part of the creation and maintenance of the language, which provides a lot of help with the resolution of issues leading to tutorials helping users. The programming language was created for statisticians, data scientists and analysts, and has helped to provide these people with good quality and numerically accurate data visualization and manipulation, predictive modeling and statistical analysis tools.

The language itself is much more mathematical than most programming languages, which allows for "simple" programming; less syntax but more complex. R also has some issues with memory management, since R holds all the data in the active workspace within Random Access Memory (RAM) [7]. This limits the amount of RAM it can access. This could prove to be an issue when processing a lot of data, requiring the programmer to write exceedingly efficient code.

Python was created by Guido Van Rossem in the late 1980s [31]. The aim was to create an easily extendable simple scripting language. It became the leading programming language for improving productivity and code readability, using tabs rather than curly braces and simple syntax. The popularity of Python is due to its flexibility and a relatively low learning curve. Like R there is a large community for Python, however due to Python having a relatively large range of features and uses beyond data analysis it tends to be fairly fragmented in comparison [30].

Libraries for both are fairly similar, both having specific packages for statistical analysis, data processing and machine learning modules. With Natural Language Processing (NLP) necessary throughout the project and being highlighted as one of the most important aspects of the project this library will take precedence on the choice of programming language. Within R there is a library called TM, whilst in Python there is a library called Natural Language Tool Kit (NLTK), both created for this issue. Having researched this in detail there seems to be an obvious choice: "R and Python are close competitors in many kinds of data analysis and digital history, but if you were going to do only NLP then the NLTK would be a clear winner." [6]

The decision for this project will be to use Python due to developing skills that are used throughout the industry. This choice also allows for memory to be used effectively, data processing to be well formatted and NLP to be far simpler. The main factor for this decision was the more established NLTK library for NLP.

2.2.2 Hardware

Hardware shouldn't be a big concern for this project. There will be a lot of data being collected, processed and stored; this may take some time to run through and a lot of memory may be used during the run through. This may require some memory management within the software. The laptop used for the project has 12GB of RAM and therefore I don't see this being a blocker, just a time consuming aspect that comes with many machine learning processes.

Here are more hardware features of the laptop used for the project:

- Processor: Intel(R) Core(TM) i3-3120M CPU @ 2.50GHz
- Installed Memory (RAM): 12.0GB
- System Type: 64-bit Operating System
- Operating System: Windows 7 Ultimate

If hardware does become an issue there are ways around this, such as upgrading the hardware within the machine (either with a more powerful processor in order to quicken the process, or via additional RAM) or by renting a cloud server. Cloud computing have options to rent infrastructure to run models on and download the predictions. It isn't overly expensive [3] and could help everything speed up if it does start becoming a hindrance.

2.3 Data Selection

2.3.1 Website Selection

The selection of the website or collection of recipes needed for this project is critical. There needs to be enough varied data to provide a good overall dataset in which to develop a learning model on. Without good varied dataset with clear classifications of rating the project is unlikely to succeed. Within the research carried out, shown within sections 1.3.1, 1.3.2 and 1.3.3, it is clear that one website is favoured. Within all pieces of research allrecipes.com was chosen due to its large dataset, activity within the website and the large number of reviews on the website.

	1.3.1	1.3.2	1.3.3
allrecipes.com	true	true	true
epicurious.com	true	true	false
menupan.com	true	true	false

Table 2.1: Table listing the research papers choice of recipe website(s)

Following research, the website chosen for the project was allrecipes.com. Allrecipes was founded in 1995 [10] and launched in 1997. It was one of the first examples of knowledge sharing on the World Wide Web. Wikipedia followed suit with its more general view on knowledge sharing in 2001. It has 16 customized international sites where users can share their own recipes, rate and review recipes features on the website. The version of the site that will be used is the English one.

Other options that were taken into consideration were yummly.com, foodnetwork.com, epicurious.com and menupan.com. Epicurious.com and menu-pan.com were both seen within the research as seen within table 2.1. These were mainly included as a way to introduce a more diverse range of dishes from around the globe. After looking through all of the websites it was determined that none had the variety of dishes, enough reviews and ratings on their own. The option to collaborate the data once collected from multiple websites, as tried within the research is a possible option, however some of these websites use 4 star reviews and different layouts on the website would cause additional issues with data collection and collaboration. A breakdown of features for each site can be found within table 2.2.

On allrecipes.com each individual recipe has instructions on how to prepare it, this features:

- Recipe Name
- Short Description of Dish
- Star Rating
- Number Of Users That Reviewed It

	Avg. No. of Ratings	Variety of Recipes	No. of Recipes
allrecipes.com	600	Varied	Over 50,000
yummly.com	20	Varied	Over 25,000
foodnetwork.com	50	Varied	Over 60,000
epicurious.com	20	Gourmet	Over 330,000
menu-pan.com	5	Korean & Asian	?

Table 2.2: Table listing a comparison of recipe websites, this has been gathered from a random sample of each recipe website and may not represent actual findings.

- Number Of Users That Made It
- Ingredients (including measurements)
- Nutritional Information
- Time To Make
- Directions Of How To Make Recipe
- Reviews

Additional information is available from allrecipes.com such as the region the recipe is from, the type of meal (breakfast, dinner etc.) and if it is related to any holiday. This data can be found within the inner Hyper Text Markup Language (HTML) of the website and likely via an Application Program Interface (API) although this would require additional processing and isn't critical to the performance of the project.

2.3.2 Data Collection

There are two key ways in which to gather data from a website: Data/Web Scraping and via an API. The two approaches differ vastly in the way the programmer collates the information and both can be considered to be forms of Data Mining.

An **API** is often the go-to source of data needs from online and stand-alone programs. An API acts as a window to the code, and is created so one piece of code can talk to another. This allows for information to be passed between the two. This isn't just used for data collection but can be found within many programs; for example:

1. Libre Office <–> API <–> Microsoft Office
2. Yelp <–> API <–> Google Maps (pinpointing locations)

An API works by revealing some of the code and internal functionality of the program to make it possible to reveal information to the user without exposing all of the developer code. There are even APIs for open source code, due to the time it would take to look through all of the code available and pick out the necessary pieces of information. Using an API limits what you can reveal about a program to a small subset of features that may be needed for data purposes. This avoids any legal implications and reduces the time it takes to gather the information having been

predefined. There can also be issues with up to date information within an API. These stem from having to gather the data second-hand via another application rather than straight from the source. APIs collecting data often have charges for a number of "calls" (requests sent to the server for data). This limits the amount of data you can have access to unless you are willing to spend a lot of money, an example is BigOven API, one of the leading Recipe APIs available. BigOven has a cost of 500 requests an hour for $99 per month that runs all the way up to 10,000 requests an hour for $699 per month [1]. This is far too expensive for this project and therefore other methods of data collection were found.

Data Scraping is another method of retrieving data from a website. This approach is very different to using an API due to having to use the inner workings of a program to gather the information yourself and programmatically find a way in which to connect your program to another one. This works fairly well with online resources due to being able to view the inner HTML of a webpage and scrape the data from it by looking within the fields you need and pulling the information. There is a HTML library within python that allows visualisation of the inner HTML that simplifies lifting the necessary data. Creating a data scrape isn't as simple as just plugging in features into an API, however a data scrape has no limits or fees attached. The programming of the data scrape is a little more tedious and there are legal implications if the web page being scraped has copyrights within the page.

The legal implications for this project were considered and resolved by contacting allrecipes.com and receiving notification that the website could be used via a data scrape for educational purposes. This can be seen within figure 2.1.

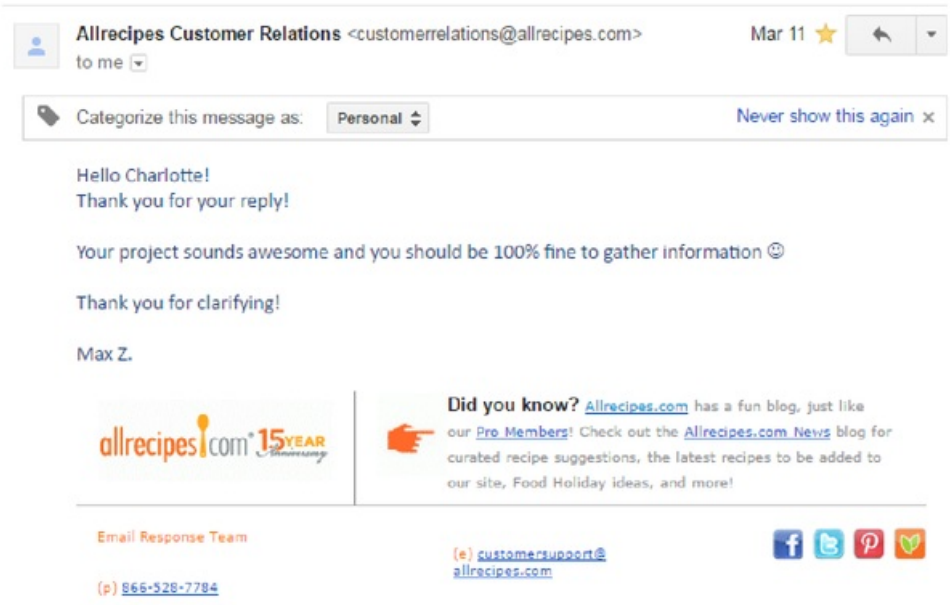


Figure 2.1: An email confirming the use of the website for a data scrape. This allowed the data to be gathered and progress to be made on this project, the entirety of this conversation can be found within appendix 4.1.

2.3.3 Data Storage

There are a few sources of data storage that can be used for this project. A database structure could be followed, this could be local, server based or via the cloud, or there could also be the use of file structures such as .txt and .csv files. There are pros and cons for all of the methods mentioned.

Using a **database-type structure** would require a lot of additional work. This method is however brilliant for long-term use and large amounts of data. The application requires a structured database to be created with relating tables and referencing tables. Once created the structure of these tables are very difficult to change, and therefore when doing experimentation with algorithms and data input this can become a hindrance. Before creating a database there would need to be research between what types of database would be most acceptable for the project. This includes the consideration of cloud based versus a local database and also whether the database would be a version of Structured Query Language (SQL), potentially MySQL or NoSQL. Due to having no intention of using the database no in-depth research was carried out. However, a quick look at this area shows that anything other than a local database would be expensive over the course of the project [4] and the use of a local database would limit the use of the end-program on another device.

Using a **file structure** requires much less effort in setting up. Reading to and from the files is much simpler and modules can be created in order to make transitions smoother. Using this method also allows the user to view the raw data whenever required and gives access to look at the processed information for a better idea of how it works. The access for the user isn't good practice however, due to the potential of changing the raw data. This would only affect their version of the program though so this isn't a pressing issue. The large files shouldn't cause too much of an issue due to only saving what is necessary. This also allows for other computers to use the program without the need of a local database.

2.3.4 Machine Learning Algorithm

A perfect machine learning algorithm does not exist for this project. Therefore the limitations of the algorithm used will need to be carefully considered. The aim is that multiple algorithms will therefore be adapted and applied to help resolve the limitations. With multiple algorithms, there shall be a comparison between the effectiveness of each algorithm on their own and also when applied together.

A learning style will be chosen for the machine learning algorithm. There are three different styles and within these styles are different methods of learning.

1. *Supervised learning* is a machine learning methodoloy that learns from experience. This version includes training data which has a known outcome "label"; in this case the data gathered will have a recipe and know that the average rating for that recipe is good or bad.
2. *Unsupervised learning* does not have a known "labeled" outcome. This learning style clusters similar information and organises the data by similarity.
3. *Semi-Supervised learning* is a mixture of "labeled" and "unlabeled" data points. These cluster the unlabeled with the labeled on the assumption that they have a similar outcome.

Supervised learning will be the main approach for this model. This will train on all of the "labeled" data until an acceptable level of accuracy has been found. Semi-supervised will also be present due to the input that the user has on their own recipe. This will be used to provide a rating based from the labeled training data. Unsupervised learning isn't used due to the input data having been labeled and the result being known without any clustering being necessary.

Artificial Neural Networks was the main algorithm used throughout the research carried out, mentioned within sections 1.3.1, 1.3.2 and 1.3.3. This algorithm creates a network of interconnected nodes (mimicking the brain and neurons within it), these nodes have numerical weights that are tuned based on experience. These create neural nets that adapt to input and allow for learning to occur, inferring rules based on input . The neurons/nodes are activated based on input. This is then transformed until an output node is reached and the result is determined. Having transformed multiple inputs into a binary result as shown within figure 2.2. This type of machine learning algorithm is ideal within rule-based programming and computer vision.

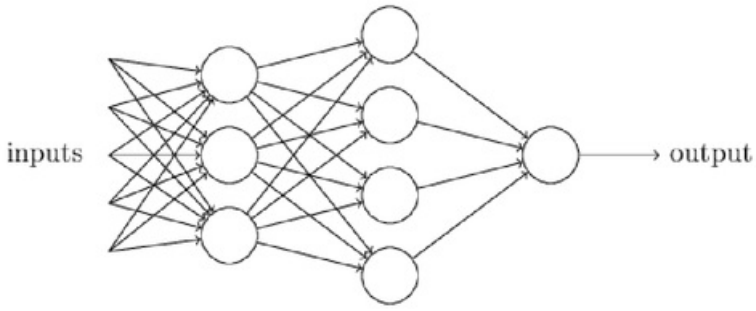


Figure 2.2: This image, taken from [21], shows a simplistic view of how the Artificial Neural Network comes to a decision. Each node is weighted in order to produce the correct result.

This machine learning algorithm could be used for this project, however the huge amount of ingredients and recipes may cause the neural net to become too large to process efficiently and create delays for the user. For research purposes this would be the go-to machine learning algorithm, as shown within the research carried out, but for a responsive and user-friendly style of program this would not be appropriate. It has high validity and accuracy rates due to it's method of learning being based on rules that it supplies from the input data. The rules created allow for even the most obscure data to provide informed decisions.

Bayesian Learning is a simple machine learning algorithm that takes a very simple view of the world. It assumes that everything is independent, as shown within figure 2.3. This theory uses the assumption that if, for example, cinnamon is in every recipe that is good then cinnamon is likely to be a good ingredient and everything that includes cinnamon is a good recipe. This style of learning is often called Naive Bayes due its very simplistic and fairly inexperienced learning technique.

There are three styles of Bayesian learning that can be used within this theorem [25]:

1. *Multi-Variate Bernoulli Naive Bayes* looks at binary values for feature vectors. In the case of this project it would determine whether a recipe has/doesn't have an ingredient. This is the most relevant choice.

2. *Multinomial Naive Bayes* looks at discrete values for feature vectors. This wouldn't really fit with the project but could possibly look at the number of times an ingredient is found within the recipe.

3. *Gaussian Naive Bayes* looks at continuous values for feature vectors. This would look at measurements of the ingredients within the recipe. This would be too much work and cause there to be excessive processing of the recipes.

