G52GRP Interim Group Report

gp09-jqg*

November 30, 2009

Contents

*Dhruv Gairola, Chris Head, Rob Miles, Amy Jane Wesson and Chenjue Xu, supervised by Julie Greensmith

1 An Introduction to the "Problem"

The aim of this project is to develop a software kitchen assistant tool. This tool must be able to provide recipes which match a supplied list of available ingredients. This is similar to the BBC's recipe search[1] (Fig 4), but is not constrained to just three search items. The software should, in essence, provide recipe matches to help the user make recipe decisions given the ingredients he/she currently has. This involves making recommendations and provides us with an avenue to make use of collaborative filtering technology. Additionally, the recipe database can be community maintained, and aspects of social networking can be implemented to encourage user participation. The main draw of the project lies in its inherent flexibility, which extends from the plethora of expansion decisions which can be made to enhance user experience.

[1] `http://www.bbc.co.uk/food/recipes/`

2 Product Specification

In response to our initial meeting with our client and after reviewing the "Problem Description", we have put together a brief specification stating the requirements for the software at three levels which have been agreed by our client. The levels can be considered as versions as they will be implemented in this way (i.e. Minimum - version 1, Realistic Best - version 2 and Ideal - version3 [2]). These requirements will be used as a basis for the design of the respective versions and the designs will be created with these requirements in mind.

2.1 Minimum - v1

The requirements listed below are the minimum requirements to make the software work which does not include extra functionality desired by our client. This will be the version 1 of our prototype and we will use this as a base to build from. Furthermore, the creation of different versions with the intent to expand on previous versions will allow for continuity between languages and eliminating the need to 'start from scratch'.

- Contains several recipes in a database.
- Has a Web Interface.
- Recipes are searchable by ingredients (greater than 3 ingredients).

2.2 Realistic Best - v2

The requirements below are what we have identified as the most realistic outcome of the software within the time constraints. This would meet all the minimum requirements outlined by the client and some desirable ones as well. This will essentially be an expanded/upgraded version 1.

- Has a large database of recipes.
- Has a clean, attractive web interface.
- Has a seperate, simpler web interface for mobile devices.
- Recipes are searchable by combinations of ingredients.
- Has user accounts.
- Allows users to rate recipes.
- Gives recipe recommendations based on past ratings.

[2] It should be noted that due to the nature of creating a solution there are likely to be intermediate sub-versions e.g. 1.2, which will be used solely in the repository.

2.3 Ideal - v3

Below are the requirements for the software if it were to be the ideal solution to the problem. This list includes existing requirements from v1 and v2 as well. Note that it is likely that only few of these requirements will be implemented in the final version.

- Has a very large, automatically updated and maintained database of recipes.
- Has a clean, attractive and user-friendly web interface.
- Has a mobile device optimised interface and an iPhone application.
- Recipes are searchable by combinations of ingredients.
- Recipes are searchable by other tags: 'vegetarian', 'italian', 'low fat' etc.
- Has a full user system with profiles and user-uploaded recipes.
- Supports social media functions.
- Allows users to rate, tag and commment on recipes.
- Gives recipe recommendations based on past ratings and accumulated data from the entire user base.
- Gives recipe recommendations for several users i.e. 'A recipe that Alice and Bob both like'.

3 Results of Technical Research

For this part of the project we looked into many different alternatives for suitable platforms, tools, technologies, algorithms and data structures. We mainly conducted our research using the internet however, we did use some of our own personal experience and preferences aswell to influence our decisions.

3.1 Platform Decisions

Microsoft Windows

The windows operating systems have long dominated the operating system industry. Approximately 90 % of users use Windows operating systems, chiefly Windows XP followed by Windows Vista. Its ease of use and engaging graphical interface is certainly an attraction. However, precisely due to its widespread usage, Windows is the prime target for malware. Microsoft however, does provide bug fixes and other help to stabalise the system. Moreover, most forms of software run on Windows.

If our group is to market our product to customers, it makes sense that we focus on Windows as the platform of choice since it is the most commonly used operating system. Moreover, if our project decides to make an application, it should be able to run in Windows, and since most software works on Windows, it is the clear choice.

