FeedMeRight-Recipe Recommendation System

A Thesis Presented to

The Faculty of the Computer Science Department

In (Partial) Fulfillment

of the Requirements for the Degree

Masters of Science in Computer Science

by

Student Name:
Pallavi Amod Chavan

Advisor:
Dr. Brian Thoms

June 8, 2020

APPROVED FOR MS IN COMPUTER SCIENCE

_____   6-29-20
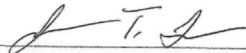Advisor: Dr. Brian Thoms   Date

_____   June 29, 2020
Dr. Michael Soltys   Date

_____   06-30-2020
Dr. Jason Isaacs   Date

APPROVED FOR THE UNIVERSITY

_____   07/01/2020
Dr. Osman Ozturgut   Date

Non-Exclusive Distribution License

In order for California State University Channel Islands (CSUCI) to reproduce, translate and distribute your submission worldwide through the CSUCI Institutional Repository, your agreement to the following terms is necessary. The author(s) retain any copyright currently on the item as well as the ability to submit the item to publishers or other repositories.

By signing and submitting this license, you (the author(s) or copyright owner) grants to CSUCI the nonexclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video.

You agree that CSUCI may, without changing the content, translate the submission to any medium or format for the purpose of preservation.

You also agree that CSUCI may keep more than one copy of this submission for purposes of security, backup and preservation.

You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. You also represent and warrant that the submission contains no libelous or other unlawful matter and makes no improper invasion of the privacy of any other person.

If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant CSUCI the rights required by this license, and that such third party owned material is clearly identified and acknowledged within the text or content of the submission. You take full responsibility to obtain permission to use any material that is not your own. This permission must be granted to you before you sign this form.

IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN CSUCI, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT.

The CSUCI Institutional Repository will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

FeedMeRight - Recipe Recommender System.
_____
Title of Item

Recommender System, Machine Learning, Content-Based, Matrix-Factorization.
_____
3 to 5 keywords or phrases to describe the item

Pallavi Amod Chavan
_____
Author(s) Name (Print)

Chavan   June 13, 2020
_____
Author(s) Signature   Date

Abstract

With so many rapid changes happening around us, people are moving towards a healthy lifestyle which includes choosing the right food. Today the overabundance of information can hinder the ability to choose the right information. Recommendation systems apply techniques that can filter information and narrow it down based on our preferences or needs and help us to choose what data is important for individuals. Recommender systems have been widely used in e-commerce sites, social networking and entertainment industries. In guiding individuals towards more healthy lifestyles recommender systems can be used in dieting, allowing individuals to find recipes that target specific foods not just based on a user's taste but also considering a user's lifestyle and calories required for that user. This thesis presents the design, implementation and evaluation of various recommendation approaches within the recipe domain. More specifically, combine different recommendation techniques and use machine learning to recommend healthy recipes based on a user's profile.

Keywords – Recommender system, Machine Learning, content-based, matrix factorization.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The internet is a vast network of machines that connects a large number of computers together worldwide and allowing them to communicate with one another. The World Wide Web is an information-sharing model that is built on top of the internet in which information can be accessed or manipulated easily, which has lead to the dramatic growth in increased usage of the internet. One of the results has been the dramatic increase in data and a phenomenon known as BigData.

BigData refers to exponentially increasing data at a high volume, the high velocity with variety. This huge amount of data has intrinsic value but provides no utility until it's discovered [1]. With such information overload, it is problematic to find exactly what a user is looking for.

One way to find value in BigData is to deeply analyze it and its interrelated features such as new products, corresponding reviews, ratings, and user preferences. Formulating information from raw data is an entire discovery process that requires an insightful analysis that would recognize patterns to predict user behaviors to recommend products.

Handling BigData through manual processes is very inefficient. A more efficient way of processing such a huge amount of data is automating the process of classifying, filtering data of users' opinions, features, and preferences in order to understand and predict a new set of related products. Recommender systems are tools that filter information and narrow it down based on a user's preferences and helps a user to choose which he/she may like. They consider the opinions of communities of users to help each individual to understand the content of interest from overwhelming information [2]. Recommender systems can be defined as a tool designed to interact with large and complex information spaces to provide information or items that are relevant to the user [3]. Today, recommender systems are widely used in a variety of applications. Initially, it applied for commercial use to analyze data. Amazon is a good example of such an e-commerce website. However, it is now present in several different domains including entertainment, news, books, social tags, and some more sophisticated products where personalization is critical such as recipes domain.

According to the World Health Organization (WHO) [4], globally 39% of the adults were over-weighted and 13% were obese in 2016. Overweight and obe-

sity can cause diabetes, blood pressure, heart disease and many other chronic diseases [5]. A well-balanced diet plays a very important role to improve overall health. With rapid changes in our busy lifestyle, people find it difficult to choose healthy eating options [6]. People tend to move towards options that satisfy their taste or easy by ignoring the health factor which includes required calories and nutrition. Exploring better dishes in such huge information is very tedious. To solve this problem computer scientists can build systems that can help to narrow down the information considering our health and eating history. This thesis further discusses the different approaches for the recipe domain to recommend healthy recipes.

Chapter Background provides an overview of the process of building recommendation algorithms, basic recommendation techniques, and evaluation metrics to measure the performance of recommendation engines. Further, set up to run experiments, features extraction process, implementation approaches and experiments are described in chapter Implementation. Finally results and conclusion are discussed in chapter Results and Conclusion and Future Work respectively.

1.2 Contributions

- **Additional Attributes:** In papers [6, 7] content-based technique is used based on only ingredients attribute. This thesis considers the content-based technique with the usage of more attributes such as cook methods and diet labels in addition to ingredients.

- **Consideration of Health Factor:** The work done in [6, 7, 8] considers only user preferences either in the form of ratings or tags but this thesis considers user's preferences along with user's health factor in terms of understanding how much calories a user should consume.

- **Hybrid Approach along with Calorie Restrictions:** The work in [6, 7] uses only a content-based approach on the other hand the work in the paper [8] uses only collaborative filtering approach. Contrary, this thesis considers a hybrid approach that uses both content-based and collaborative approaches combined together with calorie restrictions so that system can consider user's preferences along with a user's health factor.

Chapter 2

Background

2.1 Related Work

In recent years web application development has advanced and grown. With this development, food or recipe sharing applications have emerged. Hence the scope of recommender systems in the food domain has increased as it is easy to get user feedback in the form of ratings, reviews, or comments in the web applications.

There are many reasons why recommending food or recipes is challenging. Such challenges consider changing user's minds towards their own healthy behavior. Another challenge is in predicting what people would like to eat because it depends on many factors including region, culture, and demographic, to name just a few. Prior work in recommender systems includes approaches ranging from content-based systems to collaborative filtering sys-

tems to hybrid approaches.

According to Mika [9], there are two types of food recommender systems present. The first type considers the user's preferences. It recommends recipes that are similar to the recipes liked by a user in the past. The second type considers only those food items which have been identified by health care providers. It recommends recipes based on nutritional requirements for that user with no consideration of a user's preferences.

Research of Freyne and Berkovsky [6] uses a content-based technique to predict a rating of an unrated recipe based on the corresponding ingredients included in the recipe. For a user, the framework breaks down unrated recipe into ingredients then calculates a rating of each ingredient based on the rating of all other rated recipes which contain the same ingredients. After calculating ratings for all ingredients of an unseen recipe, the framework predicts the rating of a user for the unseen recipe based on an average of all rating values of all ingredients in this targeted unseen recipe. Recipes with high prediction values will be recommended.

Continuation of Freyne and Berkovsky's [6] work resulted in considering the weighting factor for ingredients [7]. This framework takes a different approach by using two matrices for user features. One of them is weighted positively for rating values 4 and 5. Weighting factors for rating values 5 and 4 were 2 and 1 respectively. Contrary, another matrix weighted negatively for rating values 1 and 2. Weighting factors for rating values 2 and 1 were

1 and 2 respectively. The matrix with positive values implies strong likes between ingredients. The matrix with negative values implies strong dislikes between ingredients. Based on ingredients similarity scores, recipes were recommended.

Recent work experiment done by Chun-Yuen Teng, Yu-Ru Lin, and Lada A. Adamic [10] shows that how can one bring a 'healthy' factor in recommendations by substituting ingredients.

Other than content-based, Collaborative filtering based algorithms have also been proposed. Freyne and Berkovsky experimented with using the nearest neighbor approach (CF - KNN) on ratings. In [7] a recipe recommender implemented using Singular Value Decomposition (SVD) outperformed a content-based and collaborative approach experimented by Freyne and Berkovsky. Elahi proposed a food recommender system using active learning algorithm and matrix factorization [8] which collects ratings and tags in the form of feedback. This system provides human-computer interaction for collecting user preferences in the form of ratings and tags. Every user and every recipe are modeled by vectors that represent their latent features. Continuation of this work resulted in incorporating calorie count [11]. These approaches performed well in predicting what users may like according to user's taste, but it did not help in changing a user's behavior towards a healthy lifestyle. By incorporating calorie count we can restrict a user to choose the right food within range of their personal tastes. In this thesis, I present a comparative analysis of recommender approaches in the food domain and varying recom-

mender approaches to recommend healthy recipes by incorporating calorie count and look to extend the work by [8, 11].

2.2 Data Collection and Filtering

2.2.1 BigData

Recently, the ubiquitous availability of internet connections along with mobile technologies is generating a huge amount of data. This data is considered as BigData and it can be described as a large volume of complex data [1]. Volume refers to the amount of data or quantities of data generated in a unit of time. Velocity refers to the speed at which data is generated and the speed at which data moves around. For example, social media messages going viral in seconds. Variety refers to the type of data that is generated. For example, as shown in Figure 2.1, data generated from mainframe source was less in volume and structured than today's data generated by different channels such as mobile messaging, social media, and internet.
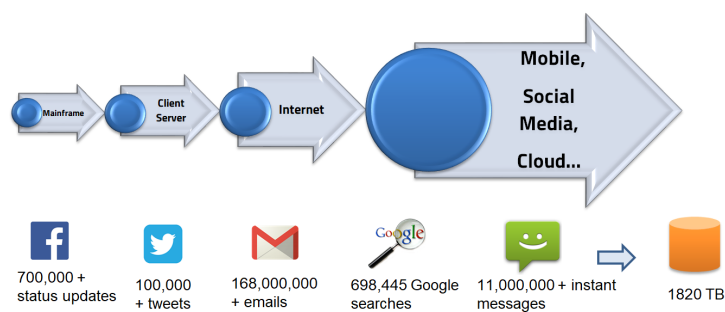

Figure 2.1: Growing Data Velocity per second

Recommender systems can perform well based on a large amount of data.

Big Data is a driving force behind recommendation engines. To make this data useful, systems need to collect data, filter data in specific patterns from which we will get useful information.

2.2.2 Data Collection

The data collection phase is a foundation of the accuracy of the recommendation engine. It is helpful to generate user profiles or models for making recommendations. To well construct a user profile, a recommendation engine relies on different types of inputs such as explicit feedback which is explicitly specified by a user to notify a user's interest in an item or providing implicit feedback by understanding user preferences from user's interaction with the system [12].

2.2.2.1 Explicit Data

In explicit data collection, an application prompts the user through a user interface to provide feedback about the product in order to understand the user's preferences to improve the model. The user then interacts with the system by responding to these prompts. Explicit data is any information provided by a user explicitly or knowingly in the form of ratings, reviews, or comments. For example, below screen-shot shows that Netflix is collecting the data explicitly in the form of ratings given by the user to different movies.
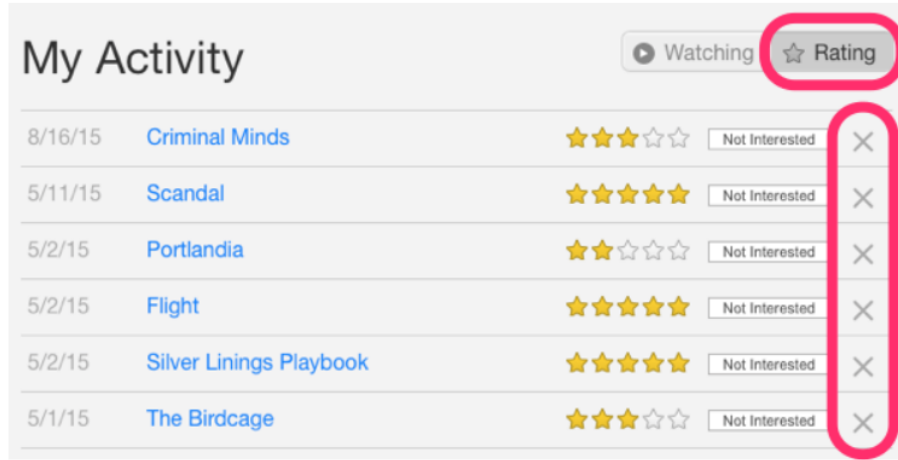
Figure 2.2: Explicit Data [13]

Although explicit feedback requires more effort from the user, it is still reliable data as it does not require any extraction of preferences and it maintains transparency into the recommendation process [14].