The only version of the Bayesian Learning Theorem that would fit the project soundly without the need for a lot of additional processing, storage space and time for the processing to occur would be the Multi-Variate Bernoulli Naive Bayes (Bernoulli NB). This method is quick in comparison to most other machine learning techniques, takes a simple view of the information provided and allows for additional data to be tested on it after being processed (preventing additional unnecessary processing every time a user wants to test a recipe). There are issues with this method however, due to its extremely simplistic idealistic view it doesn't take into account mixtures of ingredients. This method, instead of networking the ingredients simply assumes that a good recipe has x therefore x makes it good, and relies on independence of ingredients which is rarely the case. There are also issues with the pre-processing. Since every recipe would need to include all the ingredients in order to compare against other recipes (whether the ingredient is or isn't in the recipe). This may cause extremely large arrays of numbers to pre-process and store.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Figure 2.3: The simple statement of the Bayesian Theorem, where A and B are events.

Decision Tree learning uses a predictive model commonly seen within data mining. When training a decision tree, the method usually is to work from the top and find the best metric that will split the dataset and continue splitting until an efficient model is found. It uses a flowchart structure, posing a multiple choice question (root node) that has the answers (branch) leading to the next multiple choice question (leaf node). The leaf node then becomes a root node and the process continues as a tree of questions, almost an "if statement" tree, that finally ends in a result.

There are two types of tree:

1. *Classification Tree* - produces a result outputted as a class label which identifies groups from relationships. This would be the chosen method, allowing determination of either star rating or the status of good/bad recipe.

2. *Regression Tree* - produces a result outputted as a continuous value as an estimated response. This could possibly be used allowing for the star rating to be found from the continuous zero to five result produced. This likely wouldn't be as accurate however as the classification tree.

Decision trees provide an easy to understand, interpretation and visualisation of the results. This allows the user to understand the process behind the decision made about the recipe they inputted. They could possibly figure out what went wrong with it and may be able to decide on an alternative to ensure it's taste matches expectations. The data wouldn't need too much preparation, and would continue to produce a tree which adapts based on training. Having two different types of

tree and there being an ability to combine the two via Classification and Regression Trees (CART) makes the model very flexible.

There are issues with this model however, these stem from the larger datasets. Although the algorithm can handle these, there are issues with the *greed* of it. This occurs when there are overly complicated trees that over-fit the data. As it handles every single recipe trained on correctly but often doesn't agree with real world insights and training data. Additional processing can occur that will "prune" the leaves and nodes that don't offer a lot but do work with some of the training data. This reduces the over-fit but also reduces the accuracy. Due to the algorithm working by asking if something has x or y there are some issues with its ability to ask it if it contains both (XOR), this isn't handled very well.

Random Forest is an ensemble learning method. This is different from the other three mentioned above. This method of learning helps to rectify the over-fitting of the training sets that often occurs, especially within decision trees. The ensemble learning method uses something called Bootstrap AGGregatING (BAGGING). This selects a random sample of features from the training set, fits multiple trees to unique samples and monitors the error rates [16]. It achieves this by finding all the possible solutions of the random subset that could occur when splitting the nodes within the tree so that most, if not all, circumstances are covered.

Due to the multitude of trees, the testing data is processed through all of the trees that were created. The outcomes are averaged over all of the results from the trees and in principle provides an unbiased final result. This takes a divide and conquer approach, with smaller trees and more of them it can provide more realistic and accurate results, removing noise and variance from within the models. It also allows for strongly correlated features to be identified without being diluted by all the other data. This method simplifies the model, as smaller trees are easier to interpret by users and the smaller trees also reduce training time and testing time. This can be seen within figure 2.4.

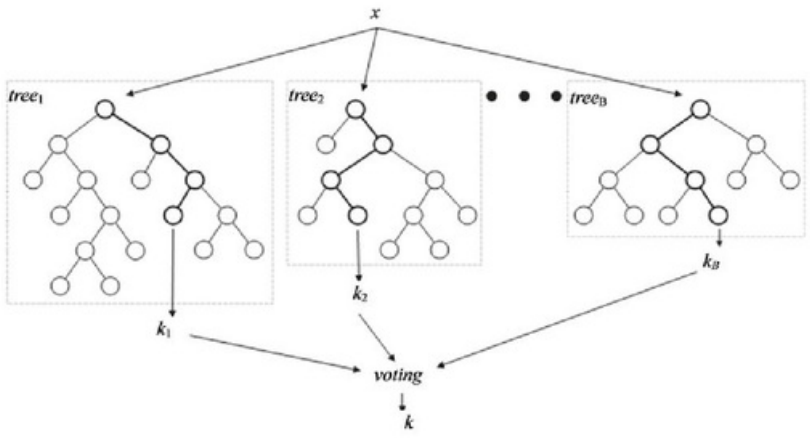


Figure 2.4: A visual representation of a random forest algorithm, taken from a study of the use of random forest classifiers within cancer research [20].

To **compare**, all of the highlighted machine learning techniques are supported within machine learning libraries within the Python programming language. This sets all of them on an even playing field. Within table 2.3.4 it summarises the comparison of the algorithms based on how

	Neural Networks	Bayesian Learning	Decision Trees	Random Forest
Training Speed	Slow	Fast	Average	Average
Testing Speed	Slow	Fast	Fast	Average
Accuracy	High	Medium	Medium	High
Simplicity	Low	High	Low	Average
Depth	Deep	Average	Deep	Average

Table 2.3: Table showing the pros and cons of the various machine learning algorithms mentioned for this project.

fast they run, their assumed accuracy ratings, the simplicity of running the algorithm (including the pre-processing required and the run through of the added user input once the initial run through has been completed) and the depth of the algorithm (this refers to how much memory will be required to run through the algorithm on a lot of data and whether it seems feasible). Table 2.3.4 shows that the use of a Bayesian method of learning will likely be the most effective for this process despite the issues highlighted above. If any additional time is available at the end of the project it may be worthwhile looking into random forest algorithms as an addition in order to provide a less bias decision tree version that can be visualised by the end user. This would replace an additional algorithm such as neural networks, as it already has extensive coverage within this area of the field and new explorations and experimentations being more interesting.

2.4 Foreseen Challenges

There are a few things that could be seen as challenging and may cause some blockers. These were highlighted within the research carried out within section 1.3.

Some causes for concern were found within the initial processing of the data within the research. This required a lot of parsing of the ingredients to stem what is actually meant by the directions within the recipes. This can be caused from misspellings by the user that entered the recipe, brand names being used and things such as brand names affecting what an ingredient is classed as. This will likely be the biggest issue that will have to be faced from the initial analysis.

Another issue that may occur will be from a lock of processor power within the hardware used for the processing of the ingredients. There will be tens of thousands of recipes, each holding on average (judging from the research) six ingredients. Being able to process these ingredients on a laptop may prove more of a challenge due to memory allocation (alternatives mentioned within section 2.2.2) and storing the recipes that have been processed rather than reprocessing them will be critical.

As with most machine learning approaches there is always the risk of over-fitting the data. This occurs when the model is better at making predictions on training data than additional unseen/test data. This could occur if the training data is based on the first 20% of recipes and these all happen to have the topic of baking; the model therefore wouldn't have the ability to know what a good pizza or pasta dish would look like. This will have to be addressed via a certain control over the data used for testing and training. There may be some issues with more unique dishes. This will likely be due to only taking data from one website rather than multiple ones in order to avoid ambiguity and multiple dishes that are the same. This does have some pitfalls in some cultures cuisines not having a more extensive range of dishes but will remove a lot of processing at the

same time. This will have to be tested once complete in order to see if further work should be carried out.

2.5 Process Methodology

2.5.1 Overview and Alternatives

There are many processes that can be used during the development of a program or project, both agile and classic methods such as "Waterfall". The one that was chosen for this project was an agile approach: Feature Driven Development (FDD) .

FDD was chosen due to its agile methodology which tackles a problem by breaking it down into manageable chunks called features. These features are listed and worked through as progress is made. This project works for this method due to its three main features: data collection, feature extraction and machine learning processes. Within these three features there are smaller tasks and this allows for the process to seem much less daunting and plots the best course of action for the remainder of the time available. The breakdown of the individual aspects of the project allows for detailed planning to occur without the need for additional design documents and detailed diagrams to be created. This removes a very time consuming and tedious aspect to the process.

Alongside its good design and breakdown features to this project, FDD is also extremely suitable for a single developer project. It has a good mixture of both design and implementation rather than focusing on one or the other. This approach is also extremely flexible, with priorities and features being able to change, adapt and be organised based on progress and additional information gathered during implementation.

Alternatives that were in consideration for this project were approaches such as eXtreme Programming (XP) and Scrum. These additional agile methods would also be suitable due to the ability to monitor progress and focus on flexibility rather than procedure. However, due to the single developer aspect of this project it would be less useful as pair programming has a large impact on XP, and Scrum is known to be a much more team or group development type.

2.5.2 Feature List

Referring to the requirements for this project, as seen within section 2.1, a set of features must be created in order to follow the FDD approach. They are as follows:

- Data Collection
  - Research Websites and Choose Appropriate One
  - Gather All Necessary Information from Website
  - Store Raw Data in Appropriate Format
- Feature Extraction
  - Find Individual Ingredients
  - Remove Measurements from Ingredients

  - Cluster Ingredients
  - Remove Unnecessary Preparation Instructions
  - Store Data in Appropriate Format (Stops Unnecessary Additional Formatting)
- Machine Learning
  - Split Data into Testing and Training Data
  - Run Machine Learning Algorithm Over Data
  - Add Ability for User to Input their Own Recipe
  - Add Processing on User Input
  - Run Processed Input Through Algorithm and Display Results
- Build User Interface GUI

Design of these features will be discussed within Chapter 3, whilst the implementation of these features will be addressed within Chapter 4.

Chapter 3

Design

3.1 Overall Architecture

Due to the FDD methodology chosen within section 2.5, with a list of proposed tasks detailed in 2.5.2, extremely detailed design documents aren't necessary. Within this section there is a brief overview of the proposed design for this project. There will be three separate programs that will run in order to ensure the best results for both the end user and the programmer. The end user will be able to use all of these, however the test data supplied with the program will ensure that the user only needs to use the final application. For the more adventurous user the additional files will be included if they wish to download additional training data. The three programs will be:

1. Data Scrape
2. Processing the Data
3. Final Application

3.1.1 Data Scrape

Within this section the entirety of the data scrape will occur. The initial idea is to use this to cycle through the website to ensure that as many recipes are gathered as possible. The flowchart shown within figure 3.1 shows the process of scraping data from the website and the plan of what to then do with the data. The process ensures that even when there are errors, it will account for and discard erroneous recipes. Once the data for the recipe has been obtained it will then store this information into a .csv file for further processing.

Whether a recipe is valid or not depends on:

- *The recipe existing* - some IDs won't exist due to either there being a fault with the website or the recipe possibly being deleted.
- *The recipe having been rated* - this will filter out those recipes which have yet to be rated, ensuring inclusion of the popular recipes, and also that the supervised learning approach is fully supported throughout the data scrape.

For the test data provided with the package of programs there will be a limit of five thousand records. For any additional data the user may wish to add, it will run exactly as before and there will be no limit. This will ensure that the user can download up to date recipes that may have only just been added to the site.

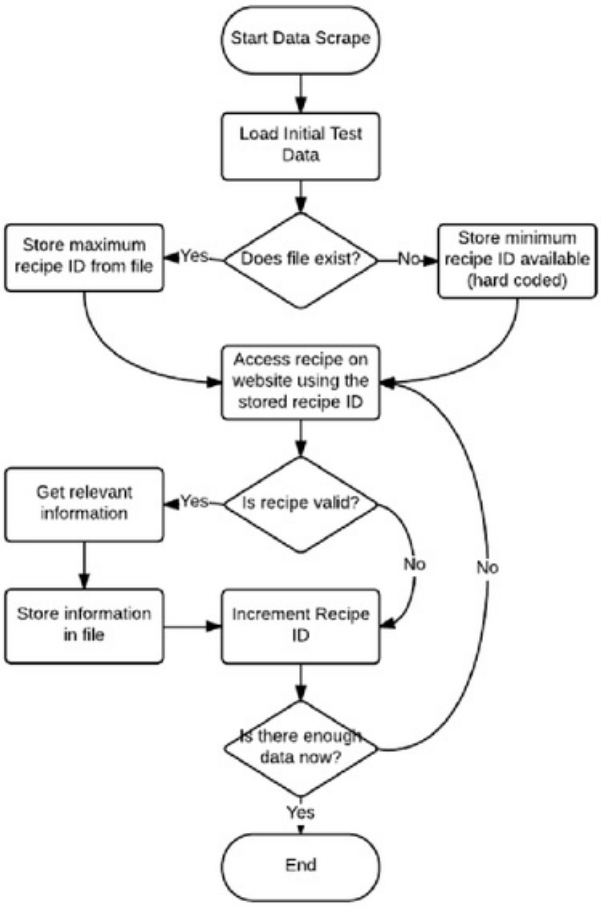


Figure 3.1: A flowchart showing the initial design of the data scrape script.

3.1.2 Processing the Data

The pre-processing of the data will occur in this file, it will look at the raw data collected from the Data Collection phase and begin processing it in order to make it relevant and useful. The majority of the work for this stage will be related to natural language processing and formatting it in the best way for the machine learning algorithm(s) within the final application. This can be seen more clearly within the flowchart in figure 3.2.

As shown within the flowchart, each recipe will be split into individual ingredients in order to process them using the NLTK library within Python. Each individual ingredient within the

individual recipe will have the measurements, cooking instructions and anything else deemed unnecessary removed. Once removed the ingredients will be transformed into a new list of processed ingredients. This will be repeated for every recipe within the file and then stored for use within the final application.

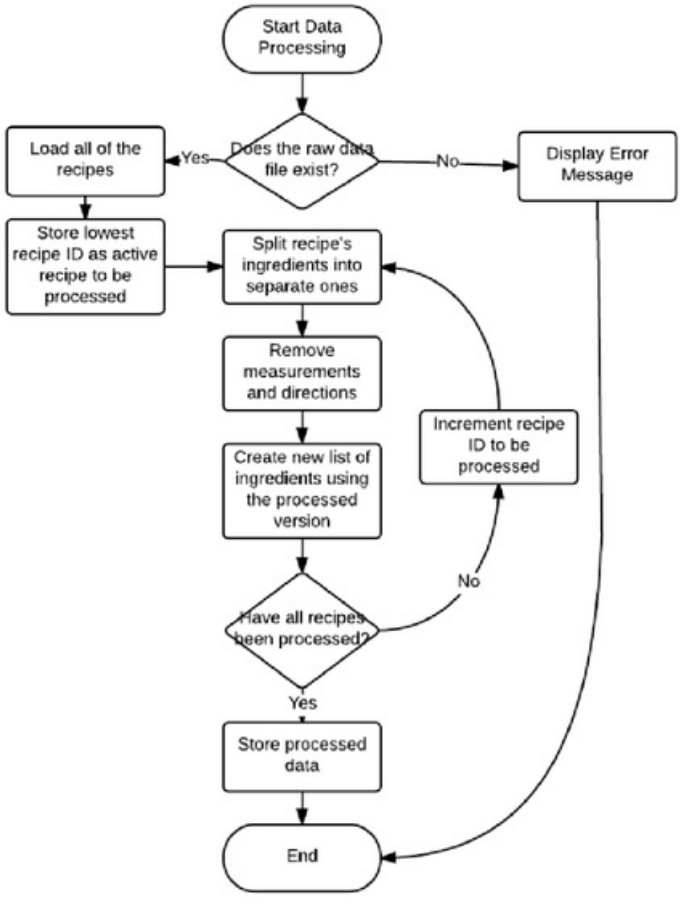


Figure 3.2: A flowchart showing the initial design of the data processing script.