Mac OSX

Although not as widely used as Windows, this operating system has a very encouraging user interface which is easy to pick up. It is claimed as being more secure than Windows, due to its UNIX base. However, recent reports suggest the Apples Snow Leopard system is less secure compared to Windows Vista and XP[3]. Of course we must take into account the comparatively fewer threats from malware on Mac OSX. Mac OSX also uses pre-emptive multitasking for all native applications to which decreases the incidence of multiple program crashes.

Linux

One of the biggest advantages of Linux over other operating systems is the Linux kernel which ensures a basic level of security. Its hardware requirements are also much lesser than Windows and Mac OSX. Additionally, Linux, being open source is a free system. Linux distributions like Ubuntu, also provide a friendly and graphical user interface for users to work with. However, latest hardware is typically slower to reach linux. Moreover, depending on the distribution, the learning curve of Linux might be daunting for users [4].

[3][http://www.wired.com/gadgetlab/2009/09/security-snow-leopard/].

[4][http://packratstudios.com/index.php/2008/04/06/the-pros-and-cons-of-linux-windows-and-osx/]

3.2 Technologies

Django

A web framework based on Python language, Django is relatively easy to understand, Python being easier to program due to its natural language-like syntax. One of chief arguments for the use of Django concerned software reuse. Various existing libraries can be used to aid our software development efforts. The group software head also backed Django, and his recommendation was well received since the group could learn a new form of technology while benefiting from his expertise.

- Advantages
  - Our Technical Officer has experience developing with Django which is beneficial when developing and learning the language.
  - Python is an easy language to learn and use, with a focus on simplicity and ease of use whilst providing an elegant solution to the problem.
  - A variety of third party plugins coincide with our site's functionality, saving a lot of work by maximising code reuse.
- Disadvantages
  - Requires learning a new language.

Ruby on Rails

A web framework based on Ruby language, it allows users to create powerful applications using simple coding without compromising on the functionality of powerful languages[5]. The Rails framework also has many pre-defined libraries and functions that we may be able to use to our advantage.

- Advantages
  - Increased code reuse due to vast array of pre-defined libraries and functions available.
  - Also provides an esay and elegant solution to complex web programming problems.
- Disadvantages
  - Abstraction may mean sacrificing fine control even when it would be useful.
  - None of our group are familiar with the Ruby framework which may affect our pre-defined timetable and/or our time constraints.

[5][http://www.hosting.com/support/rubyonrails/faq/]

PHP with SQL

This option was an attractive one, considering that members had some experience with PHP and SQL previously. Moreover Java, Python, C++, Ruby are normally used to create complex systems which is not necessary for us at this stage in the project.

- Advantages
  - Overall group experience with these languages.
  - Common technology means it is well supported with many tutorials and guides on usage.
- Disadvantages
  - Low level control means making large systems is complicated and difficult.

3.3 Server setup

Why web based and not a downloadable app? The implementation decisions made concerned the web scripting language and web framework. We considered a variety of scripting languages to work with. For the prototype website, we decided to make use of HTML. It provides an ideal outlet for group members to acquire a feel of the project by working very simply and speedily. Other more complex client side scripting languages were considered for the prototype, namely Javascript and VBScript.(cons of html before pros) The web framework provides us a way to reduce overhead associated with web development, for example

4 Implementation Options/Designs

4.1 Summary of Project Description and Specification

The project description implies the implementation of some form of a database, either online (i.e. on a website) or offline (i.e. run on a local machine as an executable program). There are no specific details as to which language, layout or structure etc, are to be used when creating the solution. There are also no details as to which Operating System the solution should be created for and whether additional software/hardware is allowed. Considering this we have taken it open ourselves to discuss and choose what we thought was a suitable target platform and have also discussed availability of software/hardware needed to create the solution. During an initial meeting with our client concerning the problem specification/requirements it has become clear that the preferred solution is to create a website with a database backbone. This can be implemented in many different styles/languages which we have also discussed extensively.

We have created the Problem Specification (section 2 on page 4) with the intent that at each stage, the structure and format of the solution allows successive stages to be completed without changing the entire structure too much. This prototyping adheres to a large extent with the concept of Extreme Programming, where frequent releases introduce checkpoints where new customer requirements can be adopted. Additionally, the focus will be on upgrading versions of prototypes. To accommodate this dynamic character of our project, we need a framework which ties in the database with all the different elements of the website.