2.2.2.2 Implicit Data

Implicit data implies to when a system attempts to understand more about user's preferences by monitoring interactions of a user such as browsing history, purchase history, button-clicks, links followed by a user. Implicit data is information provided by user discretely in the form of different interactions. For example, screenshot in Figure 2.3 shows that Amazon is collecting the data implicitly in the form of storing a user's order history.
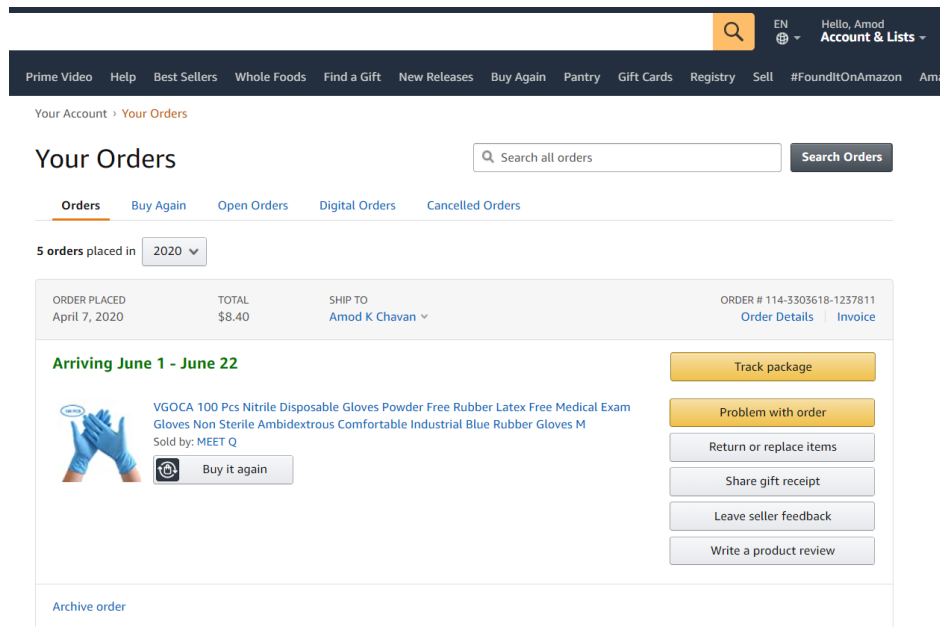
Figure 2.3: Implicit Data [15]

Based on a user's prior purchase, Amazon is recommending other products as shown in screen-shot Figure 2.4.
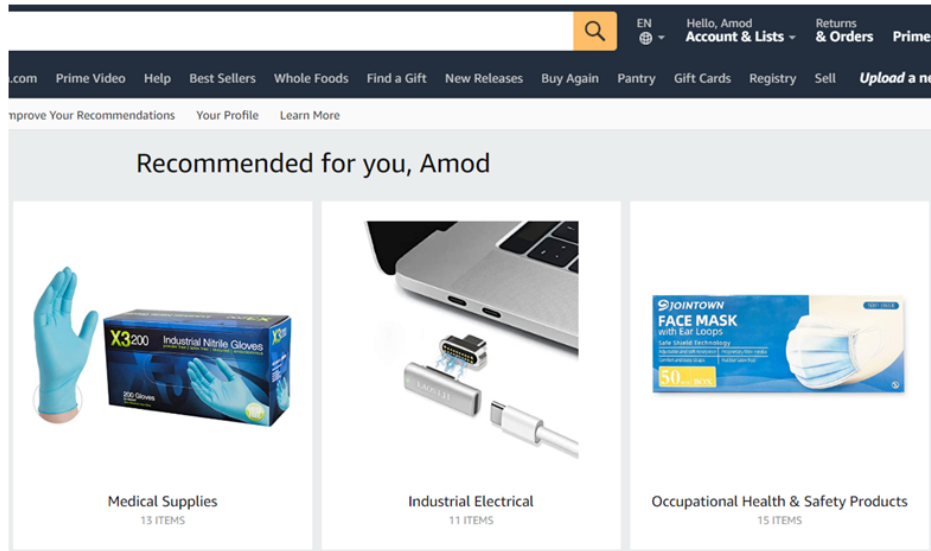
Figure 2.4: Recommendations based on Implicit Data [15]

Although implicit feedback does not require more effort from the user, it is less accurate. For example, a user of an e-commerce system purchase items as gifts for others. Those purchases and related activities do not necessarily communicate anything about the user's tastes. A whole family may share a single account for purchases from e-commerce websites. In that case, input for the system will be an aggregation of all people's preferences.

Explicit and implicit data collection methods are used for building an online recommendation system. The work in this thesis is based on offline data. Different techniques are involved in offline data collection.
The information-gathering process involves many techniques such as web crawling, document processing, indexing and query processing. A crawler

processes all URLs via techniques like breadth-first search and depth-first search and stores the web server's response for each URL. The documents retrieved are then processed in order for its meta-data and to remove any noisy data. Data indexing is then applied so that the retrieval and processing of extracted data are quicker.

2.2.3   Information Filtering

Information Filtering is associated with a data search. When a user requests information, it is treated as a query in the form of keywords and is applied on the indexed data. The objective of the query processor is to return the most relevant documents to the user.

Filtering is a key component to retrieving adequate information per user preferences or user profile. User behavior is studied from past user-profiles activities which help to filter out any irrelevant data and provide relevant suggestions pertaining to current needs. Recommendation process has three different phases as highlighted in Figure 2.5. Data collection using different techniques such as explicit data and implicit data is known as the information collection phase. The different learning algorithms apply to the collected data to filter information. Applying algorithms to learn more about data is known as the learning phase. Retrieving most relevant products or items by predicting what the user may prefer from filtered data is known as a recommendation or prediction phase.
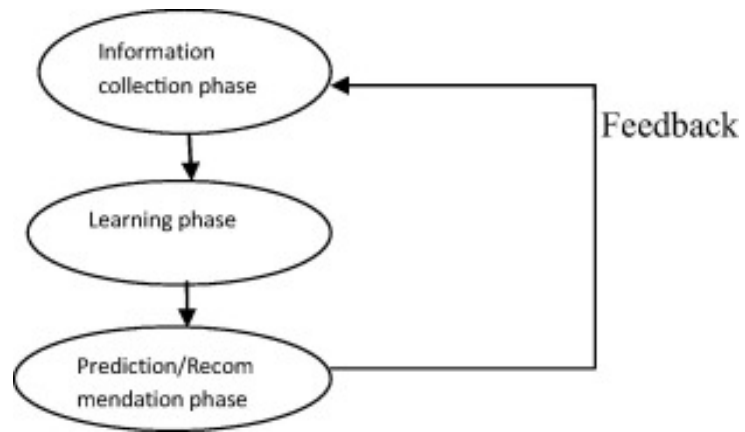
Figure 2.5: Recommendation Phases [16]

2.3   Recommender System

Recommender systems are information filtering systems that provide a solution for the problem of information overload [17]. The process involves filtering important information out of a large amount of data according to the user's preferences and interests. The recommender systems can predict item or product relevancy to the user based on the user's profile and preferences. The basic idea of the general recommender model is given in Figure 2.6 which explains the interaction of users and items with the system. Dataset of items represents the description of items for instance recipes. Description of a recipe may contain information about different factors of recipes such as ingredients used in the recipe, cooking type, calories, cuisine type. Consider a user who likes Mexican recipes with low calories. The profile exploitation step matches the user's profile information with all recipes present in the data set. At this point, the recommendation algorithm gets applied to match the user profile and recipe profile. All Mexican dishes with low calories will get filtered out from the data set for our user. All filtered recipes hold some ranking based on the recommendation algorithm. Top- N recipes will get recommended to our user based on the ranking predicted by the recommendation algorithm. The user adopts those recommendations and responds to the results generated by the recommendation system by providing feedback. The system updates the user profile based on the feedback received by the system from a user. Recommendation systems correlate one user with a

group of other users to find similar users. In this case, the system will match a user's profile with all other users who prefer Mexican low-calorie food.
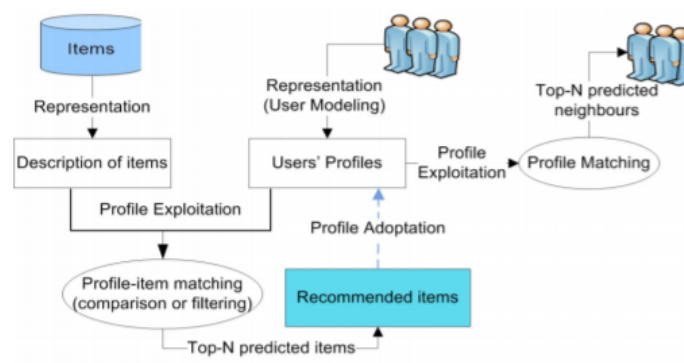
Figure 2.6: General Recommender Model [18]

Based on recommendations generated by the system, interaction between users and items may vary. For that reason, we need to understand the features of different recommendation techniques. Figure 2.7 shows broadly categorized recommendation techniques.
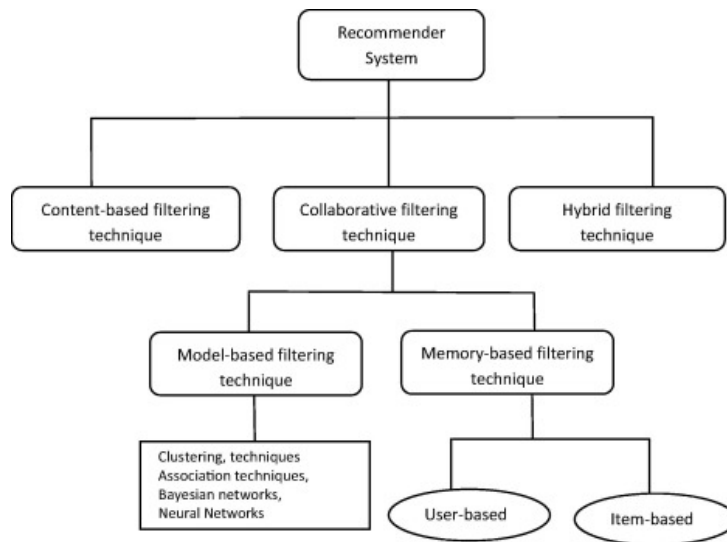
Figure 2.7: Recommendation Techniques [16]

As shown in Figure 2.7, traditionally there are two basic models of recommender systems.

- Content Based Filtering

- Collaborative Filtering

These commonly used filtering approaches are discussed in the next section.

2.3.1 Content Based Filtering

In the Content-based method algorithm, user preference is considered based on the item description. The rating and buying behavior of users are combined with content information available in the items. The main aim of content-based filtering is to create a profile for each item as well as each user in order to find similar items that reflect a user's taste [19].
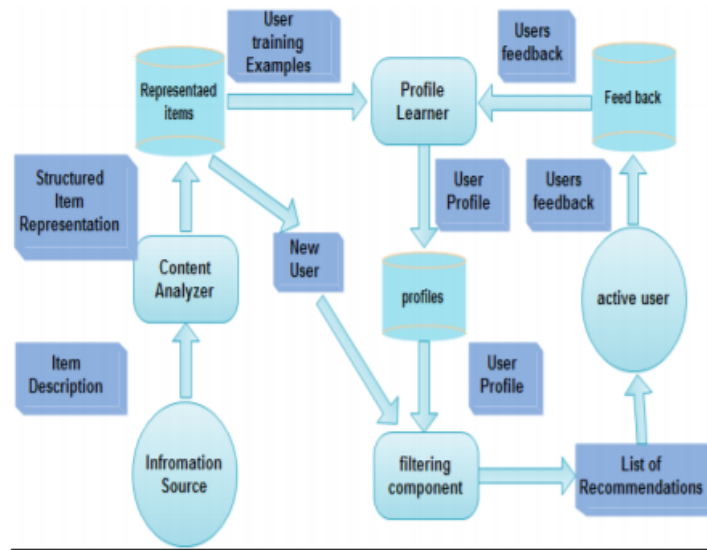
Figure 2.8: Content Based filtering Architecture [20]

The architecture of content based model highlights the process in Figure 2.8.

Content Analyzer

When information has no exact structure, the pre-processing step is necessary to extract relevant features from that document. Content analyzer analyzes

items such as recipes, documents, books, product descriptions to extract and present the content of items. Information source contains data in the raw format. Each data item coming from an information source is pre-processed and analyzed by feature extraction techniques to transform original information into a more structured format. The output of the content analyzer is the input for the profile learner and filtering component.

Profile Learner

Profile learner constructs a user profile by collecting user preference data. The generalization strategy is applied to the collected data using machine learning techniques. For example, the profile learner of a movie recommender can implement a relevance feedback method in which rating on a scale of 0 to 5 is considered. Rating value above 3 is considered as a positive rating implies likes for the movie and rating value below 3 considered as negative rating implies dislike for a movie. The learning technique combines liked and disliked movies by the user and the vector of positive and negative examples into a vector that represents user profile.

Filtering Component

The filtering component uses the user's profile to find matching items from the items' data set. The items are matching with a user's data profile are decided by calculating the similarity between a user's data profile and item vectors. A list of potentially interesting items is recommended by the filtering component.

In a content-based algorithm, each user's information can be stored in vec-

tor form which contains past behavior of the user. This vector is known as user vector or user profile. All the information about an item is stored in an item vector or item profile which contains all the details about item-specific attributes. Based on the similarity score between the user profile and item profile most relevant items are recommended to the user. The calculation of user or item vector is discussed in section Vector Space Model.

2.3.1.1 Vector Space Model

Vector Space Model (VSM) for information retrieval represents documents as queries as vectors of weight [21]. It is also known as the term vector model as it uses term occurrences as a vector identifier.

Each item profile and user profile can be represented in the form of vectors. For instance, consider an example of users and books rating relationships based on ratings given by users to books based on the book's genre. The Table 2.1 represents the book and its genre relationship in binary form. Here, 1 is considered as book shares that genre and 0 represent that book does not share the genre. Book1 is solely based on Machine Learning while book3 talks about Security and Databases. Here, the genre of book is an attribute or feature that is considered to represent a book in vector form. Rows in Table 2.1 represents item vector for book1, book2 and book3 respectively. Similarly, a user profile can be represented in a vector form. Consider a user liked book1 and book3 where 1 represents like and 0 represents dislike as shown in Table 2.2.

	Machine Learning	Databases	Cyber Security
book1	1	0	0
book2	1	0	1
book3	0	1	1

Table 2.1: Books in Vector Form

	user ratings
book1	1
book2	0
book3	1

Table 2.2: User Rating for Books

The simplest way to form a user vector is to take a sum of dot product of matrices as resulted in Table 2.3.

The content of an item can be complex such as text containing many words with high frequency. In such cases, a binary representation of rating considers all terms with the same weight. To consider only important terms, various components are combined to calculate the weight of the term. One way to calculate the weight of a term is to use term frequencies and inverse document frequencies (TF-IDF).