3.1.3 Final Application

The final application will hold the main functionality. The functionality will include:

- Loading of processed data
- Training and testing on this data
- Allowing the user to enter their own recipe and test out what rating it would receive

The final application will be the main script that is used by the end-user and will possibly have a Graphical User Interface (GUI), the design of this is demonstrated within section 3.2.2.

The final application, as shown within the flowchart in figure 3.3, has a fairly simple method of finding out the predicted rating of a recipe. This is due to all the pre-processing having been completed and libraries being available in Python to create a simplistic program design. The user enters how many recipes they want to process, the user will then be kept updated whilst the loading, training and testing occurs. The user can enter this number because this gives them the power to determine the time they wish to wait and the accuracy rating due to smaller or larger datasets. The data is then split into training and testing data (likely an 80:20 split) thereby allowing enough data to train and some data to validate the training set. Once processing and training is complete the user can then enter as many recipe combinations as they wish and have the output of whether or not the recipe is good or bad.

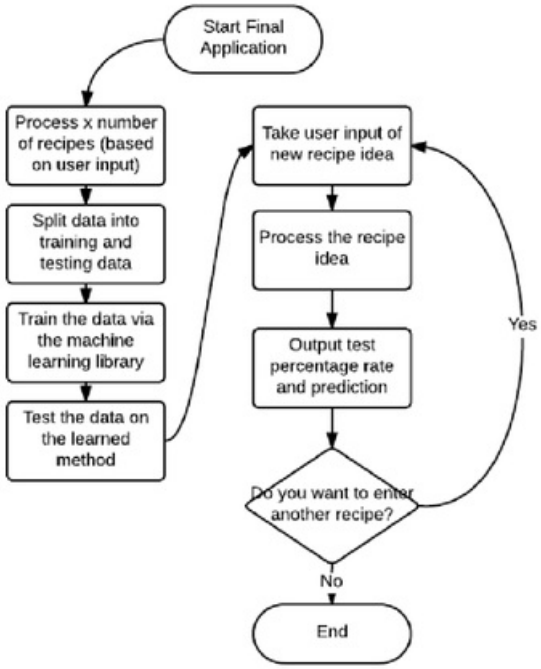


Figure 3.3: A flowchart showing the initial design of the final application script.

3.2 Detailed Design

3.2.1 Algorithm Design

There are two key algorithms that must be designed for this program:

1. Natural Language Processing as explained within section 3.1.2. Implementation of this will be a complex process and require experimentation of different library packages in order to find an effective solution.

2. Machine Learning Algorithms as explained within section 3.1.3, this will use one of the libraries within Python.

Both of these items will need further exploration, implementation and experimentation. The design will likely change due to the complexity and the non-exact art of Natural Language Processing. Therefore no detailed plans were created.

3.2.2 User Interface Design

The GUI will be one of the last things implemented if there is time within the project. Although this will give a clean and more finished look of the project for the end-user it is an unimportant aspect of the functionality behind the program. The functionality behind the GUI is explained within section 3.1.3. If implemented the GUI should include the following:

- An entry field for the number of recipes that the user wishes to process and train on.
- Progress bars to ensure the user remains aware of progress.
- An entry box for the user to enter their own recipes.
- A result for the user's own recipe.
- An assumed accuracy level based on the training data.
- Additional information about the project.

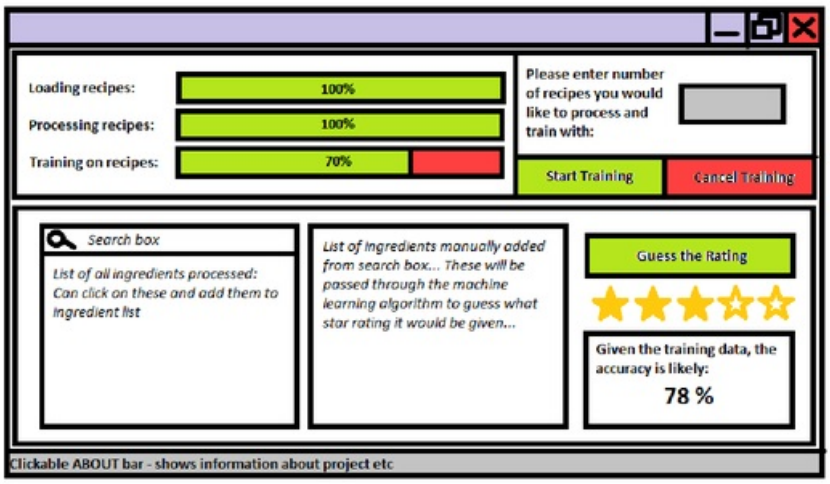


Figure 3.4: A simple mock-up of the planned GUI.

Figure 3.4 shows the initial idea of the GUI. The design was created within a simple piece of graphical editing software, designed from initial sketches that were created to visualise the end product, shown within appendix 4.2.

The idea of having separate sections for different parts within the GUI rather than a progressive GUI allows the user to look at the progress and also make changes to previous decisions. The user will start by entering the number of recipes they would like to train with. This will then load that number (or as many as possible if the number was too high) from the already processed recipes. The user will remain updated due to the loading, processing and training potentially taking a long time. The user will wait for these to load correctly, once this has occurred they will enter the ingredients they would like to process for their own recipes. Once processed, the output of the accuracy (based on testing data from the initial training) will be revealed as an assumption of the accuracy of the recipe that they entered. Alongside this there will be the predicted star rating. The user will be able to enter as many different recipes that they want without having to reload all of the training data.

There will also be an about bar that will provide information about the project in general and a contact email address for any issues/bugs to be reported.

Chapter 4

Implementation & Experimentation

4.1 Data Collection

Data collection, analysed within section 2.3.2 and designed within section 3.1.1, is a critical aspect of the entire process. Without having the data to process and train on there isn't a project at all. With data, also comes variance (how spread out the data is) and in theory the more data, the less variance will occur. Having less variance within data is necessary in order to find accurate predictions and remove some inaccuracies within predictions. The website chosen was allrecipes.com and collecting data via the data scrape rather than an API was decided upon. The legal implications already having been dealt with allowed for the collection to commence via scripts written for this purpose. An example webpage can be seen within figure 4.1.

The process of collecting data via scraping it from HTML isn't an exact science. Data scraping looks at the inner HTML within a webpage and takes the information specified from the chosen labels and divs. Once this information is received the script defines that a label contains a certain value. This value can then be stored and used throughout the rest of the process. Finding these labels and assuming the same HTML template is used on every recipe rather than individual pages of potentially terrible code is risky. Due to time restraints the only valid way to do this is to run through a random selection of webpages and ensure that it worked correctly on those.

To begin this process the information that needs to be collected from the website has to be listed and then found within the HTML. The information deemed necessary for this project are as follows:

- Some form of Recipe ID (if not found then can be generated)
- Recipe Title
- Ingredients (preferably minus the measurements)
- Rating of the Recipe
- The Number of Users that Rated it (for validation purposes)

To gather this information the inner HTML needs to be understood and studied, finding where information is stored within the code. To begin this process the recipe modeled within figure 4.1

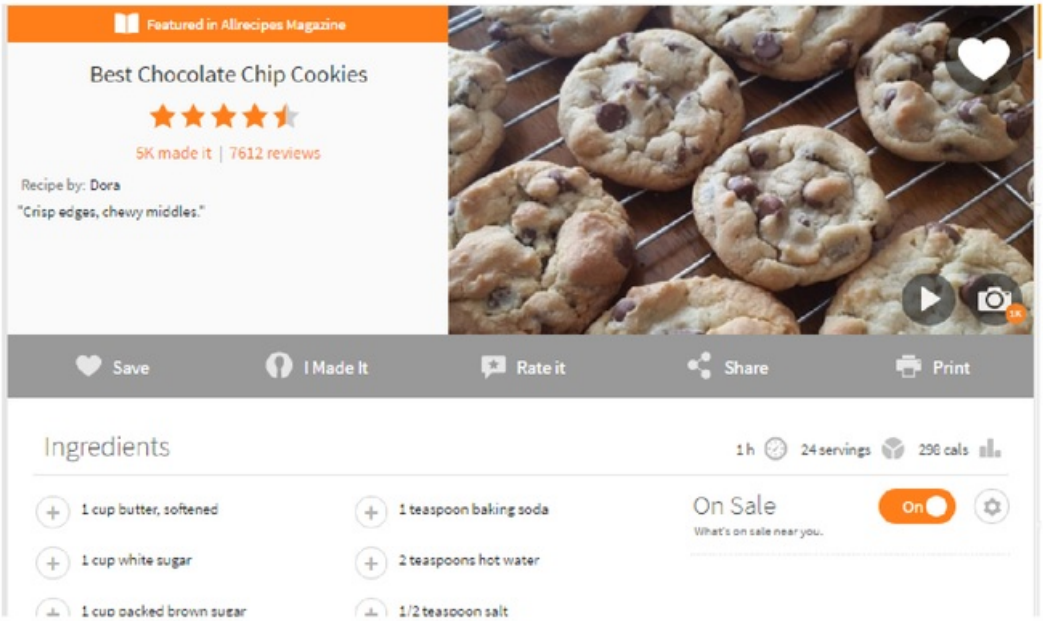


Figure 4.1: An example of the normal view of a recipe within allrecipes.com. This one depicts a Chocolate Chip Cookie Recipe (http://www.allrecipes.com/recipe/10813).

was randomly chosen and the information was searched for within the code. All of the information was found within the HTML and the tags were noted down. A few examples of these HTML tags are shown within appendix 3.1.

To transfer the webpage's inner HTML to be readable within Python, the lxml library was used from within the html library (shown within appendix A). The library uses a simple and powerful API that parses the eXtensible Markup Language (XML) and HTML. Using very simple commands the library accesses a Uniform Resource Locator (URL) from the script, requests to view the page and finally accesses the "content" or background information. Once this has been stored in a tree-like structure the information can be parsed to find the tags and divs that hold the more critical data for this project.

The unique number identified within the URL was used as the unique recipe ID and this allowed for a random recipe to be chosen without bias. The tags were then tested on fifteen of these random recipes, each individually, from allrecipes.com and they were found to be consistent throughout. That made the job much easier in order to retrieve the valuable information and discard everything else. The information that was received was printed to command line and checked manually in order to ascertain the validity. This can be seen within figure 4.2. However, one issue was that the ingredients were stored within the HTML with their initial measurements. This added additional processing to remove the pre-processing section due to there being no alternative way in which to pull the information without the additional directions.

Once it had been discovered that the data scrape worked well for these fifteen, a more general random thousand recipe search was applied and processed. Each recipe took roughly one second to process and display. The purpose of this was to quickly determine if any errors or anything obvious appeared when testing on a larger scale. When using this on a much larger scale, however,

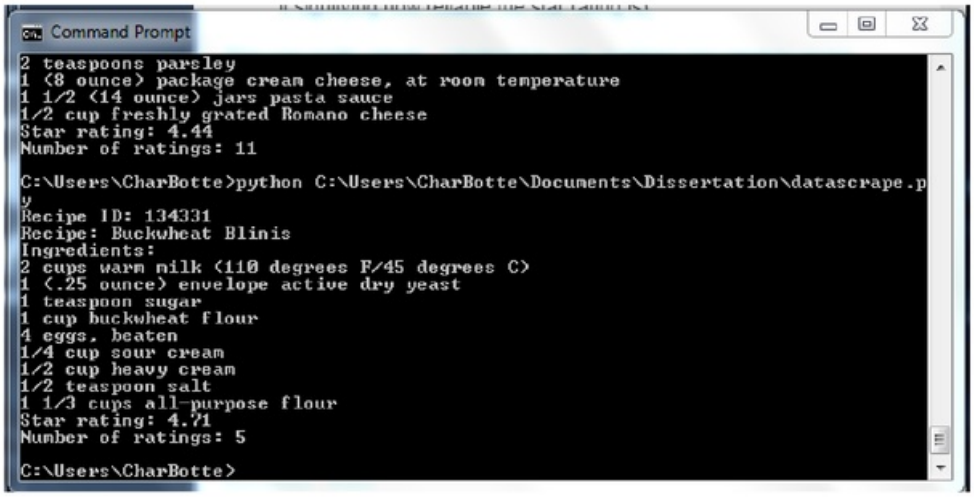


Figure 4.2: A random recipe was found and the information was scraped and checked manually against the original copy in order to ensure the validity.

an issue was discovered; there was a bug or a default recipe that was being used hundreds of times. This would not be good for usage within a machine learning algorithm as it would significantly skew the results in favour of this recipe. The recipe was "Johnsonville® Three Cheese Italian Style Chicken Sausage Skillet Pizza". Having seen this being used multiple times additional processing was added to the script. All of these recipes had no reviews and therefore I added an additional clause to only accept and begin processing recipes with one or more reviews/ratings on them, as seen within appendix 3.2.

Johnsonville® Pizza also brought attention to any registered and/or trademarked ingredients. Removing these was also deemed necessary in order to aid the reduction of processing and any errors that may have been caused by odd characters when saving or reading from files. The removal of these also helped to ensure that ingredients could be grouped more easily within the processing stage as, for example: tomato sauce may not have efficiently grouped with Heinz® Tomato Ketchup. An alternative approach to remove these recipes was to implement specific exceptions for these trademarked products in order to keep the additional data. This wasn't implemented due to the thousands of other recipes that were less difficult to process and the addition of these exceptions not making enough of an impact to make it worth the time spent on additional processing.

Once these errors had been resolved, an additional process was added so that every valid recipe was stored into a .csv as it processed it. This allowed for the processing to run for as long as necessary and gather as much information as possible as it ran through without any memory issues. Storing as it processed however, did take slightly longer due to having to open, write, then close the file many times. There were some timeout issues that were addressed but for an extra safety precaution having the information saved as it ran ensured that data didn't have to be reprocessed in the case of an unforeseen program failure or crash.