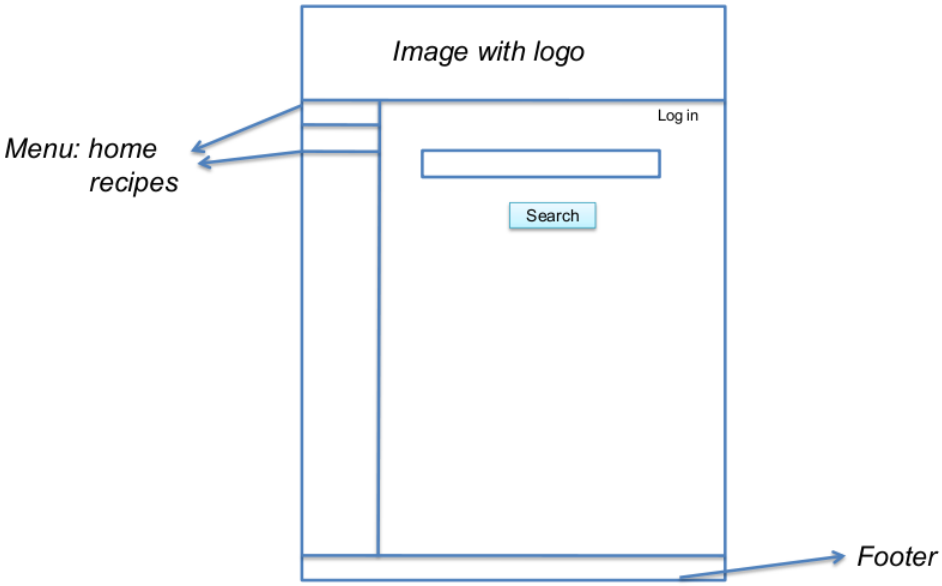

Figure 1: Layout of the Home Page

5 Initial design of the proposed system and its user interface

The version 1 website colour scheme is an amalgamation of white (for the background) and orange (for miscellaneous design features). The white background leverages on its simplicity and clarity, an important criteria for retaining the attention of the user. Moreover, the orange colour is reportedly stimulating for the users appetite. [6]. The group also agreed on an orange logo design.

5.1 Version 1

The merit of the version 1 website design is in its simplicity. Upon accessing the website, (Fig 1) users are presented there are three drop-down menus which allows people to select ingredients. The drop-down menus are transversely positioned to accommodate the vertical cascading of the ingredients of the menus. Users are provided with three menus to select ingredients and submit a search. This then links to the recipe list page. Additionally, the recipes button in the sidebar links the page to the complete list of recipes alphabetically.

The recipe list page (Fig 2) contains a list of recipes with at least one of the three ingredients. However, the recipe may contain other ingredients which were not specified by the user. Upon recipe selection, the specific recipe page will be displayed.

The recipe page (Fig 3) contains recipe details, for example recipe name, ingredients, the instruction, and recipe tags.

[6][http://desktoppub.about.com/cs/colorselection/p/orange.htm]