2.3.1.2 Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF is the most common computation used in text analysis and information retrieval process. It is used to measure the importance of a term with

Machine Learning	Databases	Cyber Security
1	1	1

Table 2.3: User Vector for Book's Genres

respect to document corpus. TF-IDF weighting negates the effect of high-frequency words in order to understand the importance of the term [22]. It consists of two parts, the first is 'term frequency' and the second is 'inverse document frequency'. The term frequency of term i in a document j is given in Equation 2.1

$$TF_{ij} = \frac{N_{i,j}}{\sum_k N_{kj}} = \frac{\text{Number of appearances of a term in a document}}{\text{Total number of terms in a document}} \quad (2.1)$$

Where $N_{i,j}$ is a number of occurrences of term T_i in the document d_j. The summation in the denominator provides the number of occurrences of all terms in the document d_j.

Inverse document frequency computes the importance of a word by comparing its occurrence in other documents. Inverse document frequency is given in Equation 2.2

$$IDF_{ij} = \log \frac{|D|}{|\{j : T_i \in d_j\}|} = \frac{\text{Total number of documents in the corpus}}{\text{Number of documents appear with a term}}$$
$$(2.2)$$

Where, $|D|$ represents the corpus of documents. Denominator gives number of documents containing the term T_i.

$$TF\text{-}IDF = TF * IDF \quad (2.3)$$

The value of TF-IDF is calculated by multiplying value of term frequency and value of inverse document frequency. Consider an example of 100 documents

corpus and a document X contains 100 terms in it. The term 'excellent' appears in a document X 10 times. then term frequency of $X = \frac{10}{100} = 0.1$. If the term 'excellent' appears in 20 different documents then $IDF = log\frac{100}{20} = 1.60$. TF-IDF of the term 'excellent' $= 0.1 * 1.60 = 0.160$

2.3.1.3   Advantages of content-based filtering

Content-based recommender systems are heavily reliant on the contents of the items that have been rated by the user. So, while making recommendations, this approach considers a user's taste and accordingly recommends an item that matches the user's preferences. Generally, the most popular items dominate less popular items. But this approach will not miss less popular items if it matches the user's unique taste [19].

Consider pepperoni pizza is the most popular dish and veggie pizza is a less popular dish in a restaurant. The restaurant will always recommend pepperoni pizza to every person because the dish is popular with no understanding if a customer is vegetarian or not. The content-based system analyzes that only veg items are preferred by vegetarian customers. In that case, veggie pizza will be recommended by the content-based system even though it is less popular.

2.3.1.4 Disadvantages of content-based filtering

User profiles are generated based on rated items. But for any new user who has not rated any items yet, the user profile will be empty. In that case, recommending a perfect item that matches to user's taste is difficult as the system does not have a user's taste information. This problem is known as the cold start. Also, to understand each item's feature, the system needs to examine the content of every item. Therefore if a number of items rise quickly, the performance in terms of speed of the system decreases [19].

2.3.2 Collaborative Filtering (CF)

The collaborative filtering system collects and analyzes a user's behavior based on a user's preferences given in the form of feedback, ratings, and activities. It is a domain-independent prediction technique. This technique can be used in the field where content can not be easily described by meta-data. To predict and recommend items for an active user, the collaborative filtering technique uses other than the active user's behavior in the system. It works on a user-item rating matrix of preferences for items by users. From this matrix, it matches the users with similar preferences and interests by calculating similarities between user profiles. Similarities between profiles can be calculated in different ways as discussed in section Similarity Methods. The fundamental idea of collaborative filtering is, it selects other users' opinions

and aggregate in such a way that it provides a prediction for an active user based on his preferences [23].

The main source of input for the Collaborative Filtering algorithm is in the form of a matrix of collected user-item ratings as highlighted in the Figure 2.9. Here rows represent user ratings for items on a scale of 1 to 5 where 1 is the lowest and 5 is the highest rating. Columns represent items that have received ratings from users. For example, user U1 has given rating 5 to the item i1 and rating 4 to item i3. The system finds similar user profiles in different ways but here we can see that user U4 likes the same items which are rated high by user U1. From this analysis, we can consider that user U1 and user U4 are quite similar. Based on this matrix, the system provides recommendations as an output. The first step of output is to predict ratings for items that the user may prefer. From our analysis, the system can predict that user U4 may like item i4 because item i4 is rated high by user U1. Prediction is a numerical value that represents the predicted score of a specific item for a specific user. The second step is to recommend a list of top-rated items as top-N items. In this example, the system recommends item 4 to the user 4.
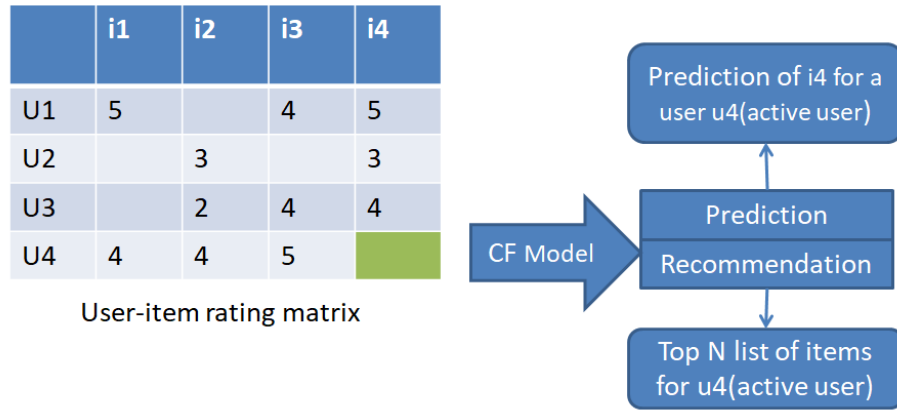
Figure 2.9: Collaborative Filtering Process [16]

Collaborative Filtering technique is broadly divided into 2 categories [24].

- Memory-based CF

  A memory-based collaborative filtering approach predicts item ratings based on ratings given by different users for an item.

- Model-based CF

  In contrary to memory-based collaborative filtering, the model-based algorithm takes the data that has been already preprocessed where it is cleansed, filtered, and transformed and generates a learned model to make predictions. This algorithm calculates the similarity between users or items by generating a model and analyzing their pattern to predict ratings on unseen items [25, 26, 27].

2.3.2.1 User-User CF

In user-user collaborative filtering similarity between users is calculated based on how similarly they rate several items. For any active user, it finds users other than active users whose ratings are similar to the active user and use their ratings on items other than items rated by the active user to predict what active user may prefer. Thus, it recommends items to the users that are most preferred by similar users.

Consider an example of users and ratings given by users for different recipes. This algorithm will find a similarity between each user based on the ratings they have given to the recipes in the past. The prediction of a recipe for a user u is calculated by computing weighted sum of the user ratings given by other users to the recipe i. The prediction for recipe i is given in Equation 2.4

$$P_{u,i} = \frac{\sum_v (R_{v,i} * S_{u,v})}{\sum_v S_{u,v}} \tag{2.4}$$

Where,

$P_{u,i}$ = prediction of recipe i

$R_{v,i}$ = rating given by user v to recipe i

$S_{u,v}$ =similarity between users.

To predict the ratings for a user other than active user we need to calculate similarity score. The similarity between users can be calculated with

the help of several methods described in section Similarity Methods. Prior to that, it is important to find items rated by both users and their ratings. Based on those ratings, if we choose to calculate similarities by using Pearson correlation, then we will get a correlation score between users. A higher correlation implies a higher similarity. Recommendations are made based on these predicted values.

This algorithm is quite expensive in terms of time as it involves calculating a similarity score between each user and from that score calculating predictions.

2.3.2.2 Item-Item CF

Item-Item CF filtering can be introduced to solve challenges in User-User CF. As seen in user-user CF may become so expensive if we have a large number of users. If we have a huge number of users than items then it is ideal to adopt item-based CF.

This algorithm calculates the similarity between items instead of users. It considers ratings of the active user to make predictions for item i, as i will be similar to the items rated in the past by the active user. Therefore, a user may prefer to use his own ratings than using some other users' ratings. It helps in maintaining user preferences and choices. The similarity between items can be calculated using any method discussed in section Similarity Methods.

The rating prediction for item-item collaborative filtering is calculated in Equation 2.5

$$P_{u,i} = \frac{\sum_N (S_{i,N} * R_{u,N})}{\sum_N (|S_{i,N}|)} \tag{2.5}$$

Where,

$P_{u,i}$ = prediction of item i for user u

$R_{u,N}$ = rating given by user u on item N

$S_{i,N}$ = similarity between item i and N.

2.3.2.3   Singular Value Decomposition (SVD)

SVD is a technique of matrix factorization which is used to reduce the number of features in the data set. The matrix factorization is done on the matrix which is generated by the user's feedback in the form of ratings on different items. In SVD, the technique is used to detect a latent relationship between users and items. Then it will generate a low dimensional representation of original matrix space to calculate neighborhood in the reduced space [28]. The original ratings matrix decomposes by SVD in two matrices in such a way that the product of decomposed matrices is the original rating matrix. The SVD is calculated as shown in Equation 2.6

$$SVD(M) = U \cdot S \cdot V^T \tag{2.6}$$

Where,

$SVD(M)$ denotes matrix M with dimensions $m \times n$ which are total number of users and items respectively.

Dimensions of matrix U will be $m \times m$

Dimensions of matrix S will be $m \times r$

Dimensions of matrix V will be $r \times n$

U and V are called left and right singular vectors. To reduce features of dataset one can keep only k highest values of V and S and eliminate lower entries. So, $(r - k)$ columns from U and $(r - k)$ rows from V^T are discarded to generate U_k and V_k^T matrices. Now M_k can be constructed with multiplication of U_k and V_k together using S_k. Generated M_k will closest rank k matrix to M. Mathematically it can be represented as in Equation 2.7

$$M_k = U_k \cdot S_k \cdot V_k^T \tag{2.7}$$

Rating prediction for user u for item i is given in Equation 2.8

$$r_{ui} = r_u + U_k \sqrt{S_k^T(u)} \cdot \sqrt{S_k} \cdot V_k^T \tag{2.8}$$

With SVD we can predict ratings with good accuracy.

2.3.2.4 Advantages and Disadvantages of collaborative filtering techniques

Collaborative Filtering technique has major advantages over content-based technique in such domains where there is a selection of many features is not necessary. Collaborative filtering has the ability to recommend a new item such that even if the content of that item is not stored as a user's preference in the user's profile. It is a very successful algorithm but it suffers from some drawbacks [29].

Data Sparsity Problem

Data sparsity problem is a result of a lack of availability of information about ratings of items in the dataset. A dataset with a huge number of items but only a few of them rated by users leads to a sparse user-item matrix. This problem causes the generation of weak recommendations as locating successful neighbors is difficult.

Cold Start Problem

Cold-start is a situation where the system does not have enough information about a new user or a new item in order to make relevant predictions. The user profile or item profile will be empty as a user has not rated any item so, the preference of a user will be unknown to the system. In Collaborative filtering, the similarity between users is considered based on a user's ratings.

Scalability

The nearest neighbor algorithm requires high computations. It grows with a

number of users and number of items in the system. Any web-based system which has a huge number of items and users may suffer from high scalability.

2.3.3 Hybrid Filtering

Content-based filtering technique do not involve the opinions of all users while recommending items. It has limitations to recommend only those items that are in the range of user's taste as discussed in the Content Based Model. However, collaborative filtering cannot give prediction to items that have never been rated as discussed in the Collaborative Filtering Model. Hybrid filtering techniques uses a combination of different recommendation techniques to overcome limitations of pure recommendation techniques and to improve performance [30, 31]. The performance of recommendation techniques can be measured by different evaluation metrics as discussed in section Evaluation Metrics for Recommendation Systems. Many researchers have combined content-based and collaborative filtering techniques to gain better results. The idea behind combining different recommendation techniques is that the resultant algorithm will provide more accurate and effective recommendations than any single algorithm [32]. Burk [33] has categorized hybrid techniques in different types. One of them is weighted hybridization.

2.3.3.1 Weighted Hybridization

In this technique, the results of multiple algorithms are combined to generate predictions by integrating the scores of each algorithm used. For example, the P-Tango system combines content-based and collaborative filtering techniques [34]. At the start, both techniques were equally weighted but based on performance and user ratings the weight for different techniques has gradually adjusted. In this thesis, the weighted hybrid technique has been used. Similar to the P-Tango system, this thesis combines content-based and collaborative filtering using SVD techniques to overcome the limitations of traditional algorithms. This thesis performed experiments on different weighting factors starting with equal weights but looking at offline evaluation results, weight for content-based and collaborative techniques has gradually adjusted as discussed in section Hybrid Model(CB and CF).

2.4 Similarity Methods

In this section several methods are discussed to calculate similarity score between two vectors.

2.4.1 Cosine Similarity

In cosine similarity measure [35], the result is the cosine of the angle between two vectors. It discovers the direction between two vectors if it is same or not. Often, cosine similarity is used to calculate document similarity in text analysis. The item ratings or preferences are stored in one vector called as item vector. The preferences of user are stored in another vector based on user's ratings and likes-dislikes is known as profile vector. Consider $A = (5, 0, 3, 0, 2)$ and $B = (3, 0, 2, 0, 2)$ are profile vector and item vector respectively, then the similarity between them can be calculated as per Equation 2.9.

$$sim(A, B) = \frac{A \cdot B}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \qquad (2.9)$$
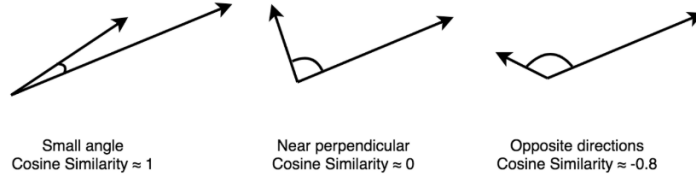
Figure 2.10: Cosine Similarity

Using Equation 2.9, if we compute cosine similarity between vectors A and B, then we get:

$$\| A \| \| B \| = (5 \times 3 + 0 \times 0 + 3 \times 2 + 0 \times 0 + 2 \times 2) = 25$$

$$\| A \| = \sqrt{5^2 + 0^2 + 3^2 + 0^2 + 2^2} = 6.164414$$

$$\| B \| = \sqrt{3^2 + 0^2 + 2^2 + 0^2 + 2^2} = 3.74165$$

$$sim(A, B) = 0.92$$

The value of cosine angle ranges between -1 to 1. Calculated result 0.92 shows that vector A and B are quite similar as the result inclines towards 1. As per Figure 2.10 lesser angle depicts less distance between two vectors. Hence, calculated vectors considered more similar to each other.