4.2 Pre-Processing Data

Pre-processing the data involves taking the human input of the recipe's ingredients, removing anything not food related and providing a list of ingredients that can be used within the machine learning process. There are three key features to the processing of the data; these being the Natural Language Processing (NLP); the removal of any features that prove erroneous or aren't critical for the design of this project; and, the formatting to ensure that the data doesn't need any additional processing within the final application.

To begin the NLP only a small dataset of around one hundred recipes was used. This allowed for manual checking to ensure nothing was going wrong and also quickened the process up during experimentation.

Before any actual processing of the ingredients occurred the .csv with all of the data from the data scraping was pulled into the program and stored into variables. In order to do this a new 'dictionary' of lists was used (discussed within appendix A). This dictionary allowed for each column to be referenced separately via its column name. Each individual column of each row was added to a list, thus allowing for the recipe name, rating and ID to be separated from the ingredients. For example *columns['Star Rating'][0]* holds the information about the star rating of the first recipe and *columns['Recipe ID'][0]* holds the same recipes' Recipe ID and so on.

Once the information from the .csv file has been processed properly it can be used to process the information, as discussed within subsections 4.2.1 and 4.2.2. During the planning stages for this section a piece of pseudocode was created in order to figure out the most direct and basic way to do everything, this can be seen within figure 4.3.

```
go through each recipe
split the ingredients
lemmatize and clean the odd words
if ingredient is within list of reference ingredients
    add a 1 under that index number for the recipe
else
    append ingredient to the end of reference list
    and add 1 to end of ingredient list


recipe[0] = good
recipe[1] = bad
ingredients[0] = [0, 1, 0, 0, 0, 1]
ingredients[1] = [1, 0, 1, 1, 1, 0]
reference_ingredients = [apple, flour, banana, orange, sugar, milk]
```

Figure 4.3: A very early and brief draft of the processing data stage, written in pseudocode.

4.2.1 Natural Language Toolkit

The Natural Language Toolkit (NLTK) was a key library (discussed within appendix A) that was used within the NLP of this project. The NLTK is one of the leading platforms allowing for Python to work with the human language and retrieve meaningful and interesting results. The library is used in order to classify and stem words so that the information can be read and understood without the need for human input. The library is so advanced that it can even detect whether a word could

be meant sarcastically. One of the key reasons Python was chosen as a programming language for this project was due to this library, as mentioned within section 2.2.1.

The library was used throughout the experimentation and implementation of the processing of the recipes that were entered by humans into the website. Human entry is not perfect and therefore spelling mistakes and issues including formatting do appear within recipes.

To begin processing the ingredients of the recipe, they were taken from the dictionaries referred to within section 4.2, and split into individual ingredients and then into individual words. Each of these was put into nested lists in order to ensure the ownership remained with the original recipe. Once split, REGular EXpressions (REGEX) was used to ensure any numbers were removed from the equation and all ingredients were changed to lowercase, ensuring all data was on a similar level. The words that were in fact numbers were discarded at this stage, leaving each ingredient with the most likely measurement type (cup, grams etc.) and the food type itself.

4.2.1.1 Stemming

The first task necessary was to remove plurals of words and look at the ingredient as a singularity. This is a necessary task due to the fact that plurals aren't part of a lot of the dictionaries and therefore can't be processed correctly. If they aren't a part of the dictionary they will likely be discarded and when running through that possibility it was found that words such as "raisins" and "walnuts" were being removed due to being pluralized. There were many options available that could be used for the stemming of the words, in fact there is an entire package full of different libraries on stemming within NLTK.

The first experiment that was run involved the *Snowball* version of stemming of words. There are many versions of the Snowball stemmer; the one used for this was the English version due to the ingredients all being in English due to the version of allrecipes.com being chosen. The use of this library first was due to having used it within sentiment analysis during previous work. This package takes each individual word and reduces it to the most common variant of the word entered (a demo can be found online: http://snowballstem.org/demo.html). This version worked but caused words to be changed quite dramatically: "cheese" would become "chees" and "cherries" became "cherri". Unfortunately this brings back the issue of these words not being associated to actual words within the dictionary and other libraries had to be addressed in order for more in-depth processing to occur.

The second and ultimate experiment revolved around the use of a *Lemmatizer*; another library from within the NLTK stemming package. The lemmatizer, instead of just removing chunks from the end of the word and simplifying it that way, takes the word and reduces it to its simplest form. This turns "cherries" into "cherry" and leaves "cheese" as it is. This worked really well and remained the basis of the stemming purposes for this project.

There are some issues with stemming words due to the English language not being overly simple. Some words need the plural in order to keep the same meaning, an example being peppers as otherwise the vegetable can be confused with the seasoning. The positives of the stemming of the words outweighs the negatives however, and the stemming remained within this project. The number of affected ingredients was minimal.

4.2.1.2 Removing Measurement Types

Once the initial stemming of all the remaining words was complete, the removal of the measurements was necessary in order to create a list of pure food ingredients that could be used within the machine learning module.

Finding out what was a food and what was in fact a measurement was a difficult process. There are many ways that a dictionary can be used within this, the method chosen to explore was synsets. Synsets are a method of finding synonyms and finding a degree of similarity between words. This is commonly used to determine whether words such as "puppy" are more similar to the word "dog" or "cat" in order to group animals, objects or anything else necessary. The idea behind this was to find a possible method to allow for the grouping of all food items and all measuring tools. Unfortunately, after a lot of time and effort the synset method wasn't accurate enough to use. This method often mistook food items for measurements and vice versa due to the very similar nature of them. Time was spent adjusting the variance for these synsets, and multiple versions of the similarity synsets were checked against.

Within wordnet there are multiple varying similarity checks. Each uses a slightly different approach and each was tested against varying numbers of datasets in order to find an accurate version as described below (paraphrased from the wordnet guide [8]):

- *path similarity* - This version of synset similarity checks the similarity between two word senses. This looks at the shortest path between hypernyms (broad meanings) and values this within a zero to one range.
- *lch similarity* - This version is the Leacock-Chodorow Similarity. It returns a score based on the shortest path that connects the senses via a maximum depth taxonomy. The relationship is given as a $-log(p/2d)$ where p is the shortest path length and d is the taxonomy depth. The values vary greatly.
- *wup similarity* - This is the Wu-Palmer Similarity method. It checks the similarity between the words' senses, then, based on depth of these senses in the taxonomy, finds the least common ancestor node. The score given will range from zero to one.
- *res similarity* - This is the Resnik Similarity method. It returns the similarity between the two word senses by basing it on the Information Content (IC) of the Least Common Subsumer (LCS - most specific ancestor node) and uses this to generate a result based on the corpus chosen. The result varies based on corpus and whilst testing this method a few were used including brown's corpus (one of the more commonly used corpus' available via wordnet). The range of values varies greatly.
- *jcn similarity* - This version is the Jiang-Conrath Similarity method. This method finds the similarity by using the Information Content (IC) of the Least Common Subsumer (LCS - most specific ancestor node) like the res similarity method. The score relationship is given as $1/(IC(s1) + IC(s2) - 2 * IC(LCS))$. This score ranges from zero to one.
- *lin similarity* - This method again uses the IC and the LCS in order to find the similarity rating. The relationship is instead however, depicted by $2 * IC(LCS)/(IC(s1) + IC(s2))$. The values range from zero to one.

All methods, despite tweaking them and attempting to find the best number that would define what is/isn't a food type, couldn't efficiently and more importantly accurately define a good measurement removal device. The easiest versions to work with were those with set ranges, such as the path similarity due to having fixed numbers to tweak and test against.

Another method of attempting to remove measurement types was addressed and attempted. This method was to use a URL to pass through the ingredient word into BBC Food's ingredient website (www.bbc.co.uk/food/) to ascertain whether or not the ingredient word was in fact food or not. A sample of this method working can be found within figure 4.4. This worked but took a long time to process words, and in turn ingredients, and then full recipes. Therefore the idea was scrapped.

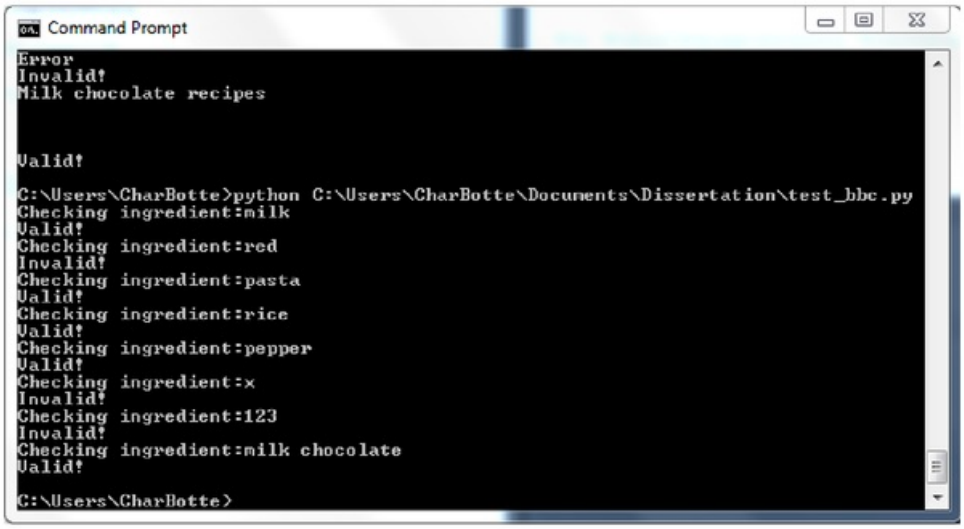


Figure 4.4: A view of the code working in order to find whether an ingredient was food or not, this used the BBC food website to see whether it had a page and therefore inferred validity.

A lot of time was spent looking into these methods and tweaking them, attempting to get them to work, but unfortunately it became too time consuming and focus was drawn towards a few other methods and then on to the final application.

4.2.1.3 Removing Non-Nouns

Due to some words still being within the list of ingredients, as not all words are measurements that need to be removed, additional checks were added. These included the removal of any non-nouns. Due to additional descriptive words being found within the directions, such as "dried and chopped parsley" the removal of everything but the keyword, in this case "parsley", is necessary. All food types are nouns and therefore there was no issue with removal of these or any potential problems associated with their removal.

This was available via a synset dictionary within the NLTK wordnet corpus. All the nouns were extracted from the dictionary within this corpus and made available to refer to throughout the script. If the word was contained within this shortened dictionary it was classed as valid, anything else was removed. The difference this made was monitored over a small subset of data

in order to ensure that it was worth taking the extra memory for this process; this can be seen within figure 4.5. It shows that the ingredients were cleaned up a little more by roughly 4%. Due to memory management being necessary throughout this project, it was classed as a worthy use of additional processing during this stage, detailed within 4.2.3.

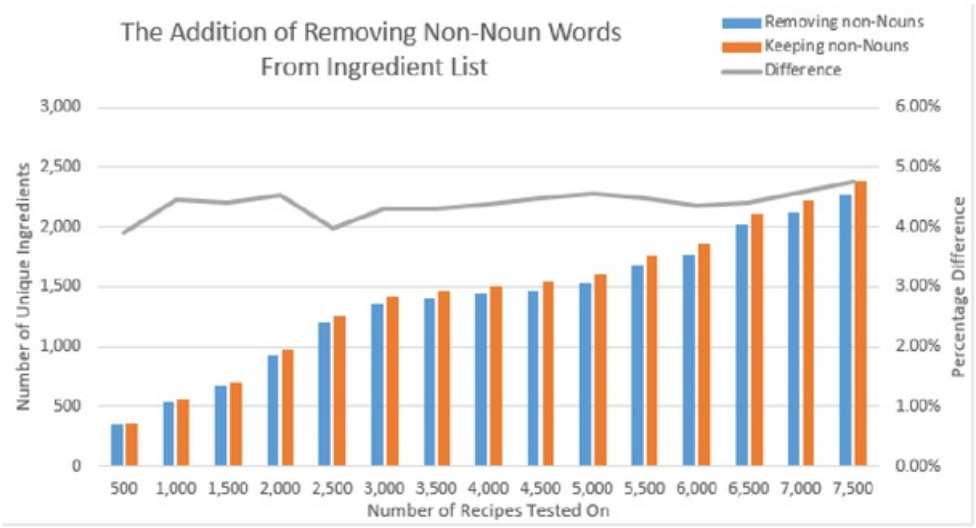


Figure 4.5: A graph showing the effect of removing additional nouns from the ingredient list.

Some issues were found with this however, as there are many different meanings of words within the English language and the dictionary has to account for each one. This means words that need to be removed such as ”sharp” from ”sharp cheddar cheese” will not be removed due to its less commonly used ’sharp’ term as in a sharp note within music (which happens to be a noun). As this method does hold more value than this, it was accepted and a few additional words were added to the stop words introduced within section 4.2.1.2.

4.2.2 Formatting for Machine Learning Processes

Once the processing of the ingredients, and as much reduction and groupings of ingredients had been completed, the next job was to format the information so that the machine learning process could be used. The machine learning algorithm that was chosen to test on was the Bernoulli Naive Bayes model. This required a very specific format to begin fitting and using this method of prediction. Bernoulli Naive Bayes looks at a list of features; this list has only two varying information points, zeros or ones. The list can be as long as necessary but does require that each and every list it processes is the exact same size.

These lists pose an issue, detailed within section 4.2.3: the progressing size of having to list each and every ingredient for every single recipe, and whether it does or doesn’t contain it, takes up a lot of space. This causes problems when processing huge amounts of data, even though there are binary (integer) values that are held within the list.

Below shows an example of the process created for the recipe system (code can be seen within Appendix 3.3):

A recipe contains: flour, sugar, egg, milk and vanilla extract. This will be converted into a list of [1,1,1,1,1] as every ingredient has been used and a list of reference ingredients is created.

Recipe 1 = [1,1,1,1,1] and Reference Ingredients = [flour, sugar, egg, milk, vanilla extract]

Another recipe contains: flour, egg, milk, chicken, cream, cheese. This recipe would look at the reference list and check to see the ingredients it contains and doesn't contain whilst adding additional ingredients to the reference list. The list creates a new list but the list is longer due to additional ingredients being added.

Recipe 2 = [1,0,1,1,0,1,1,1] and Reference Ingredients = [flour, sugar, egg, milk, vanilla extract, chicken, cream, cheese]

With each additional recipe and any additional ingredients the number of features within the list necessary for understanding what is and isn't in a recipe for the Naive Bayes model expands drastically. This can cause issues, as referenced within section 4.2.3. This requires an additional run through at the end of all processing, to add zeros up to the length of the longest list to create equal and processable lists of data for the machine learning algorithm.

Once this has been fully processed, the list of reference ingredients is written to file. Another file is also written which contains a column showing the average rating (as a decimal format) and the list of zeros and ones created as shown above. The average rating is written to the file so that the ideal split can be found either via a 1 star - 5 star rating or good/bad recipe. This will be decided within the testing phase whilst developing the final application and can be seen within section 4.3.3.