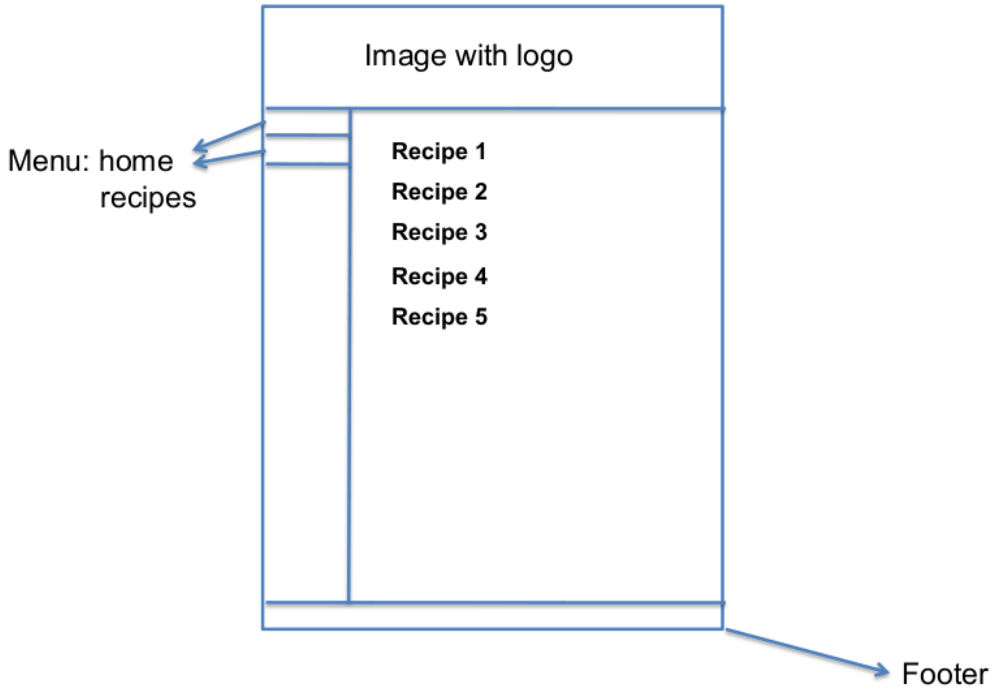

Figure 2: Layout of the Recipe List Page

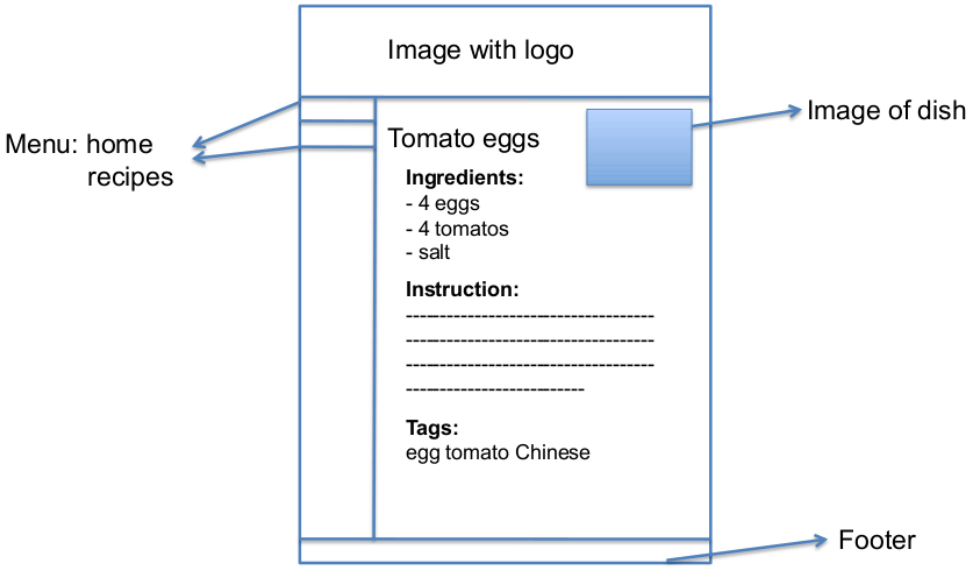

Figure 3: Layout of the Recipe Page

One design limitation of version 1 is that the website only has three drop-down menus, and the user is unable to type in ingredients. Being a prototype, the version 1 database is fitted with few recipes hence such functionality is redundant.

5.2 Version 2

Version 2 is an upgraded version of the original version containing more functions (described by the Product Specification). With the use of technology such as JAVA Script, the web interface will look more polished. Users will be able to enter text data into ingredient selection text boxes, which will have the tab completion feature for ingredients instead of using a drop-down menu. There will also be a larger database of ingredients hence justifying the use of tab completion as apposed to drop-down menus.

Additionally, for version 2, users are allowed to have web accounts. The benefits of the account include the user having access to past recipes which he/she rated and more importantly receive recommendations of recipes they might like (this uses collaborative filtering technology). In addition, users could choose to click tags and get a list of recipes that contains that particular tag.

Additional website link may likely be created, for example a link to the most popular recipes. The website is also expected to be optimised for mobile phone viewing.