2.4.2 Euclidean Distance Similarity

The Euclidean distance between two points is the length between two connecting points. If we plot n-dimensional space and plot similar items, then they will fall under close proximity. Consider an example of a positive quad-

Movie/User	Lady in the water	Snakes on a plane	You Me and Dupree
A	2.5	3.5	2.5
B	3	3.5	3.5
C	2.5	3	
D		3.5	2.5
E	3	4	2
F	3	4	3.5
G		4.5	1

Table 2.4: User-Movie Rating Relationship [24]

rant of space and we plot items on the axis which are rated by the user. The Table 2.4 illustrates the relationship between users and movies with ratings. The points are drawn on the graph in Figure 2.11 represents ratings given by the user to those particular movies.
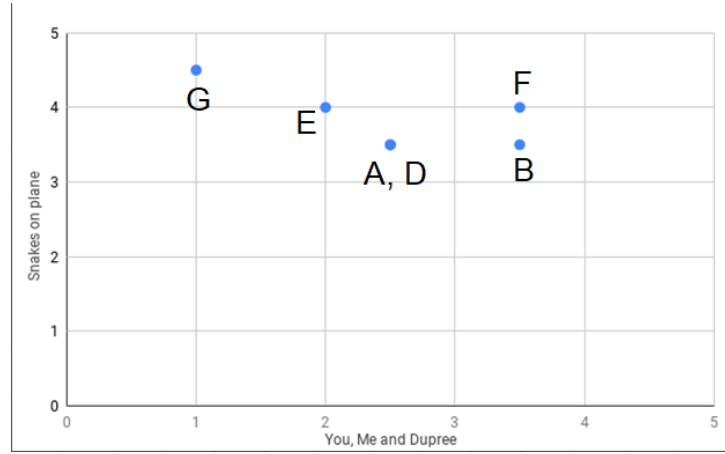

Figure 2.11: Movie Rating Plots [24]

In that case, we can calculate distance between items with Euclidean distance formula which is given by Equation 2.10. Euclidean distance is the square root of the sum of squared differences between corresponding elements of the

two vectors x and y. Similarity score based on Euclidean distance can be calculated using Equation 2.11. Similarity score based on euclidean's distance is defined as maximum distance divided by actual distance added to the 1. Higher distance represents lower similarity while lower distance represents higher similarity.

$$\text{Euclidean Distance } = d(x, y) = \sqrt{(x_1 - y_1)^2 + ... + (x_n - y_n)^2} \quad (2.10)$$

$$\text{Similarity score } (A, B) = \frac{1}{1 + Distance} \quad (2.11)$$

From above example, if we calculate Euclidean's distance between user A and user B then we get similarity score based on euclidean's distance :

$$Distance(A, B) = \sqrt{(3.5 - 3.5)^2 + (3.5 - 2.5)^2}$$
$$\text{Similarity score } (A, B) = \frac{1}{1 + Distance} = 0.5$$

2.4.3 Pearson's Correlation Similarity

Person's correlation helps in finding correlation between two users or items. Correlation values ranges from -1 to 1 [36]. Correlation on higher side implies more similarity. Equation 2.12 gives correlation between two users r_u and r_v.

$$sim(u, v) = \frac{\sum(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{(\sum(r_{ui} - \bar{r}_u))^2}\sqrt{(\sum(r_{vi} - \bar{r}_v)^2}} \quad (2.12)$$

38

Where r_{ui} and r_{vi} are rating scores from two users u and v for an item i. $\bar{r}_u$ and $\bar{r}_v$ denote the average rating by the two users. Pearson correlation score > 0 indicates positive association. On the other hand, Pearson correlation score < 0 indicates the negative correlation and score $= 0$ indicates no correlation. Hence a correlation value can capture a rating similarity between two users.

2.5 Evaluation Metrics for Recommendation Systems

In this section evaluation metrics that are used to evaluate recommender systems algorithms, are explored. Evaluation can be done in two ways, offline and online evaluation [37, 38]. In offline evaluation methods, analysis of collected data can be divided into training sets and testing sets. The dataset used in this thesis is divided into training sets and testing sets in the proportion of 80% training to 20% testing. The model of the recommender system is trained on the training dataset and the testing dataset is hidden from the engine. To understand the quality of a recommendation engine, one or more combinations of evaluation metrics are used. There are several methods available to evaluate the performance of the recommender systems [37, 39].

2.5.1 Recall and Precision

Recall and precision are most commonly used metrics to evaluate recommendation engines [40]. These metrics can be explained by a confusion matrix [41] as shown in Table 2.5.

TP = True Positive denotes the number of relevant items that are recommended by the system.

TN = True Negative denotes the number of irrelevant items that are cor-

	Recommeded	Not Recommended
Relevant	TP	FN
Irrelevant	FP	TN

Table 2.5: Confusion Matrix [41]

rectly not recommended by the system.

FP = False Positive denotes all irrelevant items which are incorrectly recommended by the system

FN = False Negative denotes relevant items but the system failed to recommend.

Based on Table 2.5, precision is calculated as ratio of the relevant items from recommended items to the number of all recommended items. It is given in Equation 2.13.

$$Precision = \frac{TP}{TP + FP} \qquad (2.13)$$

Based on Table 2.5, recall is calculated as the ratio of relevant items from recommended items to the number of all relevant items. It is given in Equation 2.14.

$$Recall = \frac{TP}{TP + FN} \qquad (2.14)$$

Larger value of recall and precision implies better recommendations.

2.5.2 Accuracy

Accuracy is an another evaluatiion metrics for recommender systems. Accuracy is the fraction of correct predictions predicted by a system. It is calculated by equation given in Equation 2.16

$$Accuracy = \frac{\text{Number of Correct predictions}}{\text{Total Number of Predictions}} \tag{2.15}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.16}$$

2.5.3 Mean Absolute Error (MAE)

Mean absolute error used to calculate the average deviation or error generated from predicted ratings and actual ratings [42].

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |Predicted_i - Actual_i| \tag{2.17}$$

Where,

$Predicted_i$ denotes predicted ratings given by user to the item i.

$Actual_i$ denotes actual ratings given by user to the item i.

n denotes number of items. With this formula, MAE can calculate the general performance of recommender systems but to compare engines with a different rating scale, we can normalize MAE by dividing by the mean MAE

value as shown in Equation 2.18.

$$NMAE = \frac{MAE}{\frac{1}{n}\sum_{i=1}^{n} Actual_i} \qquad (2.18)$$

2.5.4 Root Mean Square Error (RMSE)

RMSE is a variation of MAE. It also measures the average magnitude of the error. But it puts more weight on large errors as shown in Equation 2.19. RMSE can be defined as a square root of the average of squared deviations between predicted ratings and real ratings.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (Predicted_i - Actual_i)^2} \qquad (2.19)$$

Where,

$Predicted_i$ denotes predicted ratings given by user to the item i.

$Actual_i$ denotes actual ratings given by user to the item i.

n denotes number of items.

2.5.5 Mean Reciprocal Rank

Mean Reciprocal Rank is one of the rank-aware evaluation metrics. It is a retrieval measure that calculates the reciprocal of the rank at which the first relevant document was recommended [43]. From the list of generated recommendations, it finds the rank of the first relevant recommended item

and computes the reciprocal of that rank. It can be calculated as shown in Equation 2.20

$$MRR(O, U) = \frac{1}{|U|} \sum_{u \in \epsilon} \frac{1}{k_u} \tag{2.20}$$

Where,

k_u denotes the first relevant recommended item for user u.

U denotes the number of users.

ϵ denotes the number of recommended items.

MRR focuses on the first relevant item of the recommended list. A higher reciprocal rank implies a better recommendation engine.

2.5.6 Mean Average Precision at cutoff k (MAP at k)

MAP at k considers the subset of the list recommended by the system while measuring precision. It does not concern about ordering the items in the list. For each user for each relevant item, it computes the precision of the list through that item. After that it average sub-list precisions. This sub-list can be indexed by k. Mathematically it is show in Equation 2.21

$$MAP_i = \frac{1}{|R_i|} \sum_{k=1}^{|R_i|} P(R_i[k]) \tag{2.21}$$

Where,

R_i denotes rating for item i.

k denotes rank of recommended item from 1 to k.

P denotes the precision of the first relevant item. This calculation is done on the subset of recommendations from rank 1 through k. A higher mean precision value implies correct recommendations.

Summary

Various recipe recommender systems are built to understand user's taste based on a user's preferences. The work done by Freyne and Berkosvky uses a content-based technique to recommend recipes based on ingredients [6]. A similar approach is used in the first part of the hybrid approach in this thesis where ingredients, cook methods, and diet labels attributes are used as contents to get more knowledge about a user's preferences. To make this approach towards a healthy diet, calorie restrictions are also applied. The research of Mouzhi Ge, Francesco Ricci, and David Massimo introduced the health factor by incorporating calorie count with a matrix factorization approach[8]. A similar approach is used in the second part of the hybrid approach in this thesis where matrix factorization using the SVD algorithm is used along with calorie restrictions. This thesis aims to build a system that considers a user's preferences along with the health factor. To achieve this goal, the hybrid approach is used in this thesis that uses content-based and collaborative filtering along with calorie restrictions.

Chapter 3

Implementation

3.1   Environment Setup

In this thesis, different techniques of recommendation algorithms are used to recommend recipes to users. The process of building recommendation models and evaluating those models requires high computational time and memory. The initial analysis is done on my personal machine but to achieve better efficiency and to reduce the execution time of the application, I chose to run it in the cloud environment. In the current market, many cloud service providers are available such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), IBM Cloud. To accelerate the execution performance of the analysis of this application, the Google Cloud Platform (GCP) is used.

GCP platform provides custom, configurable high-performance virtual ma-

chines with ease of application or scripts deployment via its User Interface (UI). There are three basic steps to run an application on a Virtual Machine (VM) on GCP.

3.1.1 Create GCP Account

GCP account is linked with a user's Google account. One can enter into the GCP console dashboard by signing in with Google account. By default GCP creates a project for a user with the name 'My First Project'. Users can create a new project here. I have created a project with the name 'FeedMeRight'. As per Figure 3.1 project id and project number have been created by GCP for the 'FeedMeRight' project.
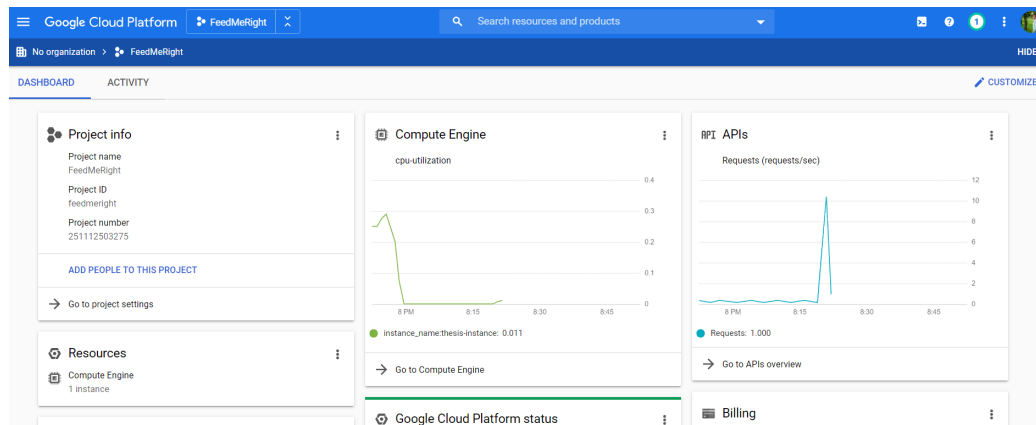
Figure 3.1: GCP Dashboard

3.1.2 Virtual Machine (VM) on GCP

Compute Engine lets you create and run virtual machines on Google Cloud Infrastructure. It helps to easily launch applications that require high computational capacity by offering scale, performance. Virtual machines are designed to be fast and to provide strong consistency of performance.
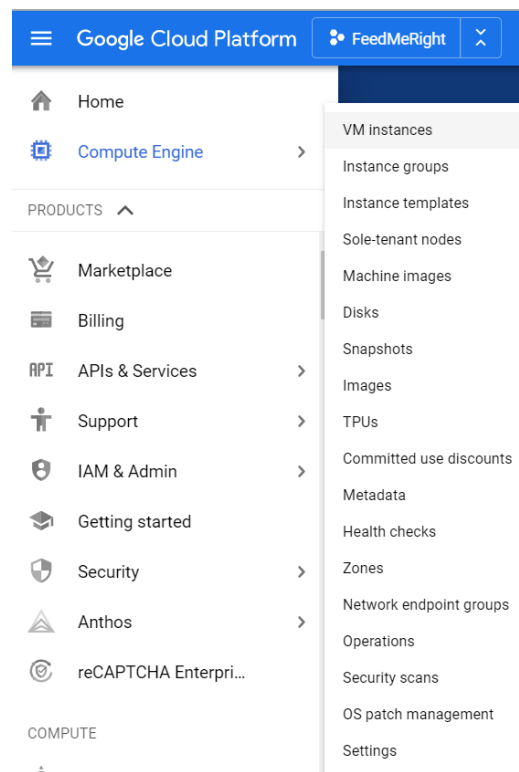

Figure 3.2: GCP Service List

As per Figure 3.2, with the help of 'Compute Engine', virtual machine instance can be created. Created VM instance's settings can be customized. Configurations for 'FeedMeRight' are used as shown in Figure 3.3.

Figure 3.3: VM Configuration for FeedMeRight

3.1.3 Python Scripts Deployment

After successful creation of VM instance, the application can be configured on this instance. To get all scripts created for 'FeedMeRight' Git software has been used. Git is a version control system to keep track of changes in the

source code. The source code for the project is stored on GitHub. GitHub is a cloud-based hosting service that lets you manage Git repositories. With the help of Git, we can pull all the source code stored in the cloud repository. To run the application, the installation of a set of libraries are required on VM.

- **Numpy:** Numpy library provides support for large, multi-dimensional arrays and matrices. Also, it supports high-level mathematical functions.