4.2.3 Memory Issues

During the process, some issues were uncovered; these predominantly were caused via memory problems during this data processing section. Unfortunately, during this process everything is saved in-memory rather than reading and writing to the file for each individual recipe, due to both time and functionality within. Reading and writing each recipe's data as it is processed would take an exceedingly long time. Time wouldn't be an issue if it were 100% non-user facing but due to the user having access to download and process additional data this wasn't acceptable. Time isn't the only factor. The functionality of the program needs to have access to every ingredient and recipe in order to store it correctly, as shown within section 4.2.2. This poses an issue. With a lot of data to process, everything stored within the memory, and time being a factor within the process, something had to give.

The issue with memory only occurs when running through the entire 75,000+ recipes at once. Fortunately, the user is not likely to have such issues due to processing a smaller number of recipes at a time. There were no actual blockers being caused by these small issues other than an occasional crash when the entire dataset was processed. The decision was made not to use any additional RAM or use a third party via the cloud.

This problem is both caused by the programming language and the choice of table structure, but the decision to continue usage of this and not purchase additional processing power was outweighed by the cost:efficiency ratio. Whilst using the computer for this initial process, as detailed

within section 2.2.2, the only fault in the program was the occasional crash due to this issue. If there had been a bigger obstacle, alternatives may have been explored. This would either have been via the use of manual memory management over Pythons default automatic memory allocations and garbage collections or the use of third party servers or possibly even using bitarrays [2] over regular arrays.

4.3 Final Application

The final application is the main focus for the user. This links them with the main purpose of the project and allows them to view the accuracy rating, train the machine learning algorithm and ultimately add their own recipes in order to test what rating they expect. Initially nothing is loaded into the program and the user controls everything from within the GUI.

4.3.1 GUI

The final application is the only key user-facing application, it is also the only graphical user interface implemented within the whole program. The GUI was a last minute refinement after a suggestion that the end result would have a much more "finished" look if it had an interface rather than text within the command line. The GUI allows for the user to interact with the program and direct what they want via text boxes and buttons that direct everything in the order the user wishes rather than having to follow a certain structure.

Tkinter was the main library used to create the graphical user interface. Using this allowed the creation of different sections via canvases, buttons and textboxes. The idea of using canvases rather than one entire GUI screen was to allow for different x and y co-ordinates for each canvas and the customisable aspects of each individual canvas being much simpler to apply. Once these were added, text boxes, buttons and labels allowed for intuitive usage via the interface. Each button references different functions and passes the user input through this to create a completely unique experience for individuals. If user input is invalid, or nothing has been entered, message boxes appear to indicate why the user cannot proceed with the program.

The final application looks like figure 4.6. There were a few notable changes from the original design within section 3.2.2 and figure 3.4. The top two sections were reversed in order to give a more fluid design; as most people read from left to right the best idea was to begin at the top left and work right and down. The "cancel" button was removed due to a lot of bugs that couldn't be fixed (seen within section 5.1). Instead of a search box, the user can input their own recipes via the text box and the ingredients are processed (detailed within 4.3.2); this allows for the user to enter any recipe they wish without the need to search for ingredients. The remainder of the GUI stayed similar to the design plan.

Each time the program is initialized, no information is automatically loaded. The amount of data loaded is decided upon via the user input. Unfortunately, if they enter a lot of recipes, significant time and memory will be required on their machine.

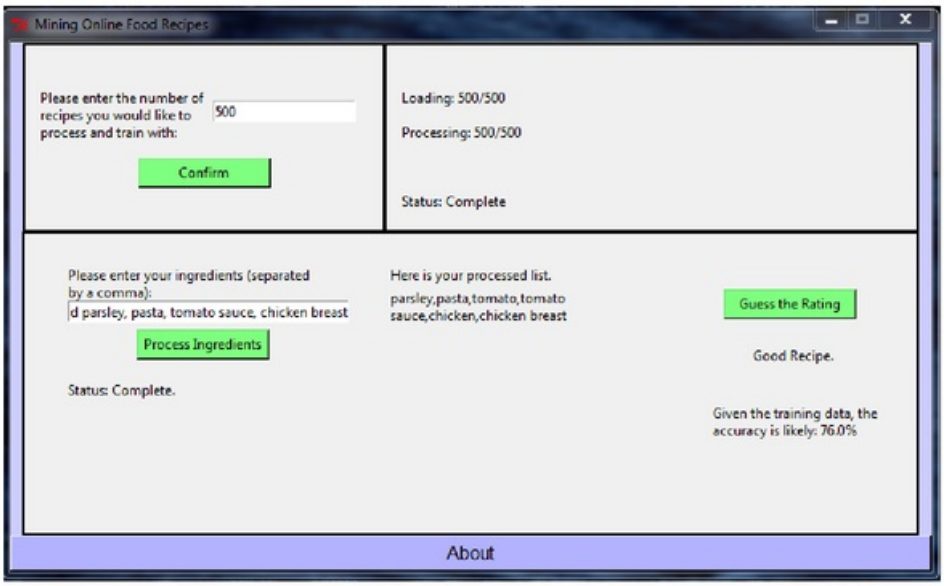


Figure 4.6: The final look of the GUI.

4.3.2 Processing User Input

The user can enter and use the program as they wish. The main processing, other than the small number of checks on the "number of recipes to train" box, is on the users own recipe that they can enter and determine the predicted rating.

The small amount of processing for the "number of recipes to train" checks to see that a positive integer was entered as the number of recipes they wanted. If invalid, the user receives a message box explaining they need to enter a valid number into the box. If valid, then this number is passed to a function that will load that number of recipes. To store them correctly requires two different files to be read: reference list of ingredients (used to compare the user's recipe and create a valid list later on) and the number of recipes (including their decimal rating and list of formatted ingredient lists). These are then used to train and test the Bernoulli Naive Bayes on this dataset (testing allows for a % valid to be displayed as a guideline). The trained model is then stored and can be used to "Guess the Rating" via inputting the user's recipe, processing it and then pressing the button.

The much larger section of processing resides within the recipe entry. This entry uses the same ideology as the original processing. It will use the same lemmatization and, in order to ensure additional matching in the case of more unique ingredients, it takes each word and the combination of words. For example: "tomato sauce" as demonstrated within figure 4.6 will be turned into "tomato" and "tomato sauce". The processed version of the ingredients is displayed within a label next to the entry.

Once both the recipe has been entered and the machine learning algorithm has been loaded the user can finally press the button to "Guess the Rating". Once this is pressed it ensures that both the machine learning algorithm has been trained and a valid recipe has been entered with at least one ingredient. If the user's inputs have been accepted, then the recipe will be formatted correctly

to be input, and fit to the trained Naive Bayes model. A result will then appear for the user to see. Both the training and fitting can be seen within section 4.3.3.

The user can change their mind and add new recipes without reprocessing all of the background data due to the Naive Bayes algorithm accepting additional input once processing has occurred. The extra functionality to process a different set of recipes has been added as well, allowing the user to enter a new number and process more or less recipes to increase the accuracy of the model.

4.3.3 Naive Bayes Algorithm

With the formatting already handled for the training, testing and fitting of the data, the only thing to consider is the Bayesian method itself. A simple addition with the help of the library sklearn.

To begin this process the data is firstly filtered; this changes the "average rating" for each recipe to something that is more relatable to a lot of recipes. To start, the idea of 1 star to 5 stars was brought up and an easily changed function (the final version of which is displayed within appendix 3.4) was developed, so that a recipe could be classified within one of these five classes.

As soon as the initial classes of results were decided upon, the data was then split into training and testing data. This was done via randomly assigning each recipe to either training or testing based on a ratio (code available within appendix 3.5). This is easily changed and can be modified if necessary, however, it is currently showing at 90% training and 10% testing data. The training data is then completely split from the testing data and is used to create the Naive Bayes model. This is then used to test the data and the results are compared to the actual results for the recipes. This provides a base idea of how accurate the prediction model is, and can be used to verify the effectiveness of the model. The Naive Bayes model is extremely easy to use to train and test and fit due to the scikit learn library that has a complete package for the machine learning algorithm.

Once testing began on multiple recipes it became obvious that due to the algorithm, the 5 star result set was too high and too many were being classed incorrectly. The potential results were therefore changed and tested for both good and bad or good, bad and ok recipes. When this was tested both worked much better and due to having more information for the user the final decision was to make 'good', 'ok' and 'bad' the final result set to choose from. This is again demonstrated within 3.4.

After modifying the ratios and the result sets to be as good as possible, the ability for the users recipe to then be fit into this is simple, via one line of script. There is no need to reprocess and train the data as the model exists and can be used as long as needed. When fitted it produces a result and the user can view it, to see how good their recipe would measure up to be.

Due to the time taken on the pre-processing of the data, this was the only machine learning algorithm implemented within the time frame. If more time had been available and more resources to quicken the pre-processing, more algorithms would have allowed the user to either choose between them or check their recipe against both to see how they differ. More information on evaluation of the algorithm can be found within sections 6.4 and 6.5.

Chapter 5

Testing & Results

5.1 Testing

Before testing, the suite of programs were turned into executable files. This was achieved via pyinstaller and turned all of the python files into versions that any user can use rather than having to have Python installed. This did vastly increase the file sizes however, which could cause issues with distribution.

The suite of programs, along with README.txt file were shipped to a Quality Assurance tester, Johnathan Colgrave from Jagex, to have an additional user test. He wrote his own test plan after reading the README (seen within appendix 4.3) and seeing the product first hand. This can be found within section 5.1.1. Another set of end tests was carried out by the developer. This can be found within section 5.1.2. Alongside both black box and white box testing, continuous testing was carried out throughout development and during each phase of FDD.

5.1.1 Black Box Testing

Black box testing takes an external perspective. The tests have no basis on the internal structure of the program and both valid and invalid information is inputted to provide destructive testing. Both external and internal testing approaches were implemented, allowing for full assurance that the program runs correctly on other machines and other users can easily understand the purpose and functionality of the program.

5.1.1.1 External

To get a professional view, a former colleague at Jagex Games Studio was contacted to provide additional testing. To summarize his report (the full report can be found within appendix 4.4) each program was tested separately and more information is provided below.

The data scrape aspect worked correctly. The sample set that was provided with the program was accurate and any additional data that was processed, stored properly. A sample was taken from the file and tested against the live website. Each recipe had the correct details and any IDs that failed did so without crashing the program, and provided information via the command line

interface. The IDs that failed to write to the file were found not to have a webpage and everything worked as expected.

There were issues with the other two programs, however. The process data executable file did not initialize; this was due to an error looking for libraries and corpus' that did not exist on the machine. Unfortunately this meant any data that could be scraped wouldn't be processed and this makes it almost worthless. As test data was already available the final application could still be used.

The final application initialized, but as before, the NLTK library was not found, and therefore any attempts at processing your own recipe were unsuccessful. The errors were handled so the program did not result in a crash but no error message was received. The processing of the test data however, did work and live feedback was provided, with the resulting accuracy rating displayed.

The hardware of the computer used for the external testing was as follows:

- Processor: Intel(R) Core(TM) i5-2540M CPU @ 2.60GHz
- Installed Memory (RAM): 16.0GB
- System Type: 64-bit Operating System
- Operating System: Windows 7 Enterprise

5.1.1.2 Internal

Internal testing was carried out throughout the entire development process. After each iteration of the FDD cycle, tests were done to ensure that the features that had just been implemented worked correctly. Experimentations with varying values were tweaked during the development, allowing for the program to work efficiently and accurately throughout the process. These weren't documented, but information regarding this experimentation is found within chapter 4.

Due to the issues found within the initial tests sent externally, the python files were tested rather than the executable ones. The final suite of programs was tested by the developer. This involved the creation of a test plan and results in a full list of known bugs which can be seen within section 5.1.3. The overall results of these tests are found within table 5.1. They show that all programs ran well but problems did arise within each one.

Program	Initialization	Functionality	Result	Verification
Data Scrape	Pass	Pass	Pass	Partial
Processing Data	Pass	Pass	Pass	Partial
Final Application	Pass	Pass	Partial	Partial

Table 5.1: Table showing the results of black box testing by the developer.

Data Scrape

The data scraping program was run through and didn't have any errors; information was passed to the user via the command line interface showing the progress being made. This can be seen within figure 5.1. The data within the csv file that is created or modified was then tested to ensure the information was valid and accurate to the website. Fortunately the ID provides a lookup to the allrecipes website by using the ID within http://www.allrecipes.com/recipe/RECIPEID. A random

number generator was used to find 10 random IDs from the list of recipes within the csv file. The results of these tests can be seen within table 5.2.

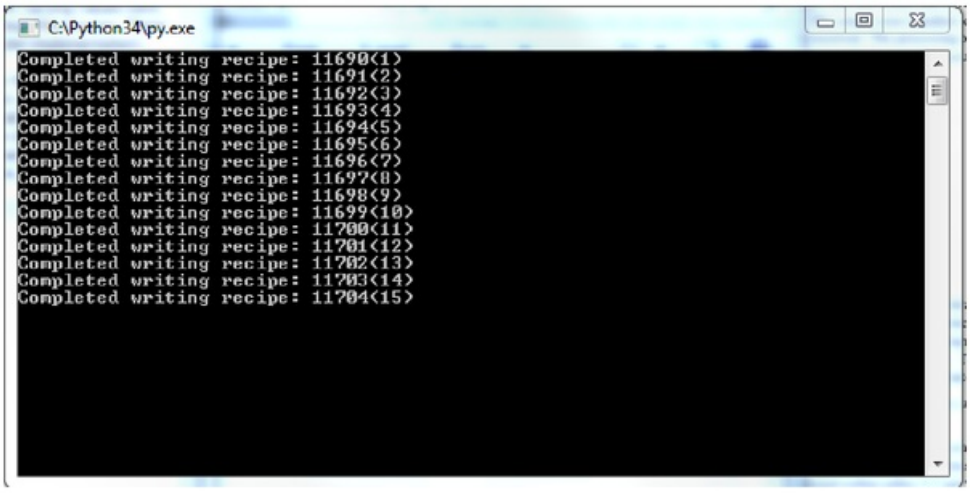


Figure 5.1: The black box test, showing the visual output of the program within data scraping. This shows that the user stays informed of progress.

Some of the reviews within the sample set were slightly wrong. This is due to the recipe having been gathered previously and therefore more reviews and ratings being given on the recipes. All of the incorrect versions were more than the number stored within the file, therefore there isn't cause for concern. This isn't an issue and is only listed as partially passing due to the figures not 100% matching. Some issues were revealed showing that if there are separate parts to a recipe (in the case of recipe ID 7,971 'filling', 'cake' and 'icing') then the words 'filling', 'cake' and 'icing' are all entered as individual ingredients. This could cause an issue during processing; it isn't a blocker or a problem that has to be dealt with immediately, but should be addressed at some point.