5.3 Version 3

The ideal version of our product involves the implementation of a variety of possible functionalities. For example, a mobile application could be developed. Recipes could be made searchable by not just ingredients but also, for instance, the type of cuisine (Chinese dish) or whether recipes are vegetarian or non-vegetarian.

Social networking is another possibility, with users being able to interact with other users and leave comments of the pages of other users. Users might be able to upload their own recipes and receive ratings from other users. The possibility for improvements are abundant.

6 Implementation Decisions

6.1 Decision Influences

6.1.1 Aims

The aims of version 1 have a strong influence over implementation decisions. They are as follows;

1. To be a complete working release of a usable piece of software
2. To allow the team to familiarise themselves with the tools and systems to be used for later versions
3. To explore the capabilities of those systems, to inform and inspire later decisions.

6.1.2 Design Principles

The project is being developed using a version of the Extreme Programming (XP) Methodology. XP's software development principles have an impact on the software design principles of projects developed using it.

Similarly the Web Framework Django has its own set of design philosophies[7] which also influence the project's design principles.

The principles of XP and Django are quite similar and complement one another quite well, so it is possible to abide by both sets of principles without contradictions.

Some of the XP/Django principles that have the most influence on implementation decisions are listed below;

DRY Don't Repeat Yourself
: Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

YAGNI You Aren't Gonna Need It
: Always implement things when you *actually* need them, never when you just *foresee* that you need them

Maximise Code Reuse If two bits of code look similar, move them out into a more general function. If two functions do a similar thing, merge them. This keeps redundancy low.

Both XP and Django have a strong basis in philosophy and principles, and while they both leave the developer the freedom to chose their implementation decisions, they are designed to work best with implementations that follow their principles.

[7] `http://docs.djangoproject.com/en/dev/misc/design-philosophies/`

6.2 Decisions

6.2.1 The `recipes` App

Recipes are the only thing that version 1 does, so it would make some sense to simply have the project as a whole perform the recipe functions, and not use any apps. However, the functionality of the site was put in a `recipes` app for two reasons. Firstly, it is best practice in Django to have all code in conceptually distinct, reusable apps, to maximise potential code reuse. Secondly, as one of the aims of this version is to introduce the team to working with Django, and apps are a major part of Django, it made sense to use apps even if they are not strictly necessary.

6.2.2 URL Design

The URL design is intended to be very simple and readable. In accordance with Django URL design principles, there are no filename extensions in URLs.

6.2.3 Model Design

The only implementation decision of note in the model design is the use of a python `property` to handle recipe tags. A `property` is a python language construct that behaves as though it is a class variable, but behind the scenes calls a getter or setter function when it is fetched or assigned to. This was used because, although it makes the model less readable, it makes all of the code that deals with the model far more readable, and it is this code which is more complex and benefits more from simplification.

6.2.4 View Design

The `recipe_list` View

There are 2 views that simply show a list of recipes:- `recipes_all` (the view of all recipes on the system) and the results section of `search` (the view of all recipes that meet the search terms). In order to maximise code reuse, the functionality of displaying a list of recipes was taken out into a separate `recipe_list` view, which is called by both `recipes_all` and `search`.

The `search` View

The search is deliberately the simplest search possible that meets the specifications. The set of results is simply the set of recipes which contain any of the ingredients searched for. This will be radically improved in later releases.

6.2.5 Template Design

> "The most powerful – and thus the most complex – part of Django's template engine is template inheritance. Template inheritance allows you to build a base "skeleton" template that contains all the common elements of your site and defines blocks that child templates can override."

– Django's Template Documentation[8]

Template Inheritance provides a good opportunity to maximise code reuse, but it was not used in version 1 in an attempt to keep template design simple.

[8] `http://docs.djangoproject.com/en/dev/topics/templates/`

Figure 4: The BBC Food Recipe Search Page

7 Research

7.1 BBC Recipes

BBC Recipes (Fig 4) is a web application that allows the user to input up to three ingredients and returns a list of related recipes.

BBC Recipes has two search options, basic and advanced. The basic search allows the user to input up to three ingredients with the option to find the recipe by television program or by chef. This search also provides the tagging options of quick recipes and vegetarian. However the advanced recipe search includes a wider choice of search preferences such as, the preparation method, cuisine, season and dietary requirements. After looking over the source code its very obvious that this web-based application uses an online database using a query language to return recipe results.