- **Pandas:** Pandas library provides data analysis and manipulation. It is a fast, powerful tool that offers data structure and operations for large files like CSV or TSV.

- **Matplotlib:** Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. It is based on Numpy and Pandas libraries.

- **Scikit-learn:** Scikit-learn is a machine learning library that provides various machine learning algorithms.

- **NLTK:** NLTK is a toolkit to process human's natural language.

- **SciPy:** SciPy library is used for scientific and technical computing. It builds on Numpy.

- **Anaconda:** Anaconda is a python package management and deployment software that makes it easy for developers to create different virtual environments.

- **Jupyter NoteBook:** Jupyter Notebook is a web application that can be used to create and share documents that have live code in it. It provides User Interface that makes easy interaction and visualization of results.

- **PyCharm:** PyCharm is an Integrated Development Environment (IDE) used for python programming language.

3.2 Data Preparation

Kaggle is a public platform which provides various types of data for research purpose. The dataset 'foodRecSys-V1' used in this thesis is received from Kaggle [44]. This dataset contains recipes and ratings crawled from 'All-recipes.com'. It consists of three files. The 'core-data_recipe.csv' file has all information about recipes such as 'recipe_id', 'recipe_name', 'image_url', 'ingredients', 'cooking_directions', 'nutritions' as shown in Figure 3.4. It has 45630 recipes with 6 columns.

Figure 3.4: Raw Recipe Data

The other two files 'core-data-train_rating.csv' and 'core-data-test_rating.csv' provide user interactions with recipes. User interaction refers to a record in the file where a user has given a rating to at least one recipe. As shown in Figure 3.5, user - 5215572 has given ratings to two recipes. These two files are combined into a single file that has all user interactions. Total user recipe interactions in this merged file are 960386.

53

Figure 3.5: Raw User-Interaction Data

3.2.1 Features Extraction

A content-based filtering algorithm relies on the contents of the items. In our dataset, recipes are items that are rated by users. To recommend any recipe for a user, it is important to understand the features of the recipe that are relevant for a user. This thesis considers 'Ingredients', 'Cook Method', 'Calories' and 'Diet Labels' features to find similarities between recipes.

3.2.1.1 Ingredients Extraction

The recipe's data received from Kaggle has the ingredients column in a clean format up to a certain level. To get a single ingredient for each recipe, the 'NLTK' library has been used. It tokenizes sentences into words. Raw ingredients data has many irrelevant words for predicting similarity between ingredients such as 'white' from 'egg white', 'frozen' from 'frozen chicken', 'thawed' from 'thawed rotis', 'piece from pork piece'. Such custom keywords are filtered out using a recipe_stopwords list to make ingredients very specific. The difference between raw ingredients and cleaned ingredients is illustrated

54

in Figure 3.6 Figure 3.7. Irrelevant words such as 'thawed' and 'white' present in the first row of the pre-processed ingredients have been removed in post-processed ingredients. On this clean data, lemmatization is performed to transform a word into its existing word. Part-of-speech tags considered are 'nouns' and 'adjectives'. Example 'potatoes' will be transformed to 'potato' as show in Figure 3.8

```
df = recipe_df['ingredients'].head(5)
df

Out[31]: 0
Parmesan cheese^ground black pepper^garlic powder^frozen puff pastry thawed^egg white
1    unsalted butter^chopped onion^cornmeal^all-purpose flour^white sugar^baking powder^salt^baking soda^buttermilk^eggs^shr
edded pepperjack cheese^frozen corn kernels^roasted marinated red bell peppers^chopped fresh basil
2
hot water^margarine^white sugar^salt^cold water^active dry yeast^all-purpose flour^eggs
3                                                  white sugar^vegetable oil^eggs^sifted all-purpose flour^ba
king soda^salt^ground cinnamon^ground nutmeg^water^cooked and mashed sweet potatoes^chopped pecans
4
active dry yeast^lukewarm milk^white sugar^unbleached all-purpose flour^salt^butter
```

Figure 3.6: Pre-Processed Ingredients

Out[29]:

	clean_ingredients
0	parmesan cheese pepper garlic puff pastry egg
1	butter onion cornmeal all-purpose flour sugar baking baking buttermilk eggs pepperjack cheese corn kernels marinated bell peppers basil
2	margarine sugar yeast all-purpose flour eggs
3	sugar vegetable oil eggs sifted all-purpose flour baking cinnamon nutmeg and sweet potatoes pecans
4	yeast lukewarm milk sugar all-purpose flour butter

Figure 3.7: Post-Processed Ingredients

```
data =  [['butter', 'onion', 'cornmeal', 'flour', 'sugar', 'baking', 'baking', 'buttermilk', 'eggs', 'pepperjack', 'cheese', 'corn', 'kernels', 'marinated',
'bell', 'peppers', 'basil']]

data_lemmatized =  [['butter', 'onion', 'cornmeal', 'flour', 'sugar', 'buttermilk', 'egg', 'cheese', 'corn', 'kernel', 'pepper']]
```

Figure 3.8: Post-Lemmatization Ingredients

3.2.1.2 Cook Method Extraction

Ingredients are most significant in measuring the similarity between recipes but instead of only ingredients, Wang et al. [49] represented the recipes as graphs which are built on ingredients and cooking directions that can be used to easily aggregate dishes. The University of Minnesota has predefined glossary of cooking methods [45] that has 74 cooking methods such as 'bake', 'steam', 'fry'. Recipes dataset of this thesis has a 'cooking_direction' column that contains the method of cooking. It is a set of sentences that provides all instructions for a recipe as shown in Figure 3.9.

index		cooking_directions
0	0	{'directions': u'Combine parmesan cheese, pepper and garlic powder. Unfold pastry sheets onto cutting board. Brush lightly with egg white; sprinkle each sheet with 1/4 of the cheese mixture. Lightly press into pastry, turn over; repeat. Cut each sheet into 12 (1-inch) strips; twist.\nPlace on ungreased cookie sheet and bake in 350 degrees F (175 degrees C) oven for 15 minutes or until golden brown.'}
1	1	{'directions': u'Prep\n25 m\nCook\n55 m\nReady In\n1 h 40 m\nPreheat oven to 400 degrees F (205 degrees C). Butter a 9x9x2 inch baking pan.\nMelt 1 tablespoon butter in medium nonstick skillet over medium-low heat. Add onion and saute until tender, about 10 minutes. Cool.\nMix cornmeal with the flour, baking powder, sugar, salt, and baking soda in large bowl. Add 7 tablespoons butter and rub with fingertips until mixture resembles coarse meal.\nWhisk buttermilk and eggs in medium bowl to blend. Add buttermilk mixture to dry ingredients and stir until blended. Mix in cheese, corn, red peppers, basil, and onion. Transfer to prepared pan.\nBake cornbread until golden and tester inserted comes out clean, about 45 minutes. Cool 20 minutes in pan. Cut cornbread into squares.'}
2	2	{'directions': u'Melt margarine in hot water. Add sugar and salt and stir. Add cold water and yeast. Stir to dissolve yeast.\nAdd 3 cups flour and mix. Add eggs and 2 1/2 - 3 cups more flour. Mix, cover and let rise until dough doubles in size.\nPunch down and let rise 30 more minutes or until doubles.\nMake walnut size balls of dough. Place about 2 inches apart in well-buttered 9 x 13 inch pan. Bake in a preheated 350 degrees F (175 degrees C) oven for 30-45 minutes. Brush top of rolls with margarine while hot.'}
3	3	{'directions': u'Combine sugar and oil; beat well. Add eggs and beat. Combine flour, baking soda, salt, cinnamon and nutmeg. Stir flour mixture into egg mixture alternately with water. Stir in sweet potatoes and chopped nuts.\nPour batter into greased 9x5 inch loaf pan (or 2 small loaf pans). Bake at 350 degrees F (175 degrees C) for about one hour.'}
4	4	{'directions': u'In a small bowl, dissolve the yeast in the milk and add the sugar. In another bowl, sift the flour and salt together and add the cooled melted butter.\nAdd the yeast mixture to the flour mixture, and turn out onto a floured counter and knead until the dough is smooth and elastic. Place the dough in an oiled bowl, cover with a clean towel and let rise in a warm, draft free place to 45 minutes.\nTurn dough out onto the freshly floured board and shape into 9 balls. Place dough balls into a buttered and floured 9 inch square pan. Let them sit, covered for another 15 minutes to rise again. Preheat the oven to 425 degrees F (220 degrees C).\nBake for 15-20 minutes until browned and puffed. Split open and serve warm.'}

Figure 3.9: Cooking Directions

To get a cooking method from cooking direction, 'NLTK Processing' is performed on 'cooking_direction' column. The first step of this processing is to convert the set of instructions into words. Conversion of sentences to words is done using 'NLTK Tokenizer.' The next step is to remove 'stop words' from sentences. In natural language processing, stop words refer to words

that do not have meaningful information such as 'a', 'and', 'an', 'the'. These words are considered as noisy data and can be ignored. These stopwords are downloaded from the 'NLTK' library. The result is a list of keywords for cooking methods and ingredients. The common words from this result and predefined glossary of cooking methods are extracted and mapped as cooking methods used in the associated recipe. The list of cooking methods used in recipes is depicted in Figure 3.10.

	index	cook_method
0	0	bake
1	1	mix bake stir blend
2	2	mix bake stir
3	3	stir beat chop bake batter
4	4	bake sift knead

Figure 3.10: Cooking Methods

3.2.1.3   Calories Extraction

A calorie is a unit of an energy. Calories in any food refers to the energy people get by consuming food. Recipe dataset has 'nutrition' column. For each recipe, nutrition values are specified with quantity. For example, 'Crispee Cheese twists' recipe has 121 calories as depicted in Figure 3.11. This information is stored in a nested dictionary. After extracting calorie's information it is stored in the 'calories' column as showin in Figure 3.12.

57

recipe_name	nutritions	calories

{u'niacin': {u'hasCompleteData': True, u'name': u'Niacin Equivalents', u'amount': 1.305051, u'percentDailyValue': u'10', u'displayValue': u'1', u'unit': u'mg'}, u'sugars': {u'hasCompleteData': True, u'name': u'Sugars', u'amount': 0.190441, u'percentDailyValue': u'0', u'displayValue': u'0.2', u'unit': u'g'}, u'sodium': {u'hasCompleteData': True, u'name': u'Sodium', u'amount': 78.67492, u'percentDailyValue': u'3', u'displayValue': u'79', u'unit': u'mg'}, u'carbohydrates': {u'hasCompleteData': True, u'name': u'Carbohydrates', u'amount': 9.370335, u'percentDailyValue': u'3', u'displayValue': u'9.4', u'unit': u'g'}, u'vitaminB6': {u'hasCompleteData': True, u'name': u'Vitamin B6', u'amount': 0.007097583, u'percentDailyValue': u'< 1', u'displayValue': u'< 1', u'unit': u'mg'}, u'calories': {u'hasCompleteData': True, u'name': u'Calories', u'amount': 120.7066, u'percentDailyValue': u'6', u'displayValue': u'121', u'unit': u'kcal'}, u'thiamin': {u'hasCompleteData': True, u'name': u'Thiamin', u'amount': 0.08243293, u'percentDailyValue': u'8', u'displayValue': u'< 1', u'unit': u'mg'}, u'fat': {u'hasCompleteData': True, u'name': u'Fat', u'amount': 8.258162, u'percentDailyValue': u'13', u'displayValue': u'8.3', u'unit': u'g'}, u'folate': {u'hasCompleteData': True, u'name': u'Folate', u'amount': 16.15505, u'percentDailyValue': u'9', u'displayValue': u'16', u'unit': u'mcg'}, u'caloriesFromFat': {u'hasCompleteData': True, u'name': u'Calories from Fat', u'amount': 74.32346, u'percentDailyValue': u'-', u'displayValue': u'74', u'unit': u'kcal'}, u'calcium': {u'hasCompleteData': True, u'name': u'Calcium', u'amount': 20.95548, u'percentDailyValue': u'3', u'displayValue': u'21', u'unit': u'mg'}, u'fiber': {u'hasCompleteData': True, u'name': u'Dietary Fiber', u'amount': 0.3293675, u'percentDailyValue': u'1', u'displayValue': u'0.3', u'unit': u'g'}, u'magnesium': {u'hasCompleteData': True, u'name': u'Magnesium', u'amount': 4.213947, u'percentDailyValue': u'2', u'displayValue': u'4', u'unit': u'mg'}, u'iron': {u'hasCompleteData': True, u'name': u'Iron', u'amount': 0.5593101, u'percentDailyValue': u'6', u'displayValue': u'< 1', u'unit': u'mg'}, u'cholesterol': {u'hasCompleteData': True, u'name': u'Cholesterol', u'amount': 1.466667, u'percentDailyValue': u'< 1', u'displayValue': u'1', u'unit': u'mg'}, u'protein': {u'hasCompleteData': True, u'name': u'Protein', u'amount': 2.300013, u'percentDailyValue': u'5', u'displayValue': u'2.3', u'unit': u'g'}, u'vitaminA': {u'hasCompleteData': True, u'name': u'Vitamin A - IU', u'amount': 7.767053, u'percentDailyValue': u'< 1', u'displayValue': u'8', u'unit': u'IU'}, u'potassium': {u'hasCompleteData': True, u'name': u'Potassium', u'amount': 18.26903, u'percentDailyValue': u'< 1', u'displayValue': u'18', u'unit': u'mg'}, u'saturatedFat': {u'hasCompleteData': True, u'name': u'Saturated Fat', u'amount': 2.25785, u'percentDailyValue': u'11', u'displayValue': u'2.3', u'unit': u'g'}, u'vitaminC': {u'hasCompleteData': True, u'name': u'Vitamin C', u'amount': 0.02419375, u'percentDailyValue': u'< 1', u'displayValue': u'< 1', u'unit': u'mg'}}\n