Sample	Recipe Name	No. Ingredients	List of Ingredients	Star Rating	No. Raters
6,989	pass	pass	pass	**partial**	**partial**
8,356	pass	pass	pass	pass	pass
8,775	pass	pass	pass	pass	pass
7,229	pass	pass	pass	pass	pass
7,195	pass	pass	pass	pass	pass
7,971	pass	**fail**	**fail**	pass	pass
6,864	pass	pass	pass	**partial**	**partial**
7,721	pass	pass	pass	pass	pass
10,701	pass	pass	pass	pass	pass
7,821	pass	pass	pass	pass	pass

Table 5.2: Table showing the results of black box testing of the data scraped from the datascraping program and the stored data within recipe.csv.

Data Processing

The data used was drawn from the test data provided within the package of programs. The user

is kept informed of progress via the command line interface providing information on how many recipes are being processed at once (shown within figure 5.2). The user can't see the actual result of this unless they look into the file. The program runs very quickly and the files are created correctly for this small dataset. A quick look into the file shows that the ingredients are sorted correctly, and the ingredients look to be sensible with no added measurements seen. Some additional processing could be beneficial as there are items that should be refined, such as within 'solid pack pumpkin puree', 'solid pack' should be removed. There are other problems with really specific ingredients that would benefit from further refining, this includes 'wheat barley nugget cereal'.

Figure 5.2: The black box test, showing the visual output of the program within data processing. This shows that the user stays informed of progress and can see that something is happening and they can estimate how much longer it will take.

The 5,000 recipe dataset processed well, even with the additional data that had been gathered from the data scrape, and there were no issues identified.

Final Application

Table 5.3 shows the program running correctly. This includes the user staying informed of all processes and any error boxes that are necessary are displayed. Some problems do occur when a lot of data is processed, however. The GUI can crash, leaving the user unaware of what happened. The program runs through and users can change their minds on different aspects, and the entirety of the program is layed out and labeled well so that no instructions are required, however, some users may be confused and an instruction manual may be helpful. There is an 'about' section with an email address that can be used to ask questions to the developer, so this additional instruction manual may not be necessary. The 'about' section discusses what the main focus of the project is and what it aims to do; this is in a separate window which allows for the continued use of the main program whilst reading.

The actual functionality of the program works well. When the user enters the number of recipes they want processing, this number of recipes seems to be processed, you can tell via the numbers changing within the status, as shown within figure 5.3. There are some issues with the actual estimations of the rating. The rating is shown but this is often inaccurate, this is assumed to be due to the machine learning algorithm that was chosen or the splitting of the data.

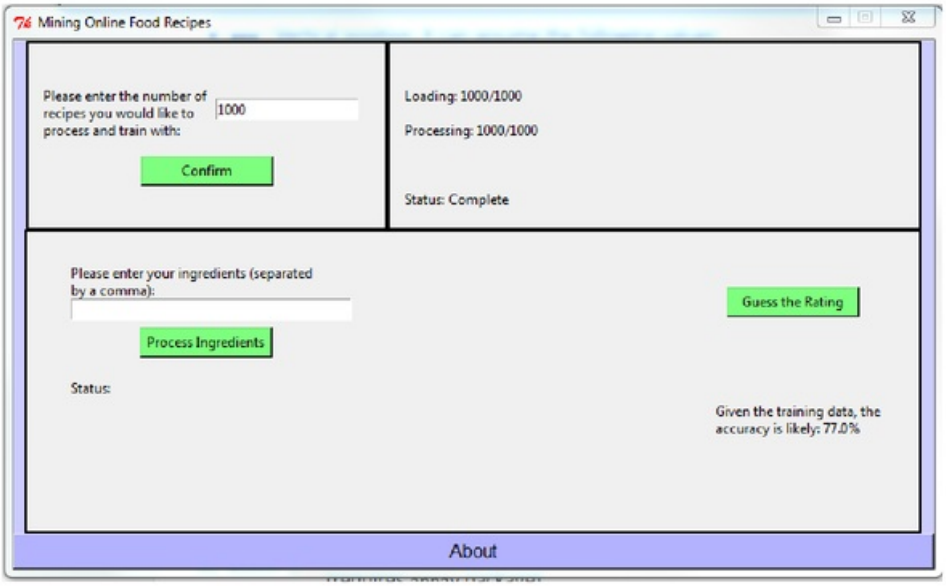


Figure 5.3: The black box test, showing the visual output of the program within the data processing. This shows that the user stays informed of progress and can see the actual number of processed recipes and the status of these.

Test No.	Test	Result
1	Valid User Input	Pass
2	Invalid User Input	Pass
3	Correct No. of Recipes Processed?	Pass
4	Accuracy Rating Shown and Machine Learning Process Run	Pass
5	User Recipe Processed Correctly	Pass
6	Guess the Rating Reveals Correct Result	**Partial**
7	Does the User Know What is Happening at All Times?	**Fail**

Table 5.3: Table showing the results of black box testing by the developer on the final application.

5.1.2 White Box Testing

White box testing takes an internal perspective. The tests are based on the system design and internal structure. These tests require programming and identify every possible way in which to use the software. These are then tested, usually via unit tests, ensuring the continued validation that everything within the program is working, even after changes are made.

Python has a unit testing library called PyUnit but this wasn't used for the project due to the sensitivity of the data. Unit testing can often lead the developer to believe everything is working correctly without fully testing each phase of development due to the green lights of the unit testing. This was prevented by carrying out full functionality testing throughout, and the results being constantly monitored in order to ensure the project was on track and everything worked correctly. This isn't documented, but can be seen throughout implementation and experimentation within chapter 4.

5.1.3 List of Known Bugs

Reference table of severity levels used within this section:

- **Blocker** = a huge problem that has to be addressed as functionality is impacted greatly
- **Major** = big problem that should be addressed
- **Minor** = small issue that most likely can be overlooked
- **Trivial** = if a solution happens to be found add it but no time needs to be spent fixing this due to the small impact

---

Listed below are all of the known bugs within this version of the project. Included is a severity level of the bug (as referenced above) and any action that has been taken to prevent the bug affecting the functionality:

- Data Scrape
  - **Major:** When scraping data from allrecipes.com some non-ingredients can be picked up and stored as an ingredient.
  - **Trivial:** Some reviews and star rating numbers were slightly out of date due to time passing after collecting the data.
- Data Processing
  - **Blocker:** Can't run unless the NLTK library is installed - this could cause major issues when distributing the suite of programs.
    - *Action:* The executable files have been removed due to not working on other systems and a new README file was created to guide users through the installation of all necessary libraries and Python. This is a temporary fix and hopefully a better way in which to distribute the executable files can be found.
  - **Minor:** If the user wanted to stop mid way through the processing of the data nothing is stored and therefore can't be used. This could cause issues as with more data than the sample 5000 recipes, it would take a lot longer and require more resources to process these.
  - **Minor:** Some of the ingredients could benefit from additional processing and grouping due to being either overly specific or having additional information that needs to be removed.
- Final Application
  - **Blocker:** Can't run certain aspects of the program unless the NLTK library is installed - this could cause major issues when distributing the suite of programs.
    - *Action:* The executable files have been removed and new README files have been created.

- **Major:** If the user clicks outside of the program or processing becomes too intensive the program will crash/turn and not respond until the processing is complete. The user is not informed of what is happening and it looks as though there is an issue with the program. This is purely a graphical problem.
- **Major:** The rating guessed by the program has a habit of being wrong. This is due to the machine learning algorithm processing and the splitting of the data. This can be rectified by adding different processing methods and other forms of learning tools.
- **Trivial:** No instructions provided for running the program, this would be useful for those that are less experienced with using computers.

5.2 Results

Some information was discussed within testing, however, the main focus of this project was on the machine learning aspect. Table 5.4 shows the different numbers of recipes being processed, and the percentage of accuracy found, after running the program over the dataset containing more than 76,000 recipes within it. The average percentage of accurate ratings found within this project was 77.8%.

No. of Recipes	Test 1	Test 2	Test 3	Average
1,000	80.0%	78.0%	82.0%	80.0%
2,000	78.0%	79.0%	75.0%	77.3%
5,000	75.8%	77.0%	74.4%	75.7%
10,000	77.8%	78.0%	78.0%	77.9%
Average	77.9%	78.0%	77.4%	

Table 5.4: Table showing the results for the accuracy rating of various numbers of recipes.

There are ways in which this could be improved and these are addressed within a section about future work, section 6.5. These will be implemented in the near future and hopefully will provide a more accurate view of recipes and the real world rating of these.

Chapter 6

Critical Evaluation

6.1 Analysis & Research

The project goals, research and analysis were extensive sections of this project; this allowed for a lot of pitfalls to be considered and avoided, libraries to be targeted and goals to be set. With the additional knowledge of the area, the forecasted issues and implementation could be designed well without any problems. This did help streamline the process, target the areas that would require more effort, identify the requirements and provide basic methods that could be used to assist the design of a new system.

The choice of Python was excellent as it had everything required for the task and assisted the entire process. The NLTK and SKLearn were huge assets in the program, and using the simple scripting language allowed for time to be spent on the actual problem rather than debugging the code.

FDD was a great choice for the development process, proving extremely useful with its flexibility and ability to prioritise the approaches as they happened. Sometimes this approach did provide too much flexibility; this was seen within the amount of time spent on the feature extraction and pre-processing of the data. This was important and the time was spent wisely, however, with more time spent elsewhere the system could possibly have less bugs and an Area Under Curve (AUC) implemented (discussed in section 6.5) or other experimentations could have been applied.

6.2 Data Collection & Storage

Data Collection was one of the strongest parts of this project. The data was legally scraped from allrecipes.com, information extracted and then stored. The collection of this data was well managed and efficiently allows more data to be gathered without any issues.

Data Storage however, caused some problems. The choice to use a file system rather than a local or hosted server was not a good idea for the amount of data that ended up being scraped and then used. The processing and feature extraction didn't work as well as expected and therefore the files became much larger than was first thought. The initial 5000 recipes that will be shipped with the program currently isn't too big but once more data and more recipes with unique ingredients appear the dataset becomes massive. The reading and writing of this data to files is quite long in

comparison to using a server and SQL rather than storing in memory. The problem with reading and writing from a file also means that everything is stored in-memory for the processing of the recipes. This causes a lot of problems due to there being heavy memory usage when running.

The overall programming of this isn't too bad as it works and accurately processes and stores everything. However, in hind sight the additional time could have been spent on creating a much more versatile system that will allow for as much data as is possible to be processed and stored without any issues. If the project were to begin again, this would be one of the big aspects that would be changed.

6.3 Feature Extraction

This was the real downfall of this project. Feature extraction caused a lot of problems throughout the programs; this is evident within the number of 'unique' ingredients that are extracted. A lot of time and effort was put into attempting to find the best way in which to find the ingredients, separate them from the measurements (this was fairly successful) and then also group the similar ingredients together (this wasn't successful). With more time and access to more in depth papers and discussions with data scientists, significant improvements would be made.

The NLTK libraries were used and extensively researched. These helped tremendously and without them the issues would have been far worse than at the moment. This could be worked on further and is discussed within future work (within section 6.5).

6.4 Performance & Algorithm Choice

The performance of the machine learning algorithm wasn't as good as expected, with the results detailed within section 5.2 at just 77.8%. This is due to multiple aspects of the program: the algorithm itself, the pre-processing and the split of classification of the rating class. All of these are addressed within future work (section 6.5) and the current progress is promising.

The choice made to use the Bayesian method was possibly the wrong choice. The method does work but due to not having the dependencies of the recipes and the mixtures of the ingredients not being assessed, a lot of mistakes can be made from the algorithm. The research showed that neural networks were a good starting point and Random Forest was identified as another possibility. These possibly would have been a better choice, but the implementation time of these and the processing power that would have been required during the use of these would have needed a much more complex system. With even the Bayesian method causing some issues with processing power the right choice was made. This method provided a great starting point that allowed an entire system to be built around it. The results, although not outstanding, do still show a lot of promise.

The overall performance and all of the program's requirements were met, the exception being the non-implementation of an additional machine learning approach. These are addressed within section 6.5 and will hopefully be added very soon.

6.5 Future Work

So far the work put into this project has been extensive. Even with some fairly poor results within testing the accuracy of the algorithm some really promising aspects appeared, and there could be some really groundbreaking work done if a little more time was put into this. The process could be applied to many different areas, not just recipes.

- Working Executable Files
  - In order to provide the best user experience and one that people will want to use, you need a program that requires very little effort to run. Currently, due to the need to install Python, install the libraries and then run the program a lot of users would likely be deterred. Therefore, the need for working executable files that are reasonably small with all of the information would be ideal. This is very important issue that could and should be addressed.
- Additional Natural Language Processing
  - To create the best program, a program that will not only output accurate results but do it in a fast and efficient way, the NLP needs to be spot on. This was a real issue during the development of this process and with each iteration of tests it was found that the one process that really affected the entirety of the program was the NLP. Any changes made to this would decrease the memory used and the number of unique ingredients whilst quickening the calculation and pre-processing and increasing the accuracy. Adding something that would create a way in which to bundle similar ingredients together would create a much better model. This is a key point to address and during further experimentation there will be a lot of work and research dedicated to this.
- Addressing Memory Problems
  - Due to the vast amount of data required for this to work efficiently and well there needs to be a good memory system in place. This could be addressed by rewriting sections of the program in order to manage the allocation of memory rather than allowing Python to automatically garbage collect. This could increase the usability, stop program crashes, quicken up the process and allow for machines that may not have the same power as the one used for this project to use this program without issues. This is a key factor and should be addressed in the near future to allow for more recipes to be processed, more data to be collected and better results to be outputted.
- Adding an Area Under Curve (AUC)
  - *What is AUC?* Receiver Operating Characteristic (ROC) is the default way in which a machine learning algorithm finds the result from a graphical plot. This is the plotted False Positive Rate against True Positive Rate and shows as a line of best fit. AUC however is a curved line that better represents the data and the result set. The decision between the two lies mainly on the idea of either minimizing False Positive Rate (ROC) or maximizing True Positive Rate (AUC).
  - *Why is this important?* The curved line of best fit will better classify what is and isn't a good recipe based on the more specific numbers and features that it requires. Adding

this could improve the results drastically and would help with the accuracy within. Adding this could help but it would also be interesting to see the difference between the results from each method.