One main advantage about this application is the advanced search option which includes a numerous amount of tag options. This option becomes quite useful when searching through a large database as this kind of search minimizes the results. A good example of this is a search including flour, butter and sugar with the season tag being Christmas and the dietary needs tag being nut-free returning only 14 recipes. However a search including just the three ingredients returns 400 recipes.

One flaw that I have noticed with the search is that the search field is a text box which involves text input, this not validating the input until creating the search. The

Figure 5: The RecipeZaar.com Home Page

validation matches the input with similar words, for example for flor it will return flour. However when having typos like aooples, instead of apples the returned match is allows. To solve this issue a drop-down menu including the ingredients from the database would perhaps be a better idea.

The design of the website is attractive with a good use of colour and images. However the navigation of the website is slightly tedious, the reason for this being when expanding the actual recipes on the home page the webpages content increases and therefore leaving the user to have to scroll through the webpage to view its content.

7.2 RecipeZaar.com

Recipezaar (Fig 5) is a website that includes a large database of recipes and user accounts. Recipes are searchable by recipes, cookbooks, ingredients and members.

The recipe search allows the user to input n number of ingredients. The input method is a text box which includes no validation. If the user inputs flor instead of flour the search returns nothing. The query used for this search seems to be very basic, the reason for this being if the user inputs flour, sugar, butter, eggs for example one of the returned recipes is vegetable casserole which doesnt contain any of the above ingredients. Another example of a recipe is leach family turkey stuffing which only contains one of the above ingredients. Because the search isnt specific and because the website has a large online database the results are too vague.

This website introduces the use of collaborative filtering by including user accounts with the option to upload recipes and rate other recipes. When viewing a members recipe theres also the option to send a private message, submit corrections, send the recipe to

the users email address or mobile device and create a shopping list. The recipe page also directs the user to other recipes like the chosen recipe. The website also has a community webpage with forums for general discussions.

The design of the website is interactive with a good use of JavaScript. The colour scheme is neutral with a good use of images.

8 Expanded Problem Description

As a group we understood that the aim of the project is to develop a software kitchen assistant tool. The tool must be able to provide recipes which match a supplied list of available ingredients. The software should provide a number of matching recipes and rank the suggestions according to how well they match. Suggesting we create a web-based tool, the online database will also take into account users own food preferences, this allows us to introduce collaborative filtering to manage the recommendations.

We have chosen to use extreme programming therefore making frequent and small releases. We have specified three versions, these being minimum, realistic and ideal.

For all three versions the interface of the kitchen assistant tool will be web-based and will allow the user to select at the least three ingredients. Once these ingredients are submitted it will return a list of recipes that are ranked in accordance to how well the recipe matches the selected ingredients.

Version 1 will be a web-based application with a simple interface using only HTML, CSS and Django template markup. The online database will contain several recipes that are searchable by ingredients.

Version 2 is an extended version of v1. The web-based interface will introduce JavaScript and possibly AJAX but will also have a separate simpler web interface for mobile devices (with just the use of HTML and Django template markup). The online database for this version will contain a vast amount of recipes that are searchable by a combination of ingredients. With the introduction of collaborative filtering we intend to create user accounts that allows the user to rate recipes and then return recommendations based on previous ratings.

Version 3 extends v2. The web-based interface will remain the same however we will create a mobile optimised interface and an Iphone application. The online database for this version will include a very large amount of recipes that are automatically updated and maintained. Recipes will be searchable by combinations of ingredients and by other tags such as vegetarian, Italian, low fat etc...Returned recipes will be returned based on past ratings and accumulated data from the entire database. Recipes will also give recommendations for several users i.e. A recipe that alice and bob will both like. This version will include a full user system with profiles and user-uploaded recipes. The application will support social media functions such as messenger, the possibility of rating, tagging and commenting on other recipes.

The project has be divided into 5 managerial areas. These being management, technical, design, quality assurance and documentation. Every member has been given the responsibility of one of these areas and is expected to ensure all targets are met.

9 Collaborative Filtering Technology

The final prototype of our project deals with users being able to rate recipes and receive recommendation of recipes they might like. There are a number of ways this system can be implemented with collaborative filtering technology. Collaborative filtering technology provides numerous prediction algorithms which we can use to implement such a system and acquire ratings.