Figure 3.11: Recipe - Nutrition

recipe_name	nutritions	calories

{u'niacin': {u'hasCompleteData': True, u'name': u'Niacin Equivalents', u'amount': 1.305051, u'percentDailyValue': u'10', u'displayValue': u'1', u'unit': u'mg'}, u'sugars': {u'hasCompleteData': True, u'name': u'Sugars', u'amount': 0.190441, u'percentDailyValue': u'0', u'displayValue': u'0.2', u'unit': u'g'}, u'sodium': {u'hasCompleteData': True, u'name': u'Sodium', u'amount': 78.67492, u'percentDailyValue': u'3', u'displayValue': u'79', u'unit': u'mg'}, u'carbohydrates': {u'hasCompleteData': True, u'name': u'Carbohydrates', u'amount': 9.370335, u'percentDailyValue': u'3', u'displayValue': u'9.4', u'unit': u'g'}, u'vitaminB6': {u'hasCompleteData': True, u'name': u'Vitamin B6', u'amount': 0.007097583, u'percentDailyValue': u'< 1', u'displayValue': u'< 1', u'unit': u'mg'}, u'calories': {u'hasCompleteData': True, u'name': u'Calories', u'amount': 120.7066, u'percentDailyValue': u'6', u'displayValue': u'121', u'unit': u'kcal'}, u'thiamin': {u'hasCompleteData': True, u'name': u'Thiamin', u'amount': 0.08243293, u'percentDailyValue': u'8', u'displayValue': u'< 1', u'unit': u'mg'}, u'fat': {u'hasCompleteData': True, u'name': u'Fat', u'amount': 8.258162, u'percentDailyValue': u'13', u'displayValue': u'8.3', u'unit': u'g'}, u'folate': {u'hasCompleteData': True, u'name': u'Folate', u'amount': 16.15505, u'percentDailyValue': u'9', u'displayValue': u'16', u'unit': u'mcg'}, u'caloriesFromFat': {u'hasCompleteData': True, u'name': u'Calories from Fat', u'amount': 74.32346, u'percentDailyValue': u'-', u'displayValue': u'74', u'unit': u'kcal'}, u'calcium': {u'hasCompleteData': True, u'name': u'Calcium', u'amount': 20.95548, u'percentDailyValue': u'3', u'displayValue': u'21', u'unit': u'mg'}, u'fiber': {u'hasCompleteData': True, u'name': u'Dietary Fiber', u'amount': 0.3293675, u'percentDailyValue': u'1', u'displayValue': u'0.3', u'unit': u'g'}, u'magnesium': {u'hasCompleteData': True, u'name': u'Magnesium', u'amount': 4.213947, u'percentDailyValue': u'2', u'displayValue': u'4', u'unit': u'mg'}, u'iron': {u'hasCompleteData': True, u'name': u'Iron', u'amount': 0.5593101, u'percentDailyValue': u'6', u'displayValue': u'< 1', u'unit': u'mg'}, u'cholesterol': {u'hasCompleteData': True, u'name': u'Cholesterol', u'amount': 1.466667, u'percentDailyValue': u'< 1', u'displayValue': u'1', u'unit': u'mg'}, u'protein': {u'hasCompleteData': True, u'name': u'Protein', u'amount': 2.300013, u'percentDailyValue': u'5', u'displayValue': u'2.3', u'unit': u'g'}, u'vitaminA': {u'hasCompleteData': True, u'name': u'Vitamin A - IU', u'amount': 7.767053, u'percentDailyValue': u'< 1', u'displayValue': u'8', u'unit': u'IU'}, u'potassium': {u'hasCompleteData': True, u'name': u'Potassium', u'amount': 18.26903, u'percentDailyValue': u'< 1', u'displayValue': u'18', u'unit': u'mg'}, u'saturatedFat': {u'hasCompleteData': True, u'name': u'Saturated Fat', u'amount': 2.25785, u'percentDailyValue': u'11', u'displayValue': u'2.3', u'unit': u'g'}, u'vitaminC': {u'hasCompleteData': True, u'name': u'Vitamin C', u'amount': 0.02419375, u'percentDailyValue': u'< 1', u'displayValue': u'< 1', u'unit': u'mg'}}\n   ... 120.7066

Figure 3.12: Recipe - Calories

3.2.1.4 Diet Labels Extraction

According to U.S. Food and Drug Administration (FDA) [46], %DV is the percentage of the Daily Value for each nutrient in a serving of the food. Percentage Daily Value can inform if a serving of food is high or low in a nutrient. The general guide is, 5% DV or less of a nutrient per serving is considered low and 20% DV or more of a nutrient per serving is considered high. From this

information, recipes are broadly divided into five categories such as **'high-protein'**, **'highfiber'**, **'lowfat'**, **'lowcarb'**, **'lowsodium'** and **'balanced'**. If %DV value is less than 5% then recipe will fall under low nutrition label category otherwise in high nutrition label category. For example, Figure 3.13 illustrates that for recipe 'Crispee Cheese Twist', percentage daily value for carbohydrates is 3 and for sodium is 3 which is less than 5%. Hence 'Crispee Cheese Twist' will fall under 'Low-Carb' and 'Low-Sodium' category under column 'diet_labels'.

Figure 3.13: Recipe - Diet Labels

3.2.1.5 User Information

For a user 'height_in_inches', 'weight_in_lb', 'age' in years, 'gender' and 'activity' attributes are considerd. An activity is divided into 5 categories. 1. Sedentary 2. Lightly Active 3. Moderately Active 4.Very Active 5. Extra Active. The user information such as height, weight, gender, age, activities is generated using python script. The height range for an adult is considered

from 52 inches to 80 inches. The weight range for an adult is considered from 64 lbs to 175 lbs. The Harris–Benedict equation is used to estimate an individual's basal metabolic rate (BMR). This estimated BMR value multiplied by a number that corresponds to a user's activity level provides the approximate daily kilocalorie intake to maintain current body weight [47]. To calculate BMR of male and female, the Harris–Benedict equation is given below.

$$BMR(Male) = 66 + (6.3 * Weight\_lb) + (12.9 * Height\_inch) - (6.8 * age)$$

(3.1)

$$BMR(Female) = 655 + (4.3 * Weight\_lb) + (4.7 * Height\_inch) - (4.7 * age)$$

(3.2)

The relationship between BMR and user's activity level is depicted in Table 3.1. The approximate daily kilocalorie intake to maintain current body of a user, is product of BMR to lifestyle factor as shown in Table 3.1.

Lifestyle	Multiplication Factor	Approximate Calorie Intake
Sedentary	1.2	BMR * 1.2
Lightly Active	1.375	BMR * 1.375
Moderately Active	1.55	BMR *1.55
Very Active	1.725	BMR *1.725
Extra Active	1.9	BMR *1.9

Table 3.1: Calorie Intake based on BMR

60

3.3 Implementation Approaches

3.3.1 Content Based Model

The content-based model uses Vector Space Models (VSM) of a user and an item to find the similarity between two vectors. The overview of content-based model is described in the Figure 3.14 where all user interactions are denoted by 'Users Ratings' and all recipes information is denoted by 'Recipes'.
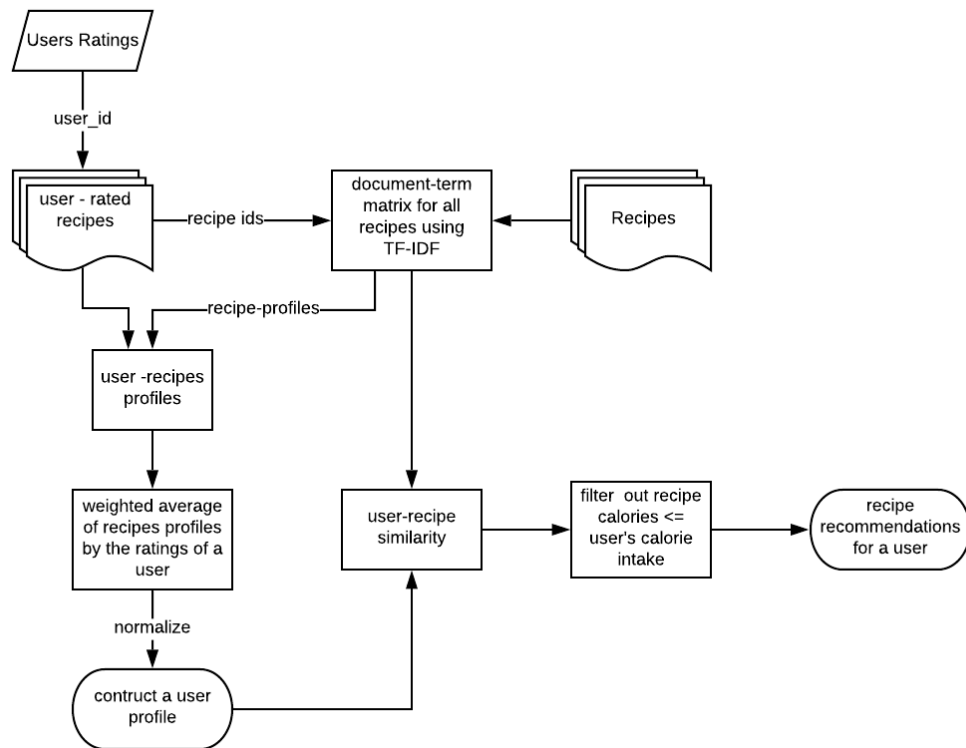
Figure 3.14: Content Based Filtering Work-Flow

Required steps to generated a user profile and a recipe profile in the flowchart are discussed below.

- All recipe profiles are generated based on the relevancy of terms to be considered. The term may vary based on the application area. In this thesis, ingredients, cooking methods, and diet labels are considered. The relevancy of terms in the documents is measured by TF-IDF as discussed in section Term Frequency - Inverse Document Frequency (TF-IDF). A document-term matrix is generated and stored for all recipes.

- Rated recipes for a single user are filtered from all user interactions. Rated recipes are then filtered out from the document-term matrix.

- To construct a user profile, recipes vectors rated by a user are considered. Then a weighted average of rated recipe profiles is calculated. The user profile is further normalized on the weighted values.

- Similarity between the user profile and all recipe profiles from the dataset is measured by cosine similarity as discussed in section Cosine Similarity. The system should not recommend the same recipes that are rated by a user previously. Hence, previously rated recipes by a user are omitted before calculating relevance. Remaining recipes are then sorted in descending order of similarity scores.

- On the resultant recipes vectors, a calorie intake filter is applied, com-

paring recipe calories and user required calories. In calorie filter, the system considers only those recipes whose calories are less or equal to user's calorie intake requirement. These recipes are offered as recommendations to the user.

Figure 3.15: Content Based Filtering Combined Work-Flow

3.3.1.1 Content-Based Using Ingredients

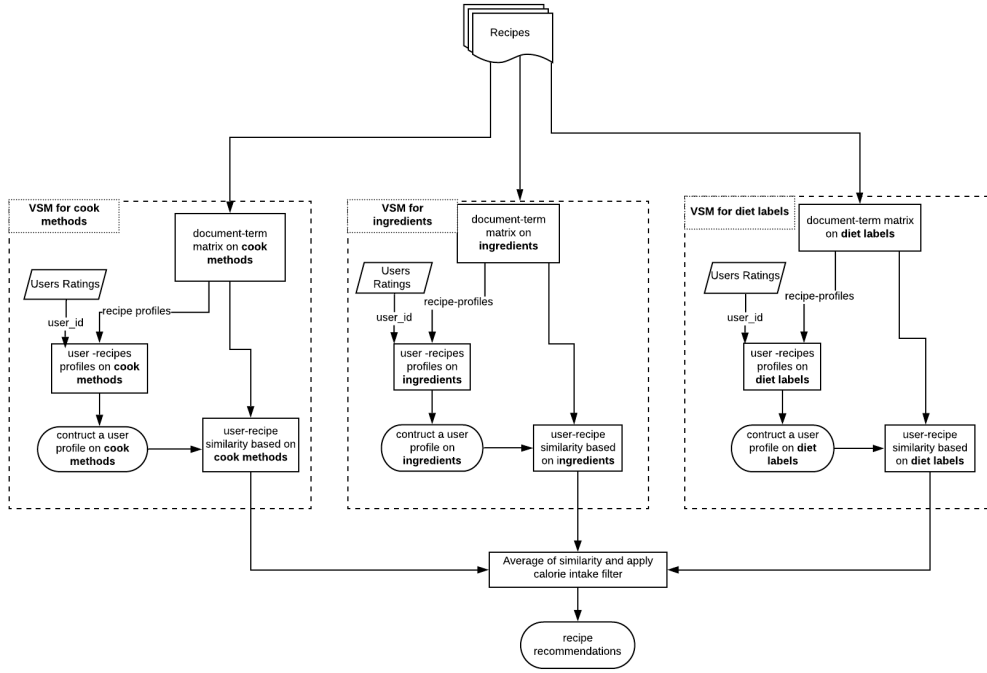Recipe ingredients are used to calculate recommendations in content-based using the ingredients approach. The user profile and recipe profiles are generated using ingredients in the recipes. The steps to generate recommendations for a user using ingredients in the content-based model are discussed below.

63

All ingredients from recipes are extracted. The similarity between recipes is considered based on ingredients but many ingredients occur in all recipes such as salt. To get the importance of ingredients that are relevant to the recipe and to negate the effect of high-frequency terms, the concept of TF-IDF has been used as discussed in section Term Frequency - Inverse Document Frequency (TF-IDF). The recipe vector or recipe profile is calculated for each recipe in the dataset. The 'TfidfVectorizer' function from 'sklearn' library has been used to calculate TF-IDF. The VSM for ingredients in the Figure 3.15 represents Vector Space Model using ingredients. The weighted average of a user rated recipe profiles is calculated to construct a user profile. The similarity between a user profile and all recipe profiles in the dataset is calculated using cosine similarity. The 'linear_kernel' function from the 'scikit learn' library has been used for the same. The resultant profiles are listed in descending order of similarity score to get more relevant recipes at the top. At this point recipes already rated by a user are omitted. From the user's information, the BMR and the required calorie intake per meal are calculated as discussed in section User Information. The resultant set of recipes is offered as recommendations.