- Better Classification of Data
  - Having the data set split accurately providing a more diverse data set would be ideal for accurate and non-bias rating predictions. Currently the rating predictions are static with a rating above 4 stars being good, 3-4 stars being ok and under 3 being classed as bad. This isn't necessarily a bad thing due to these being fairly accurate to the real world, however, most of the data set seems to be biased to being good recipes. This classifies most of the recipes as good and therefore the accuracy of the rating is skewed by this fact. Adding a dynamic splitting of the data would really increase the real world usage of this project.
- Additional Machine Learning Algorithms
  - As part of the original design and analysis there were a lot more machine learning algorithms available that could have added additional value to this project. With more research, different processing and the additions of these new machine learning algorithms the program could do a lot more, classify more and add more interesting facts. Having additional views on the data would allow for comparisons to be drawn and possibly reveal different results. This would be a really exciting and valuable resource that would be great to look at in the future.
- More User Interaction
  - Currently user interaction is good. The user can choose the number of recipes, enter their own recipe, add additional data and control what is happening. Adding more user interaction would be exciting. If the user could choose the training/testing ratio and then be able to view the difference in accuracy ratings there would be more understanding of how the results were achieved. Another possible addition would be having a GUI that runs throughout; this could possibly allow users to download data, process the data, load the data, train the data and then add their own recipe within the same interface. Having that accessibility may encourage users to explore more data and update this as they use the program so that they can test against newer recipes without the hassle of using command line and running a different Python file. This, alongside additional machine learning algorithms would add a lot of functionality and ease of use for the end user.

Appendices

Appendix A

Third-Party Code, Libraries & Technologies Used

A list of all libraries and technologies used can be found below with a short description of why each was used and where copies of the libraries were found.

Python - This is the language used for all scripts for this program to run. Version 2.7.10 was used. With the exception of the use of libraries listed below all of the code was written by the developer.

lxml Library - The library has been used to read XML and HTML from the allrecipes.com website. This acts as an API for the XML and allows for the gathering of necessary information. The library was used without modification, and is available here: http://lxml.de/index.html.

random Library - This library allowed for random numbers to be used during testing so that a varied array of recipes could be accessed via the lxml library. The library was used without modification, and is a default package within Python.

requests Library - This library was used so that the URL's could be accessed via the Python script, this was used in conjunction was the lxml library during the scraping of the data. The library was used without modification, and is a default package within Python.

csv Library - This library was used throughout the project in order to both read and write to csv files efficiently. The library was used without modification, and is a default package within Python.

os Library - This library was used in order to check whether a file existed so that appropriate headers could be added to the file. This allowed for dicts to be used within the pre-processing section of the project. The library was used without modification, and is a default package within Python.

collections Library - This library was used in order to create dicts of its own, therefore a recipe could be called upon via its own dictionary of lists. This was necessary in order to read the .csv file in a organised way. The library was used without modification, and is a default package within Python.

re Library - This library allowed the functionality of REGular EXpressions (REGEX) to be used so that the recipes could be processed efficiently and properly. The library was used without

modification, and is a default package within Python.

NLTK Library - The library was used for the Natural Language Processing section of this project. The main sections from within this library that were used were as follows: WordNetLemmatizer from within Stem and wordnet from within Corpus. The Lemmatizer was used in order to stem words to their original format for pre-processing the data. The wordnet aspect was used in order to gather a list of nouns and reduce the number of ingredients by removing descriptive words. The library was used without modification, and is available here: http://www.nltk.org/install.html.

Tkinter, **tkMessageBox** and **ttk** - Tkinter and associating libraries were used to create a Graphical User Interface easily and with already created tools. This was used for the final application and allows for the best usage of the program for users of various abilities with appropriate message boxes to ensure correct usage. The library was used without modification, and is a default package within Python.

sklearn - This library, sci-kit learn, was used to create models for the data. This allowed for analysis and predictions to be made. Modules for Naive Bayes were readily available to use and additional models were able to be used if necessary. The library was used without modification, and is available here: http://scikit-learn.org/stable/install.html.

pyinstaller - This library was used to create executable Python files. This allows users to run the program without the need for Python libraries to be installed. The library was used without modification, and is available here: https://sourceforge.net/projects/pywin32/files/pywin32/Build%20217/

Appendix B

Ethics Submission

Ethics number:4284

Below is the ethics form embedded into this document.

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
cht26@aber.ac.uk

Full Name
Charlotte Taylor

Please enter the name of the person responsible for reviewing your assessment.
Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your application
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
MMP - Mining Online Food Recipes

Proposed Start Date
24th January 2016

Proposed Completion Date
4th May 2016

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
My project, "Mining Online Food Recipes", will look at gathering recipes and data from online sources – specifically allrecipes.com. Machine Learning techniques will then be applied to the semantically processed data to determine the rating of the recipe based on similar ingredient combinations.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Yes

Will appropriate measures be put in place for the secure and confidential storage of data?
No

Does the research pose more than minimal and predictable risk to the researcher?
No

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?
No

Please include any further relevant information for this section here:

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.
Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.
Yes

Please include any further relevant information for this section here:
I have contacted allrecipes.com and they have allowed me to scrape and use their data for this project.

Appendix C

Code Examples

3.1 HTML Tags from allrecipes.com

Within listing C.1 it shows the original HTML tags that were used in order to gather the information. Knowing where and how the information is stored and presented on a web page makes the data scraping a much easier task.

```
<h1 class="recipe-summary__h1" itemprop="name">Best Chocolate Chip Cookies</h1>
<meta itemprop="ratingValue" content="4.60">
<meta itemprop="reviewCount" content="7612">
```

Listing C.1: A few examples of the inner HTML from allrecipes.com that had to be collected.

3.2 Removing any Recipes Without Star Ratings

Within listing C.2 the process of ensuring only the recipes with one or more ratings are received. The *raters* variable holds a list of [number of reviews, number of ratings]. When originally captured from the website it is received as, for example: ['3 reviews', '11']

This only takes those recipes that have actually been reviewed, which removes any issues with people who just rate without actually trying the recipe. To access the actual number of reviews, the first element within the *raters* variable is split and the number is retrieved.

```
#Uses the recipe ID to cycle through each recipe on the website
url = 'http://allrecipes.com/recipe/' + str(recipe_num)
#Adds a timeout of 10 seconds to prevent crashing of the program
page = requests.get(url, timeout=10)
#The tree structure gets all of the available html and contents from the
    webpage
tree = html.fromstring(page.content)
#Gets the number of people who have rated the recipe
raters = tree.xpath('//span[@class="review-count"]/text()')
try:
  num_raters = raters[0].split(' ', 1)[0] #need at least one review to remove
    any recipes that aren't popular and can't be judged to be good/bad
  if num_raters > 0:
    ...
```

```
except:
  print "Recipe not found: " + str(recipe_num)
```

Listing C.2: The necessary removal of any recipes that didn't have any star rating.

3.3 Pre-Processing Data Formatting for Machine Learning Purposes

This shows the initial structure processing for the use within the final application and within the machine learning process. This structures the data so that it becomes a list, with a reference, of ingredients for each recipe that it does and doesn't contain. This can be seen within listing C.3.

```
#Adds additional reference ingredients
for i in range(len(REFERENCE_INGREDIENT)):
  INGREDIENT[recipe_num].append(0)

#Creates a list of [0,1,0,1,0,0] as whether the recipe holds certain
    ingredients or not, in preparation for the machine learning algorithm
#These lists can get very long...
for i in range(len(ingredient_list)):
  try:
    ref_num = REFERENCE_INGREDIENT.index(ingredient_list[i])
    INGREDIENT[recipe_num][ref_num] = 1

  except:
    REFERENCE_INGREDIENT.append(ingredient_list[i])
    INGREDIENT[recipe_num].append(1)

...

#At the end of processing all of the ingredients an additional processing for
    this is carried out in order to make all of the recipes equal lengths (a
    requirement to fit datasets to machine learning algorithms)

#Adds zeros to the end of the recipes to make them the same size
for i in range(len(INGREDIENT)):
  current_length = len(INGREDIENT[i])
  need_adding = length_ingredient - current_length
  additions = [0] * need_adding
  INGREDIENT[i].extend(additions)
  print "Finalised: " + str(i) #Updates user on what is happening
```

Listing C.3: The necessary processing for the structure of the data within the Naive Bayes model.

3.4 Grouping the Recipe Ratings

This grouped the ratings of recipes in order to classify the predictions and output a result. These displayed the group of results possible. Throughout the development of this program these groupings were modified and tested and the best result that provided the user with an adequate amount of data was found to be the: good, ok, bad groupings. This was accurate enough and provided more information than the: good and bad groupings and was far superior to the five groupings found within the star rating groups. This can be seen within listing C.4.

```
#Splits the recipes into easier chunks - ready to be naive bayes'd

#Checks decimal value from file and turns into a good/bad/ok rating
#May need fine tuning to generalise based on the number in each category
for i in range(len(RECIPE)):
  if RECIPE[i] >= 4:
    RECIPE[i] = 0
  elif RECIPE[i] < 3:
    RECIPE[i] = 2
  else:
    RECIPE[i] = 1
```

Listing C.4: The grouping of ratings results in order to provide the best and most accurate result set from the features provided.

3.5 Splitting the Dataset into Training and Testing Data

The code within listing C.5 shows the random splitting of the data so that training and testing can be completely unbiased from any external factors.

```
splitratio = 0.9 #change this to change the split of training vs test

training_size = int(len(INGREDIENT) * splitratio)
training_set = []
testing_set = []

for i in range(len(INGREDIENT)):
  testing_set.append(i)

#Randomly assigns whether a recipe will be part of the training or testing
    dataset
while len(training_set) < training_size:
  index = random.randrange(len(testing_set))
  training_set.append(testing_set.pop(index))
```

Listing C.5: The splitting of the dataset.

Appendix D

Additional Information

4.1 Email Conversation with allrecipes.com Regarding Legal Implications of Data Scrape

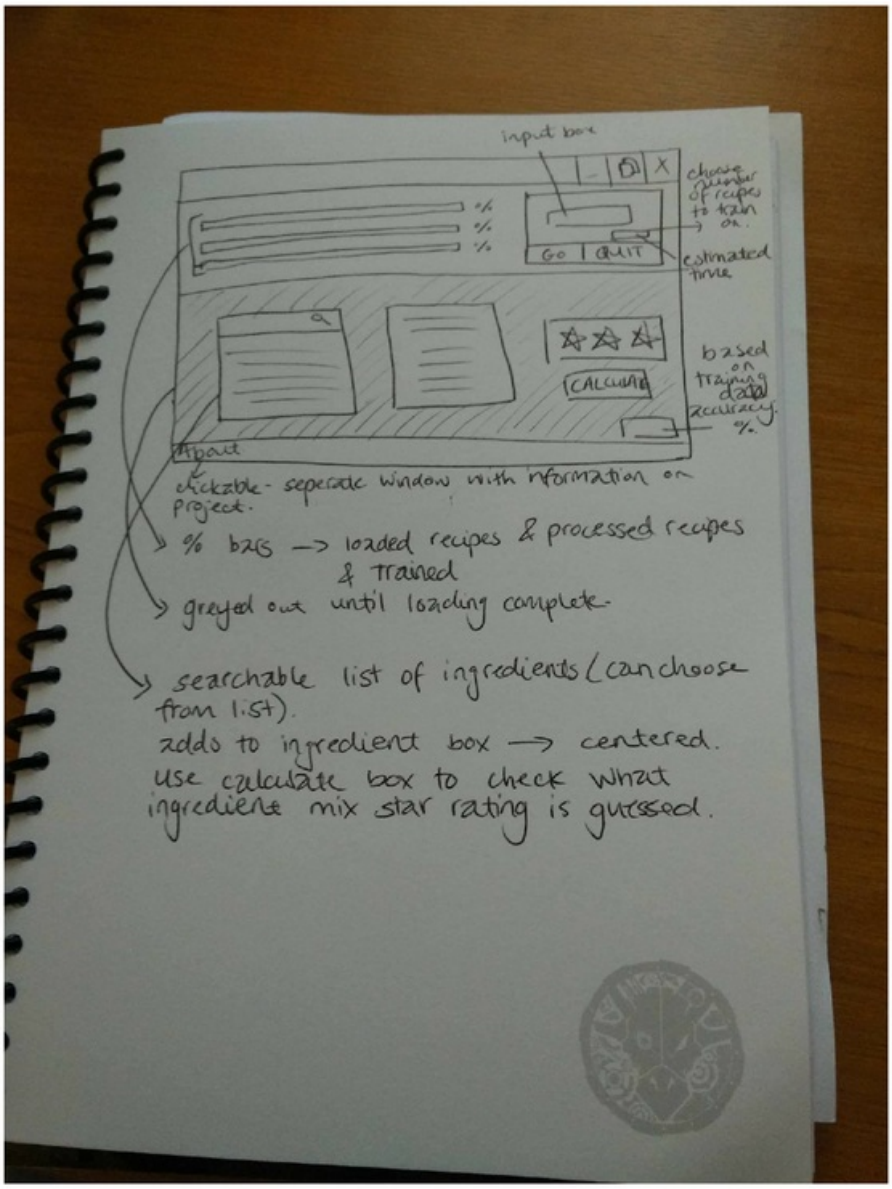
The email conversation below was a discussion about the legal implications of the data scrape process on the website allrecipes.com. The recipe website has a copyright clause and prevents users from legally scraping the information from the website. This was noticed and a conversation arose about the educational purposes rather than competitive or using the data maliciously.

This ended with confirmation of the use of the data and the gathering of this and aided the project greatly.

NOTE: The following three pages have been removed
because it included some personal information in the communication with the company.

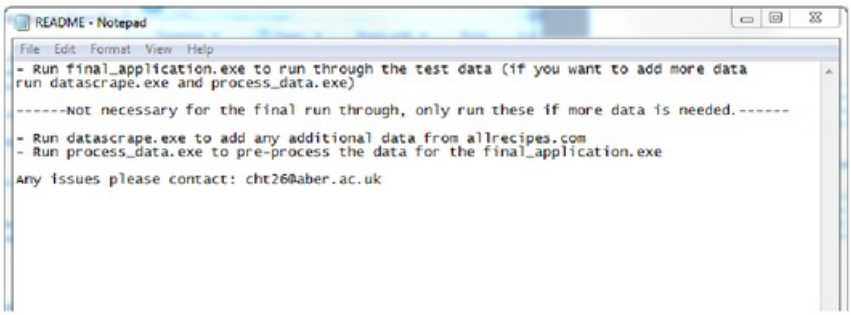
4.2 Sketches of Initial Graphical User Interface Design

Below shows the initial sketches of the GUI that were created as the visualisation of the product occured. They show some additional notes that were made alongside in order to show some functionality. These are early ideas and some of the information may not be accurate as to the final product.