9.1 About

Collaborative filtering (CF) is the process of evaluating items through the opinions of other people [9]. While the concept of collaboration is nothing new or novel- it is in human nature to take the opinions of other people- the technology itself was developed in the early 90s[r]. The main difference between collaboration in everyday life and through the internet is that the internet allows us to utilise the opinion of large communities.

One of the earliest systems utilising the collaborative approach was Tapestry, developed by Xerox Parc[10]. Tapestry stored textual information, along with metadata and annotations about this information upon, which users could apply queries.

Users are recommended items based on their ratings for particular items. There are two ways a rating may be acquired- either explicitly or implicitly. Ratings are gathered explicitly when the user submits an opinion on an item, as opposed to implicit ratings which are inferred from user actions[].

Collaborative filtering is especially relevant to our project since it addresses the user task of helping users find recipes that he/she might like. The functionality we would like our system to have is that of constrained recommendation, as opposed to dealing with prediction (where given a particular recipe, calculate its predicted rating). Constrained recommendation is suited to our project because it allows users to input particular constraints, in this case, the recipe ingredients, and from the list of recipes generated, recommend users the recipe [ ]. Provided the numerous number of recipes in the database, and given a users limited attention span, it would be logical to return a recommendation which the user is more likely to appreciate.

9.2 System Pre-requisites

However, this technology will work effectively provided the domain it is applied to meets certain criteria. In our case,

1. The database should hold many recipes If too few recipes then there is no point of recommendations.

2. There must be many ratings for a recipe For a useful recommendation, the recipe needs enough ratings. Assuming each user rates only few recipes, and the large

[9] r Schafer J.B., Frankowski D., Herlocker J., Sen S.: Collaborative Filtering Recommender Sytems

[10] Goldberg D., Nichols D.,Oki B.M.,Terry D.: Using Collaborate Filtering To Weave An Information Tapestry. Communiation of the ACM, 35(12): pp 61-70

database of recipes, this implies that there must be a sufficiently large amount of users.

3. Users must rate multiple recipes Else, we will be unable to relate recipes to each other.

9.3 Algorithms

Algorithms can either be probabilistic or non-probabilistic. Probabilistic algorithms use probability distributions to compute, for example ranked recommendation lists. However, non-probabilistic algorithms models are more popular with practitioners[].

9.4 Non-Probabilistic algorithms

9.4.1 User-based Nearest Neighbour Algorithms

Predictions are generated here for users based on ratings from similar users, called neighbours. By analysing all the users neighbours, a rating can be predicted for a particular item for the user. Neighbours who are more similar to the user will have a higher weight in calculating the rating. Such similarity is elucidated by the Pearson correlation. Additionally, a neighbours personality will be adjusted for (some may be optimistic and rate consistently high and pessimistic neighbours rate consistently low even if both neighbours mean the same thing).

However, there are many caveats with this algorithm. If there are few ratings, then the matching of a users and a neighbours ratings will be biased, such that a particular neighbour will start heavily influencing the users neighbourhood. Additionally, the formula for this algorithm does not account for popular items. If numerous people agree that a particular recipe is excellent, then this data is less important than one in which users agree on a recipe of debatable taste. Moreover, basic implementations of this algorithm have linear time and memory requirements[]. Although, there are techniques which help alleviate such requirements (e.g. clustering) they also have their own limitations.

9.4.2 Item-based Nearest Neighbour Algorithms

Conceptually similar to user-based nearest neighbour algorithms, this algorithm focuses on acquiring ratings based on similarities between items. Adjusted-cosine similarity is the most popular and accurate [] way to calculate similarity between items. It is very similar to the Pearson correlation that is used with user-based nearest neighbour algorithm. There exists reports which suggest that item-based nearest neighbour algorithms give more accurate predictions compared to user-based[11].

A big disadvantage is the complexity of the algorithm, whose size could be the square of the number of items, although, there exists ways to prune the size of the model [sarwar

[11]r52Sarwar B., Karypis G., Konstan J.A., Riedl J.:Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of the 10th international conference on World Wide Web. (2001) Hong Kong. ACM Press p.285-295

only k coratings]. However, pruning causes difficulty prediction. Additionally, items cannot have too few ratings as this may lead one item to heavily influence a prediction.