3.3.1.2 Content-Based Using Ingredients and Cook Methods

Recipe ingredients and cook methods are used to calculate recommendations in content-based using ingredients and cook method approach. The extraction process of cook methods from cooking directions is discussed in section

Cook Method Extraction. The user profile and recipe profiles are generated using the cooking method in the recipes. The VSM for cook method in the Figure 3.15 represents Vector Space Model using cook method. The recipe vector is calculated for each recipe in the dataset. The constructed user profile is based on cook methods of the recipes. The similarity between a user profile and all recipe profiles in the dataset is calculated using cosine similarity. Further, the average of cosine similarity scores generated using ingredients and cook methods is considered as a result. The resultant profiles are listed in descending order of similarity score to get more relevant recipes at the top. At this point recipes already rated by a user are omitted. From the user's information, the BMR and the required calorie intake per meal are calculated as discussed in section User Information. The resultant set of recipes is offered as recommendations.

3.3.1.3 Content-Based Using Ingredients, Cook Methods and Diet Labels

Recipe ingredients, cook methods and diet labels are used to calculate recommendations in content-based using ingredients, cook methods and diet labels approach. Extraction process of diet labels from nutrition attribute is discussed in the section Diet Labels Extraction. In addition to ingredients and cook methods, the user profile and recipe profiles are generated using diet labels in the recipes.. The VSM for diet labels in the Figure 3.15 represents Vector Space Model using diet labels. The recipe vector is calculated for each

recipe in the dataset. The constructed user profile is based on diet labels of the recipes. The similarity between a user profile and all recipe profiles in the dataset is calculated using cosine similarity. Further, the average of cosine similarity scores generated using ingredients, cook methods and diet labels is considered as a result. The resultant profiles are listed in descending order of similarity score to get more relevant recipes at the top. At this point recipes already rated by a user are omitted. From the user's information, the BMR and the required calorie intake per meal are calculated as discussed in section User Information. The resultant set of recipes is offered as recommendations.

3.3.2   Collaborative Filtering Model

Model-based collaborative filtering for the user and recipes was applied. This approach utilizes information related to user ratings on the recipes. A Singular Value Decomposition (SVD) algorithm has been applied to get the recommendations. The below approach was followed to get recommendations using SVD.
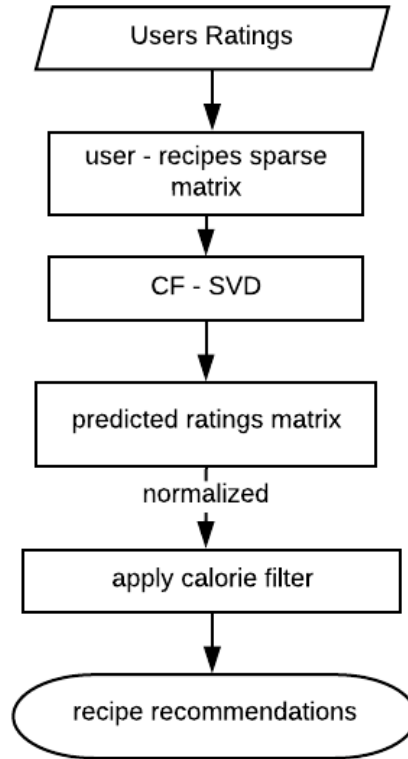
Figure 3.16: Collaborative Filtering- SVD Work-Flow

- All user ids and recipe ids are extracted in such a way that every user id and recipe id represents a unique relationship via rating. The user id refers to unique id given to the user and recipe id refers to a unique id given to the recipe in the dataset. The matrix of user-recipes ratings is formed such as a user represents row and recipes represents columns. The cell value represents a rating for a recipe given by a user. From this matrix, the sparse matrix was created.

- The sparse matrix is the same as the above matrix except it has high efficiency to perform large mathematical operations. To create a sparse matrix, 'csr_matrix' function is used from 'scipy.sparse' library. The generated sparse matrix was sent as an input to the SVD algorithm.

- SVD algorithm runs Principal Component Analysis (PCA) on the user-rating matrix to give back factors of the rating matrix as discussed in Singular Value Decomposition (SVD). The dot product of these factors returns a rating matrix with predicted ratings.

- Further, normalization was performed to get all predicted ratings for all users, for all recipes.

3.3.3 Hybrid Model(CB and CF)

The hybrid model is a combination of content-based and collaborative filtering. The work-flow of the application using the hybrid technique is depicted in the Figure 3.17.

Figure 3.17: Hybrid Work-Flow

- The system prompts to enter a user id. The existence of an entered user id is cross-verified by the system.

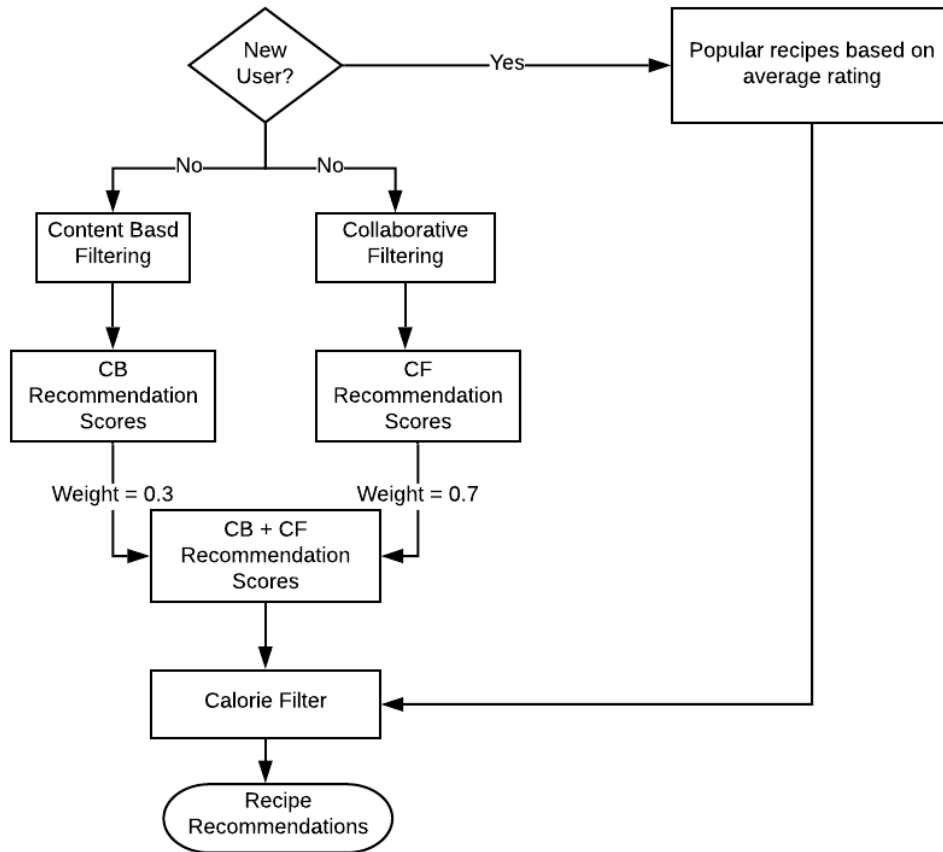- If user id exists, a content-based model that consists of ingredients, cook method and diet labels runs for this user and generates scores for relevant recipes. Once the content-based execution finishes, the collaborative filtering model runs and generates predicted ratings for

69

all users. From this result set, predictions for a requested user are filtered out.

- While combining scores, the weighting factor of 0.3 is applied to the content-based, and the weighting factor of 0.7 is applied to the collaborative filtering. The experiment was done considering equal weighing factors but it resulted in lower recall value of hybrid than collaborative. To improve the performance of the hybrid approach in terms of higher recall, weights were adjusted and finalized 0.3 weighing factor for content-based and 0.7 weighing factor for collaborative.

- Scores are sorted in the descending order to get more relevant items at the top. At this point, recipes that are already rated by the user are omitted. Next, a calorie filter is applied to filter out healthy recipes.

- Resultant set of recipes is offered as recommendations to the user.

- If entered user id does not exist in the system, the system prompts to enter user details such as 'height in inches', 'weight in lbs', 'age', 'gender' and 'activity' option. The user's BMR and calorie intake per dish are calculated as discussed in the User Information.

- Here, the user's information goes through a popularity-based algorithm. This model finds the most popular recipes by considering the average rating for recipes and the maximum number of ratings per recipe. A list of these recipes is then sorted in descending order and filtered out

by comparing recipe calories that are less than or equal to the user's calorie requirements. The resultant set of recipes is further categorized into diet labels and represented as recommendations to the user.

- System provides an option to give feedback for a recipe in the form of ratings. The user can enter a recipe name along with its rating. This feedback is saved for future model generation and recipe recommendation for this new user.

3.4   Experiments

3.4.1   Training and Testing Dataset
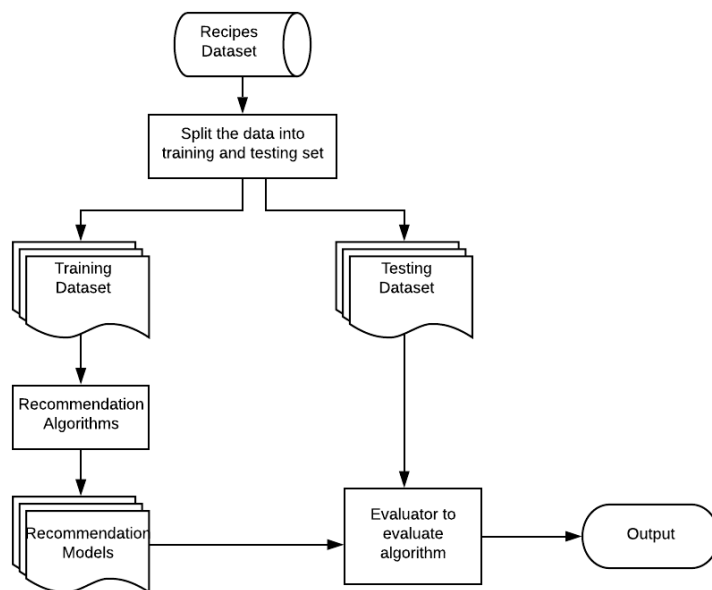
Figure 3.18: Train Test Evaluation

The overview of the training and testing dataset is given in Figure 3.18. The full dataset was randomly split into two parts - training dataset and testing dataset. The 80% of the data was split into a training dataset while 20% comprises a testing set. The training data set was fed to the algorithm to build a model and then the built model was evaluated on the test dataset to calculate recall, precision, and accuracy for experiments. Recall, precision

and accuracy evaluation metrics are discussed in the section Evaluation Metrics for Recommendation Systems.

Data statistics used for the experiments include:

- Total number of Recipes = 43,922

- Total number of users = 20,000

- Total number of user-recipe interactions = 655,609

- Train dataset - number of user-recipe interactions = 524,487

- Test dataset - number of user-recipe interactions = 131,122

The performed experiments are explained below. The results of the below experiments are discussed in the chapter Results.

3.4.2   Experiment 1

The purpose of this experiment was to evaluate user recommendations using a content-based algorithm on ingredients described in section Content-Based Using Ingredients. For this experiment, 'Ingredients' attribute is used. The training and testing datasets are split as explained in section Training and Testing Dataset. The experiment was performed for an output set of 5, 10, and 20 recommendations. There were 3 output sets considered for evaluation. First of them was recommending 5 items, second was recommending 10 items, third was recommending 20 items. The evaluation was done on all of these 3 output sets.

3.4.3 Experiment 2

The purpose of this experiment was to evaluate the user recommendations using content-based algorithm on ingredients and cook method described in the Content-Based Using Ingredients and Cook Methods. For this experiment, 'Ingredients' and 'Cook Method' attributes are used. The training and testing datasets are split as explained in the section Training and Testing Dataset. The experiment was performed for an output set of 5, 10 and 20 recommendations.

3.4.4 Experiment 3

The purpose of this experiment was to evaluate the user recommendations using content-based algorithm on ingredients, cook method and diet labels described in the Content-Based Using Ingredients, Cook Methods and Diet Labels. For this experiment, 'Ingredients', 'Cook Method' and 'Diet Label' attributes are used. The training and testing datasets are split as explained in the section Training and Testing Dataset. The experiment was performed for an output set of 5, 10 and 20 recommendations.

3.4.5 Experiment 4

The purpose of this experiment was to evaluate the user recommendations using collaborative filtering - SVD algorithm. Ratings given to recipes were extracted from the dataset and unrated combinations of user and recipes were

predicted as mentioned in section Collaborative Filtering Model. Experiment 4 determined the recall, precision, and accuracy of the results on a randomly generated testing dataset. The experiment was performed for an output set of 5, 10 and 20 recommendations.

3.4.6 Experiment 5

The purpose of this experiment was to evaluate the user recommendations using a combination of a content-based model that consists of ingredients, cook method, diet labels and collaborative filtering - SVD algorithm. Recommendations from content-based and collaborative filtering are combined by applying weightage as described in Hybrid Model(CB and CF) Experiment 4 determined the recall, precision and accuracy of the results on a randomly generated testing dataset. The experiment was performed for an output set of 5, 10 and 20 recommendations.

Chapter 4

Results

4.1 Content Based Results

4.1.1 Experiment 1 Results

Table 4.1, row one, shows recall, precision and accuracy values for Experiment 1, performed on content-based model by considering only one attribute - ingredients. The values of recall, precision, and accuracy are very low as only one feature is considered. This result can be improved by incorporating another feature as explained in Experiment 2.

4.1.2 Experiment 2 Results

Table 4.1, row two, shows recall, precision and accuracy values for Experiment 2, performed on content-based model by considering two attributes

- ingredients and cook methods. Comparing the results of experiments 1 and 2, there is a significant increase in recall, precision, and accuracy due to the addition of cook methods attribute to ingredients. The recall is improved by 42%, precision is improved by 33% and accuracy is improved by 33% when compared to the results of experiment 1 with the addition of cook methods feature. Experiment 2 shows that cook methods aggregate recipes and provides more information about recipes which is a very significant factor to consider in terms of improving performance. The performance of the content-based algorithm can be further improved by incorporating diet-labels attribute as explained in Experiment 3.