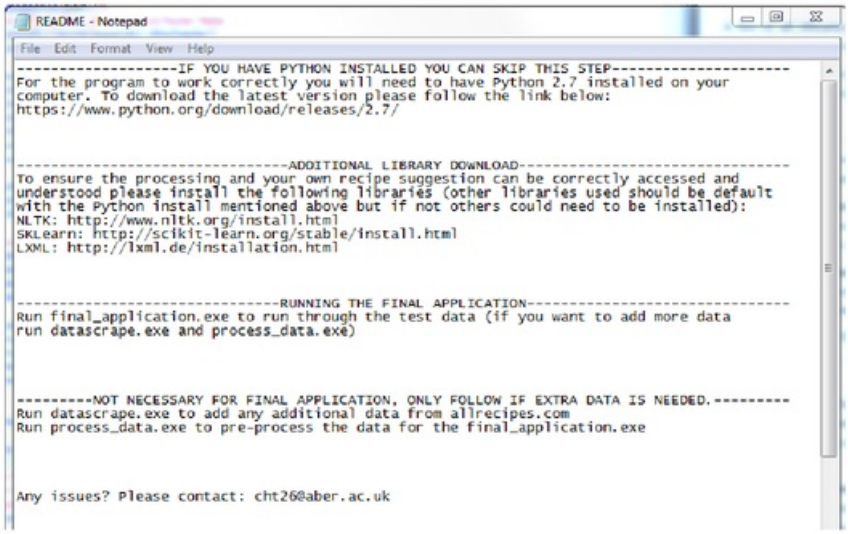


4.3 README

The Original README file that gave directions to the Quality Assurance tester was as follows:



After feedback, as seen within appendix 4.4, the README was changed in order to give more comprehensive and accurate instructions. This included changing from using the .exe files and also showing instructions to follow so that the Python file and incorporated libraries are correctly downloaded before attempting to run the program, these are as follows:



4.4 Black Box Testing of Final Programming Suite

The final suite of programs were sent to a professional Quality Assurance tester in order to find an impartial and professional view of the program. This allowed for user testing and testing on another machine to occur with the .exe files and revealed a few issues that would not have been caught otherwise. This can be seen within the document below:

Data Scrape White-box Test Report

Application functionality

The Data Scrape application passed core functionality checks:

- **Initialization** - Application runs as expected.
- **Function** – Application correctly gathers recipe data.
- **Feedback** – Application outputs basic recipe data upon completed writing of data.
- **Result** – A sanity test of gathering 100 recipes was performed with 99 successful entries and 1 failure.
- **Verification** – A sample of 5 entries, including the failure were verified to be accurate.

Data Set Verification

A sample data set of 5,000 entries were provided for verification. Due to the simple repetitiveness of and lack of variables in the data being collected the verification sample size of 10 entries was deemed adequate.

Each sample was subject to a physical comparison of the data recorded to *recipe*.csv versus the data visible on the recipe's web page with all 60 tests passing, as seen below.

Sample	Recipe ID	Recipe Name	Number of Ingredients	List of Ingredients	Star Rating	Number of Raters
7246	P	P	P	P	P	P
7372	P	P	P	P	P	P
7845	P	P	P	P	P	P
8993	P	P	P	P	P	P
9326	P	P	P	P	P	P
9662	P	P	P	P	P	P
9706	P	P	P	P	P	P
10827	P	P	P	P	P	P
11375	P	P	P	P	P	P
11541	P	P	P	P	P	P

Destructive & Exploratory Notes

Additional samples of failed entries were selected for verification, with each case successfully resulting in no recipe being found. Though none were found in testing it is deemed possible that writing of data may fail if the webpage fails to respond or times out, resulting in data being missed.

Processing Data White Box Test

Application Functionality

The Processing Data application failed to initialize due to a Lookup Error, found below. A resource within the natural language toolkit wasn't available so all testing on the application was aborted.

```
Traceback (most recent call last):
  File "<string>", line 11, in <module>
  File "site-packages\nltk\corpus\util.py", line 99, in __getattr__
  File "site-packages\nltk\corpus\util.py", line 64, in __load
LookupError:
**********************************************************************
  Resource u'corpora/wordnet' not found.  Please use the NLTK
  Downloader to obtain the resource:  >>> nltk.download()
  Searched in:
    - 'C:\\Users\\john_colgrave/nltk_data'
    - 'C:\\nltk_data'
    - 'D:\\nltk_data'
    - 'E:\\nltk_data'
    - 'C:\\Users\\JOHN_C~1\\AppData\\Local\\Temp\\_MEI26~1\\nltk_data'
    - 'C:\\Users\\JOHN_C~1\\AppData\\Local\\Temp\\_MEI26~1\\lib\\nltk_data'
    - 'C:\\Users\\john_colgrave\\AppData\\Roaming\\nltk_data'
**********************************************************************
process_data returned -1
```

Final Application White-box Test

Application Functionality

The application successfully passed initialization, function & feedback tests but due to being unable to locate a National Language Toolkit resource results could not be obtained or verified.

- **Initialization** - Application runs as expected.
- **Function** – Application correctly loads and processes the requested recipes.
- **Feedback** – Application displays the amount of recipes in loading and processing live. Application is able to predict the accuracy of the training data using the machine learning algorithm and visibly display this information.
- **Result** – Ingredients could not be processed due to the below error.
- **Verification** – Due to results failure, verification was not conducted.

```
LookupError:
**********************************************************************
  Resource u'corpora/wordnet' not found.  Please use the NLTK
  Downloader to obtain the resource:  >>> nltk.download()
  Searched in:
    - 'C:\\Users\\john_colgrave/nltk_data'
    - 'C:\\nltk_data'
    - 'D:\\nltk_data'
    - 'E:\\nltk_data'
    - 'C:\\Users\\JOHN_C~1\\AppData\\Local\\Temp\\_MEI77~1\\nltk_data'
    - 'C:\\Users\\JOHN_C~1\\AppData\\Local\\Temp\\_MEI77~1\\lib\\nltk_data'
    - 'C:\\Users\\john_colgrave\\AppData\\Roaming\\nltk_data'
**********************************************************************
```

Annotated Bibliography

[1] "Bigoven api." [Online]. Available: http://api2.bigoven.com/web/documentation/feestructure

"This pricing structure shows the cost of using APIs on recipe websites, this was a possible route for the development phase."

[2] "bitarray 0.8.1 : Python package index." [Online]. Available: https://pypi.python.org/pypi/bitarray/0.8.1

"This library turns largely memory intensive processes into strings of bits rather than bytes. This was looked into due to memory issues within the process."

[3] "Cloud servers pricing." [Online]. Available: http://www.rackspace.co.uk/cloud/servers/pricing

"This webpage shows the pricing of racks of cloud servers that were considered during the development phases of this project."

[4] "Database." [Online]. Available: https://cloud.oracle.com/database?tabname=pricinginfo

"This pricing structure looks at the cost of the average cloud server that could be used as a platform to run the programs on due to some memory issues."

[5] "Deep belief networks." [Online]. Available: http://deeplearning.net/tutorial/dbn.html

"This shows a tutorial that deciphers what a deep belief network is and the differing implementations of this."

[6] "Natural language processing." [Online]. Available: https://rpubs.com/lmullen/nlp-chapter

"This article looks at the natural language processing functionality and various libraries for different programming languages and the best usage of these languages."

[7] "R: Memory limits in r." [Online]. Available: https://stat.ethz.ch/R-manual/R-devel/library/base/html/Memory-limits.html

"This webpage shows the memory limits within R and the issues it has with running intensive processes due to only being able to use virtual memory on a machine."

[8] "Wordnet interface." [Online]. Available: http://www.nltk.org/howto/wordnet.html

"This documentation on Wordnet, a part of the NLTK library explains the different uses and corpus' that are available within this package. It details how to use them and gives examples so they are used correctly."

[9] Y.-Y. Ahn, S. E. Ahnert, J. P. Bagrow, and A.-L. Barabasi, "Flavor network and the principles of food pairing," Dec 2011. [Online]. Available: http://www.nature.com/articles/srep00196

"This paper discusses the value of focusing on diversity within cultural choices for meals. It looks at the biological aspects of food and the flavour compounds within ingredients and uses neural networking to link ingredients together based on these biological parts."

[10] Bloomberg, "Allrecipes.com, inc.: Private company information." [Online]. Available: http://www.bloomberg.com/research/stocks/private/snapshot.asp?privcapid=427563

"This stock snapshot details the life of allrecipes.com as a company. It shows when it was founded, its address and the current hierarchy within."

[11] J. Brownlee, "Practical machine learning problems - machine learning mastery," Nov 2013. [Online]. Available: http://machinelearningmastery.com/practical-machine-learning-problems/

"This article shows the wide varying range of machine learning problems that are currently being tackled within the industry. This shows the extent of the relatively new data craze that is impact almost all business sectors."

[12] L. Burtch, "Predictions: 2015 analytics and data science hiring market," Jan 2015. [Online]. Available: http://www.kdnuggets.com/2015/01/predictions-2015-analytics-data-science-hiring-market.html

"This article looks at the predictions for Big Data that was set from January 2015 and how the evolution of the business world requires the evolution of the data world. It details how the new job that will be in high demand will be the data scientists that will spotlight the prediction models and data-driven decision making."

[13] J. Cooper, "Cooking trends among millennials: Welcome to the digital kitchen," Jun 2015. [Online]. Available: https://www.thinkwithgoogle.com/articles/cooking-trends-among-millennials.html

"This google article shows the trends within the modern day kitchen. The article shows the gaps within the market for digital recipes and the growing popularity of paperless cooking."

[14] G. Emmanuelle, N. Emmanuel, and W. David, "A event of the international conference on case-based reasoning (iccbr 2015)," 2015. [Online]. Available: http://computercookingcontest.net/

"This site shows a previous competition for the computational artificial intelligence, machine learning based cooking."

[15] D. Gould, "2011 trends: Recipe websites, apps, & publishing," Dec 2011. [Online]. Available: http://foodtechconnect.com/2011/12/28/trend-report-2011-recipe-websites-apps-publishing/

"This article shows the new trends of users using recipes, it shows the new searching algorithms on recipe websites and the new uses of machine learning within the food industry to find new and exciting combinations of ingredients."

[16] T. Hastie, "Trees, bagging, random forests and boosting." [Online]. Available: http://jessica2.msri.org/attachments/10778/10778-boost.pdf

"These slides from Stanford University show the different types of trees and random forests used for machine learning purposes. It explains how they are used, what they do and how they do it."

[17] J. Milutinovich, "What is r?" [Online]. Available: http://www.inside-r.org/what-is-r

"This article explains what the programming language R is, and overviews what it is useful for and the surrounding community."

[18] L. Moses, "Food porn is the new click bait for audience-hungry publishers," Jan 2016. [Online]. Available: http://digiday.com/publishers/food-porn-click-bait/

"This article discusses the new trend of videos and images being used to entice users to click on their advertisements in order to create revenue. This details websites such as Buzz-Feed and their Tasty and Food channels and looks at the traffic caused by the perfect shot of food and how this impacts peoples lives."

[19] V. Nedovic, "Learning recipe ingredient space using generative probabilistic models," 2013. [Online]. Available: http://flavourspace.com/the_story_behind/wp-content/uploads/2014/07/flavourspace-paper-ijcai2013.pdf

"This paper looked at the varying models, including deep learning, for probabilistic models to be based from. This was presented as a networking idea for mapping cuisines from across the world in order to find good ingredient combinations."

[20] C. Nguyen, Y. Wang, and H. N. Nguyen, "Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic," May 2013. [Online]. Available: http://file.scirp.org/html/6-9101686_31887.htm

"This paper looks at Random Forest classifiers within the realm of cancer diagnosis. It uses the trees to develop a way in which to diagnose certain breast cancers in women."

[21] M. Nielsen, "Using neural nets to recognize handwritten digits," Jan 2016. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap1.html

"This paper looks at neural networks and uses handwriting recognition as a way to demonstrate how they work."

[22] A. Potapenko and K. Vorontsov, "Robust plsa performs better than lda," 2013. [Online]. Available: http://www.machinelearning.ru/wiki/images/e/eb/voron13potapenko-eng.pdf

"This paper looks at the difference between two probabilistic algorithms (Robust PLSA and LDA). It details that Robust PLSA is a better version of the algorithm."

[23] D. Price, "Surprising facts and stats about the big data industry," Mar 2015. [Online]. Available: http://cloudtweaks.com/2015/03/surprising-facts-and-stats-about-the-big-data-industry/

"This article shows facts about the big data industry, including how much data is produced worldwide every day, the future of data and other interesting facts."

[24] L. Qian, Z. Bu, M. Lu, J. Cao, and Z. Wu, "Extracting backbones from weighted complex networks with incomplete information," Sep 2014. [Online]. Available: http://www.hindawi.com/journals/aaa/2015/105385/

"This paper looks at backbone extraction methods within complicated weighted networks. This finds a method to filter the edges that are kept or discarded within neural and deep networks so provide more efficient and effective methods."

[25] S. Raschka, "Naive bayes and text classification," Oct 2014. [Online]. Available: http://sebastianraschka.com/articles/2014_naive_bayes_1.html

"This article looks at text classification and the machine learning algorithm Naive Bayes. This shows the various types of Naive Bayes and how best to use its classifiers."

[26] J. Rost and A. Mooney, "How constantly connected cooks get ready for thanksgiving," Nov 2014. [Online]. Available: https://www.thinkwithgoogle.com/articles/how-constantly-connected-cooks-get-ready-for-thanksgiving.html

"This google article shows just how many people search for recipes and when the most recipes are searched for during the year. The popularity of online recipe searching and finding the best recipe can really be seen within the article, also the trend of devices used for recipe searching."

[27] SAS, "Big data analytics: What it is and why it matters," 2016. [Online]. Available: http://www.sas.com/en_us/insights/analytics/big-data-analytics.html

"This page discusses the importance of Big Data within history, evolution and how it is relevant within the world today. It addresses business concerns over money, decisions and how products and services can be used and created to enhance their business life."

[28] E. Schadt, "The role of big data in medicine," Nov 2015. [Online]. Available: http://www.mckinsey.com/industries/pharmaceuticals-and-medical-products/our-insights/the-role-of-big-data-in-medicine

"This article poses the question of 'Evolution or Revolution' via the role of Big Data within medicine. This shows the new data-driven approaches that have begun to transform medicine via predictive modelling over medical information."

[29] C.-Y. Teng, Y.-R. Lin, and L. A. Adamic, "Recipe recommendation using ingredient networks," May 2012. [Online]. Available: http://arxiv.org/pdf/1111.3919.pdf

"This paper looks at ingredient combinations, and the comments on ingredients that say things like 'swap x for y' and correlates the ingredients that can be interchanged. This then is used to develop a network and provide good recipe and ingredient recommendations."

[30] M. Theuwissen, "Kdnuggets," May 2015. [Online]. Available: http://www.kdnuggets.com/2015/05/r-vs-python-data-science.html

This article looks at R vs Python programming languages within data science. It looks at the popularity, the average salary of a programmer who has knowledge of these and the best usage of each language.

[31] B. Venners, "The making of python: A conversation with guido van rossum, part i," Jan 2003. [Online]. Available: http://www.artima.com/intv/pythonp.html

"This paper shows a conversation had between a journalist and the creator of the Python language and the development of the pre-Python program that followed on with the Python scripting language that developed into such a huge success."

[32] B. Walker, "Every day big data statistics," Apr 2015. [Online]. Available: http://www.vcloudnews.com/every-day-big-data-statistics-2-5-quintillion-bytes-of-data-created-daily/

"This article has a huge array of facts about the big data industry, including interesting facts about its uses within 2015."