9.4.3 Dimensionality Reduction Algorithms

These algorithms help reduce the complexity of very large systems with a lot of items. They map the item space to underlying tastes of the user, which diminishes the system runtime complexity. Typically, a vector based technique like vector decomposition[] can be used to extract these tastes. This way, a prediction can be found reasonably.

However, techniques like vector decomposition require complex mathematical computation to produce the taste-space. Although heuristic methods help in updating this taste-space to avoid constant recalculation, software maintenance is hard due to this complexity. Reportedly this reduction in complexity is not a significant enough improvement.

9.5 Probabilistic Algorithms

Probabilistic algorithms employ well established concepts of probability to make predictions. Baynesian-network provide the preferred framework to generate reliance among users or items[] and can be implemented as decision trees to represent probability tables [9, diagram] for example. Expectation maximisation algorithm [hoffman,29] also uses Gaussian probability distributions to extract user underlying tastes.

One advantage of these algorithms is that not only can they help calculate the most probable rating but they also help compute the plausibility of the rating.

9.6 General concerns about all algorithms

If there are too few ratings then none of the algorithms will yield any useful data due to biasness. There are a number of techniques that help mitigate such problems. For example you can choose to discard items with ratings which are below a certain number although, this will reduce the scope of the collaborative filtering system.

9.7 Acquiring Ratings- Design Decision

There are two way to collect ratings- explicit and implicit. Explicit ratings are usually more accurate description of the users preferences however, this requires some input from the users end. It was previously believed that the users would rarely choose to spend their time rating an item, however, experience has refuted this belief as exemplified by websites such as Youtube[]. Moreover, incentives can be used to entice users into rating, for example, reward points.

In contrast, implicit ratings require no input from the user but they usually are not as accurate as explicit means. For example, the amount of time the user spends reading a recipe might be an implicit mean of ratings collection (the more time the user spends reading the recipe, the more he/she will likely like the recipe), but this may be

inaccurate as the user may not like the recipe after reading it. However this uncertainty will be assuaged if you are able to collect a large amount of such implicit data.

9.8 Rating Scales

Another issue is selecting rating scales. If you provide a rating scale with more options, it will provide more information about the user. That is not to say that you provide a rating scale with an unnecessarily large amount of options, which may not add value to the system. Importantly, you must consider the needs of the users. If you provide too few ratings on the scale, users may find that they cannot express their opinions accurately.

9.9 Cold Start Issues

These issues refer to when the system is unable to provide a useful recommendation to the user due to an initial dearth of ratings. This can occur as the following scenarios:

9.9.1 New User

No specific predictions can be made to the new user since he/she has not made any ratings yet. This can be overcome, for example, by asking the user to rate some initial items before they can register the service or by obtaining demographic information from the user and matching his/her ratings with other users of similar demography.

9.9.2 New Item

The new item will initially have no ratings so it will not be recommended. This can be overcome by recommending items through non-CF techniques like content analysis [12] or random selection of new items and asking for ratings on those items.

9.9.3 New Community

If there are no ratings then the system cannot provide recommendations to users who seek this service. User retention will diminish. A solution is to provide rating incentives to a small subset of the community[], before expanding the service to the entire community.

9.10 Challenges with Collaborative Filtering

The obvious challenge is that of privacy. For a system provide accurate recommendations, it is necessary as much information about the user as possible. However, if a user divulges too much information, he/she is at risk if the central database containing this information is compromised. Additionally, the user should trust the particular website to not misuse user information.

[12] r53 Sarwar B., Karypis G., Konstan J.A., Riedl J.:Incremental SVD-Based Alogorithms for Highly Scaleable Recommender Systems. Proceedings of the Fifth International Conference on Computer and Information Technology (2002)

Another issue is of trust. Users can purposely give artificial ratings which are not representative of their true opinions. Companies can manipulate recommender systems as well by overwhelming the system with favourable ratings of its own products [13].

[13][6 BBC News Online, "Sony Admits Using Fake Reviewer." June 4, 2001 `https://news.bbc.co.uk/1/hi/entertainment/film/1368666.stm`]