4.1.3   Experiment 3 Results

Table 4.1, row three, shows recall, precision and accuracy values for Experiment 3, performed on content-based model by considering three attributes - ingredients, cook methods and diet-labels. Comparing the results of experiments 2 and 3, there is a slight increase in recall, precision and accuracy on adding a diet-labels attribute. The recall is improved by approximately 10%, precision is improved by 9% and accuracy is improved by 9% when compared to the results of experiment 2 with the addition of diet labels attribute. Although the addition of diet labels did not add much improvement, it still helped improve the overall performance of the content-based algorithm. Experiment 3 shows that, addition of more content or attributes helps in improving the performance of content-based algorithm.

Model Name	Recall at 10	Precision at 10	Accuracy at 10
CB using ingredients	0.078617241	0.024154169	0.024236065
CB using ingredients and cook method	0.105738164	0.032184732	0.032312734
CB using ingredients, cook method and diet labels	0.111494599	0.035216746	0.035334459

Table 4.1: Comparison between Evaluation of Content Based Attributes

4.2 Collaborative and Hybrid Results

4.2.1 Experiment 4 Results

Table 4.2, Table 4.3 and Table 4.4 row two shows recall, precision and accuracy values for 5, 10 and 20 recommendations, evaluated for Experiment 4. Comparing the results of experiment 3 and 4, there is substantial increase in recall, precision and accuracy values for collaborative filtering approach. The recall at 10 has increased from 0.11 to 0.37, precision at 10 has increased from 0.03 to 0.20 and accuracy has increased from 0.03 to 0.16 when compared to results of content-based using all three attributes. The results explains that collaborative filtering using SVD outperforms content-based model for the dataset used in this thesis.

4.2.2 Experiment 5 Results

Table 4.2, Table 4.3 and Table 4.4 row three shows recall, precision and accuracy values for 5, 10 and 20 recommendations, evaluated for Experiment 5. Comparing the results of experiments 3, 4 and 5, below are some observations.

- Significant increase in recall results for recommendations of 5, 10 and 20 across content-based, collaborative and hybrid models. It was expected to improve the performance of the hybrid model by combining a content-based and collaborative model. On comparing the recall at 10

79

for content-based and hybrid, we see a significant increase from 0.11 to 0.43. Similarly for collaborative and hybrid, recall at 10 has increased from 0.37 to 0.43.

- There is a substantial increase in the precision of a hybrid model when compared to content-based model results for recommendations of 5, 10 and 20. However, surprisingly there is a negligible decline in the precision of hybrid model compare to the collaborative model results for recommendations of 5, 10 and 20.

- The accuracy of the hybrid model has increased substantially when compared to the content-based model. There is an approximately 12% increase in accuracy at 5, 10 and 20 recommendations for the hybrid model when compared to the collaborative filtering model.

- The performance of the hybrid model was expected to be high in terms of recall, precision and accuracy. The hybrid model outperforms traditional approaches in terms of recall and accuracy metrics. On the other hand, there is a slight decline in the performance of precision compared to the collaborative model using SVD. This thesis aims to recommend recipes based on the user's taste and preferences where the relevancy of recipes is important to the user. Recall refers to the percentage of total relevant results correctly classified. In that case, the trade-off between recall and precision can be accepted.

Model Name	Recall at 5	Recall at10	Recall at 20
Content-based	0.082525879	0.111494599	0.14501192
Collaborative Filtering	0.245029414	0.376785432	0.460967698
Hybrid	0.256281818	0.43370876	0.554458889

Table 4.2: Recall comparison between CB, CF and Hybrid models

Model Name	Precision at 5	Precision at10	Precision at 20
Content-based	0.050411997	0.035216746	0.023729464
Collaborative Filtering	0.20572189	0.204567584	0.20473256
Hybrid	0.204950782	0.203729964	0.203750943

Table 4.3: Precision comparison between CB, CF and Hybrid models

Model Name	Accuracy at 5	Accuracy at10	Accuracy at 20
Content-based	0.050483648	0.035334459	0.0238395
Collaborative Filtering	0.197410308	0.166999335	0.118616613
Hybrid	0.203255028	0.183965403	0.136176365

Table 4.4: Accuracy comparison between CB, CF and Hybrid models

Chapter 5

Conclusion and Future Work

This thesis aims to recommend healthy recipes by considering the user's calorie intake requirements. Experiments were performed on a large dataset of recipes and users along with unique interactions between users and recipes. In a content-based filtering experiment, results showed better performance of recommendations and metrics such as recall, precision and accuracy, when more than one attribute (ingredients, cook method and diet-labels) were considered.

Collaborative filtering experiment's performance which was solely based on ratings without considering the complexity of a user's taste showed drastic improvements in recall precision and accuracy.

To get the best out of both systems, a hybrid approach was built combining content-based for user's taste and collaborative for a variety of other recipes, to get an overall good performance out of the system. Metrics from results in

section Results, shows that even if the performance of the Hybrid approach has increased slightly compared to collaborative filtering, it's more efficient in terms of recommendations due to consideration of user's preference and calorie intake restrictions.

As future work, we will consider more features in the content-based model such as recipe diversity. Along with calorie balance, we can also account for more nutrition and user's health information such as cholesterol, blood sugar levels, to recommend targeted healthier recipes.

Bibliography

[1] T. L. Nguyen. "A Framework for Five Big V's of Big Data and Orga- nizational Culture in Firms". In: *2018 IEEE International Conference on Big Data (Big Data)*, pp. 5411–5413.

[2] Paul Resnick and Hal R. Varian. "Recommender Systems". In: *Commun. ACM* 40.3 (Mar. 1997), pp. 56–58. ISSN: 0001-0782. DOI: `10.1145/245108.245121`. URL: `https://doi.org/10.1145/245108.245121`.

[3] Robin Burke, A. Felfernig, and Mehmet Göker. "Recommender Systems: An Overview". In: *Ai Magazine* 32 (Sept. 2011), pp. 13–18. DOI: `10.1609/aimag.v32i3.2361`.

[4] WHO. *Obesity and Overweight*. WHO.

[5] Aileen Robertson. *Food and health in Europe: a new basis for action.* 96. WHO Regional Office Europe, 2004.

[6] Jill Freyne and Shlomo Berkovsky. "Intelligent Food Planning: Personalized Recipe Recommendation". In: *Proceedings of the 15th In-*

ternational Conference on Intelligent User Interfaces. IUI '10. Hong Kong, China: Association for Computing Machinery, 2010, pp. 321–324. ISBN: 9781605585154. DOI: 10.1145/1719970.1720021. URL: https://doi.org/10.1145/1719970.1720021.

[7]   Morgan Harvey, Bernd Ludwig, and David Elsweiler. "You Are What You Eat: Learning User Tastes for Rating Prediction". In: *String Processing and Information Retrieval.* Ed. by Oren Kurland, Moshe Lewenstein, and Ely Porat. Cham: Springer International Publishing, 2013, pp. 153–164. ISBN: 978-3-319-02432-5.

[8]   Mouzhi Ge et al. "Using Tags and Latent Factors in a Food Recommender System". In: *Proceedings of the 5th International Conference on Digital Health 2015.* DH '15. Florence, Italy: Association for Computing Machinery, 2015, pp. 105–112. ISBN: 9781450334921. DOI: 10.1145/2750511.2750528. URL: https://doi.org/10.1145/2750511.2750528.

[9]   Stefanie Mika. "Challenges for nutrition recommender systems". In: *Proceedings of the 2nd Workshop on Context Aware Intel. Assistance, Berlin, Germany.* Citeseer. 2011, pp. 25–33.

[10] Chun-Yuen Teng, Yu-Ru Lin, and Lada A. Adamic. "Recipe Recommendation Using Ingredient Networks". In: *Proceedings of the 4th Annual ACM Web Science Conference.* WebSci '12. Evanston, Illinois: Association for Computing Machinery, 2012, pp. 298–307. ISBN:

9781450312288. DOI: 10.1145/2380718.2380757. URL: https://doi.org/10.1145/2380718.2380757.

[11]   Mouzhi Ge, Francesco Ricci, and David Massimo. "Health-Aware Food Recommender System". In: *Proceedings of the 9th ACM Conference on Recommender Systems*. RecSys '15. Vienna, Austria: Association for Computing Machinery, 2015, pp. 333–334. ISBN: 9781450336925. DOI: 10.1145/2792838.2796554. URL: https://doi.org/10.1145/2792838.2796554.

[12]   Douglas W Oard, Jinmook Kim, et al. "Implicit feedback for recommender systems". In: *Proceedings of the AAAI workshop on recommender systems*. Vol. 83. WoUongong. 1998.

[13]   Joseph A Konstan Michael D. Ekstrand. *Introduction to Recommender Systems: Non-Personalized and Content-Based*. Coursera.

[14]   Jürgen Buder and Christina Schwind. "Learning with personalized recommender systems: A psychological view". In: *Computers in Human Behavior* 28.1 (2012), pp. 207–216.

[15]   Amazon. *Amazon Order History*. Amazon. URL: https://www.amazon.com/gp/css/order-history?ref_=nav_AccountFlyout_orders.

[16]   FO Isinkaye, YO Folajimi, and BA Ojokoh. "Recommendation systems: Principles, methods and evaluation". In: *Egyptian Informatics Journal* 16.3 (2015), pp. 261–273.

[17]   Joseph A Konstan and John Riedl. "Recommender systems: from algorithms to user experience". In: *User modeling and user-adapted interaction* 22.1-2 (2012), pp. 101–123.

[18]   Shah Khusro, Zafar Ali, and Irfan Ullah. "Recommender systems: issues, challenges, and research opportunities". In: *Information Science and Applications (ICISA) 2016*. Springer, 2016, pp. 1179–1189.

[19]   Harry Zisopoulos et al. "Content-Based Recommendation Systems". In: (Nov. 2008).

[20]   Marwa Mohamed, Mohamed Khafagy, and Mohamed Ibrahim. "Recommender Systems Challenges and Solutions Survey". In: Feb. 2019. DOI: 10.1109/ITCE.2019.8646645.

[21]   Massimo Melucci. "Vector-Space Model". In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 3259–3263. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_918. URL: https://doi.org/10.1007/978-0-387-39940-9_918.

[22]   Juan Ramos et al. "Using tf-idf to determine word relevance in document queries". In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. Piscataway, NJ. 2003, pp. 133–142.

[23]   John T. Riedland Joseph A. Konstan Michael D. Ekstrand. *Collaborative Filtering Recommender Systems*. 2011.

[24]   Charu C Aggarwal et al. *Recommender systems*. Vol. 1. Springer, 2016.

[25] Manizheh Ranjbar et al. "An imputation-based matrix factorization method for improving accuracy of collaborative filtering systems". In: *Engineering Applications of Artificial Intelligence* 46 (2015), pp. 58–66.

[26] Thomas Hofmann. "Collaborative filtering via gaussian probabilistic latent semantic analysis". In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. 2003, pp. 259–266.

[27] Kai Yu et al. "Instance selection techniques for memory-based collaborative filtering". In: *Proceedings of the 2002 SIAM International Conference on Data Mining*. SIAM. 2002, pp. 59–74.

[28] B Sarwar et al. "Application of dimensionality reduction in recommender system-a case study: DTIC Document". In: *Technology Report* (2000).

[29] Badrul Sarwar et al. "Item-based collaborative filtering recommendation algorithms". In: *Proceedings of the 10th international conference on World Wide Web*. 2001, pp. 285–295.

[30] Gediminas Adomavicius and Jingjing Zhang. "Impact of data characteristics on recommender systems performance". In: *ACM Transactions on Management Information Systems (TMIS)* 3.1 (2012), pp. 1–17.

[31] David H Stern, Ralf Herbrich, and Thore Graepel. "Matchbox: large scale online bayesian recommendations". In: *Proceedings of the 18th international conference on World wide web*. 2009, pp. 111–120.

[32] J Ben Schafer et al. "Collaborative filtering recommender systems". In: *The adaptive web*. Springer, 2007, pp. 291–324.

[33] Robin Burke. "Hybrid recommender systems: Survey and experiments". In: *User modeling and user-adapted interaction* 12.4 (2002), pp. 331–370.

[34] Mark Claypool et al. "Combing content-based and collaborative filters in an online newspaper". In: (1999).

[35] Xiaoyuan Su and Taghi Khoshgoftaar. "A Survey of Collaborative Filtering Techniques". In: *Adv. Artificial Intellegence* 2009 (Oct. 2009). DOI: 10.1155/2009/421425.

[36] L. Sheugh and S. H. Alizadeh. "A note on pearson correlation coefficient as a metric of similarity in recommender system". In: *2015 AI Robotics (IRANOPEN)*. 2015, pp. 1–6.

[37] Jonathan L Herlocker et al. "Evaluating collaborative filtering recommender systems". In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pp. 5–53.

[38] Guy Shani and Asela Gunawardana. "Evaluating Recommendation Systems". In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Boston, MA: Springer US, 2011, pp. 257–297. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_8. URL: https://doi.org/10.1007/978-0-387-85820-3_8.

[39] Asela Gunawardana and Guy Shani. "A Survey of Accuracy Evaluation Metrics of Recommendation Tasks". In: *J. Mach. Learn. Res.* 10 (2009), pp. 2935–2962.

[40] Cyril W Cleverdon and Michael Keen. *Aslib Cranfield research project-Factors determining the performance of indexing systems; Volume 2, Test results*. Tech. rep. 1966.

[41] M. Jalili et al. "Evaluating Collaborative Filtering Recommender Algorithms: A Survey". In: *IEEE Access* 6 (2018), pp. 74003–74024.

[42] John S Breese, David Heckerman, and Carl Kadie. "Empirical analysis of predictive algorithms for collaborative filtering". In: *arXiv preprint arXiv:1301.7363* (2013).

[43] Nick Craswell. "Mean Reciprocal Rank". In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 1703–1703. ISBN: 978-0-387-39940-9. DOI: `10.1007/978-0-387-39940-9_488`. URL: `https://doi.org/10.1007/978-0-387-39940-9_488`.

[44] Kaggle. *Allrecipes Dataset*. Kaggle.

[45] University of Minnesota. *GLOSSARY OF COOKING TERMS*. University of Minnesota.

[46] U.S. Food and Drug Administration. *How to Understand and Use the Nutrition Facts Label*. U.S. Food and Drug Administration.

[47] Utah Deparment of Health. *How Many Calories Do You Need*. Utah Deparment of Health.