Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The internet is vast network of machines that connects 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 internet. One of the results has been the dramatic increase of data and a phenomenon known as BigData.

BigData referes to exponentially increasing data at a high volume, 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 insightful analysis that would recognize patterns to predict user behaviors to recommend products.

Handling BigData through manual processes is very inefficient. More efficient way of processing such huge amount of data is automating the process of classifying, filtering data of users opinions, features, and preferences in order to understand and predict new set of related products. Recommender systems are tools that filters information and narrow it down based on user's preferences and helps user to choose which he/she may like. They considers the opinions of communities of users to help each individual in order to understand 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 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 World Health Organization (WHO) [4], globally 39% of the adults were over-weighted and 13% were obese in 2016. Overweight

and obesity can cause diabetes, blood pressure, heart disease and many other chronic diseases [5]. A well balanced diet plays very important role to improve overall health. With rapid changes in our busy lifestyle, people find it difficult to choose healthy eating option [6]. People tend to move towards options which satisfies their taste or an easy by ignore 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 narrowing down the information considering our health and eating history. This thesis further discusses the different approaches for recipe domain to recommend healthy recipes based on user's profile.

TO DO Chapterwise breakdown of thesis. Template to write down breakdown.

The contributions of this paper are the following. First, we provide a short overview of recommendation approaches for individuals. Second, we discuss group decision making issues which have an impact on the development of group recommendation technologies. Third, on the basis of categorizing food recommender systems, we analyze how well those systems can help individuals or groups to choose healthy food which best fits their preferences and health situations. Finally, we point out some challenges of food recommender systems with regard to user information, recommendation algorithms, changing eating behaviors, explanations provision, and group decision making as

topics for future work.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of basic recommendation techniques for individuals and groups. In Section 3, we summarize existing studies on food recommender systems for single users and categorize them according to different criteria, such as preferences, nutritional needs, health problems, and eating behaviors of users. Besides, in this section we also discuss some research related to food recommender systems in group scenarios. Research challenges for food recommender systems are discussed in Section 4. The paper is concluded with Section 5.

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 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 are challenging. Such challenge considers changing user's mind 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 systems to hy-

brid approaches.

According to Mika [7], there are two types of food recommender systems present. First type considers user's preferences. It recommends recipes which are similar to the recipes liked by user in the past. 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.

The research of Freyne and Berkovsky [6] uses content based technique to predict rating of unrated recipe based on the corresponding ingredients included in the recipe. For a user, the framework breaks down unrated recipe into ingredients then calculates rating of each ingredient based on rating of all other rated recipes which contain same ingredients. After calculating ratings for all ingredients of unseen recipe, framework predicts the rating of a user for the unseen recipe based on 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 weighting factor for ingredients [8]. This framework take a different approach by using two matrices for user features. One of them is weighted positively for rating value 4 and 5. Weighting factors for rating value 5 and

4 were 2 and 1 respectively. Contrary, another matrix weighted negatively for rating value 1 and 2. Weighting factors for rating value 2 and 1 were 1 and 2 respectively. These matrices implies strong likes and dislikes between ingredients. Based on ingredients similarity score recipes were recommended.

Recent work experiment shows that how can one bring 'healthy' factor in recommendations by substituting ingredients [9].

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. But performance of content-based approach was better. In [8] a recipe recommender implemented using Singular Value Decomposition (SVD) which outperformed content-based and collaborative approach experimented by Freyne and Berkovsky.

Recent developments in recipe recommender systems consider using matrix factorization. Elahi propose a food recommeder system using active learning algorithm and matrix factorization [10] 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 approches performs well in predicting what user may like according to user's

taste, but it did not help in changing a user's behavior towards helathy 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 food domain and varying recommender approach to recommend a healthy recipes by incorporating calorie count and look to extend the work by [10, 11].

2.2 Data Collection and Filtering

2.2.1 BigData

The World Wide Web Consortium (W3C), formed in 1994 []. It has developed standards for communication between computers such as HyperText Transfer Protocol (HTTP), HyperText Markup Language (HTML), Uniform Resource Locator (URL). W3C standards improve the efficiency of communication. It has led to large amounts of information sharing between the users and the hosts on the web. Recently, the ubiquitous availability of internet connections along with mobile technologies has lead to Big Data. It can be described as a large volume of complex data [1]. Volume refers to the amount of data generated in a unit of time. Velocity refers to the speed at which data is generated. Variety refers to the type of data that is generated. For example, as shown in Figure 2.1 we can see and compare that, data generated from mainframe source was less and structured than today's data generated by different channels such as mobile messaging, social media and internet.

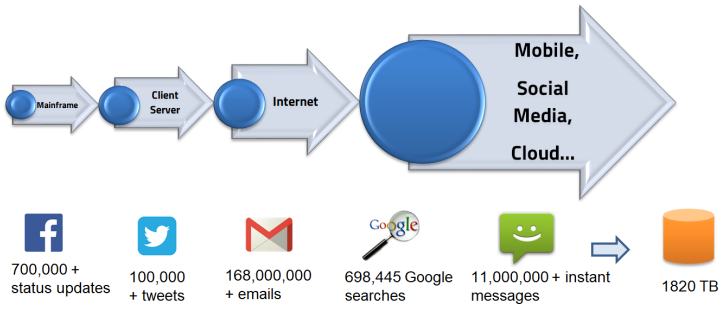


Figure 2.1: Growing Data Velocity per second

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 the foundation of accuracy of recommendation engine. It is helpful to generate user profiles or models for making recommendations. To well construct user profile, recommendation engine rely 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 with user interaction with the system [12].

Explicit Data

In explicit data application prompts the user through user interface to provide feedback about the product in order to understand user's preferences to improve model. The user then interacts with 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 user to different movies.

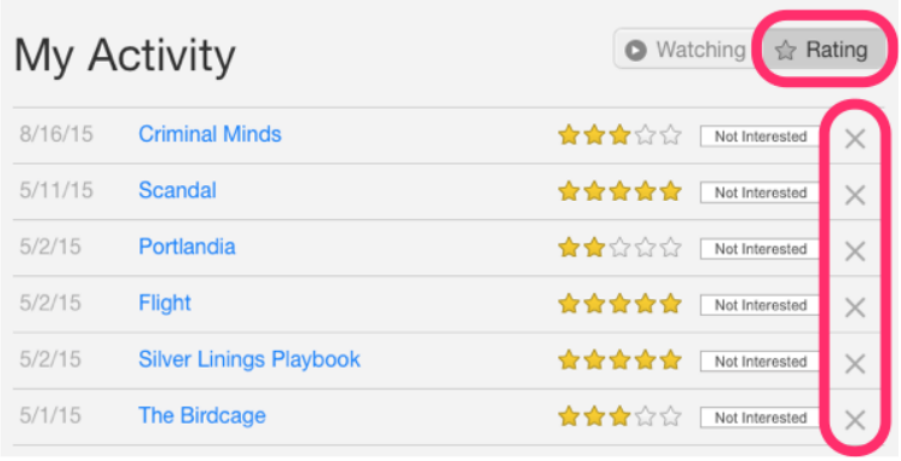


Figure 2.2: Explicit Data [13]

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

Implicit Data

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

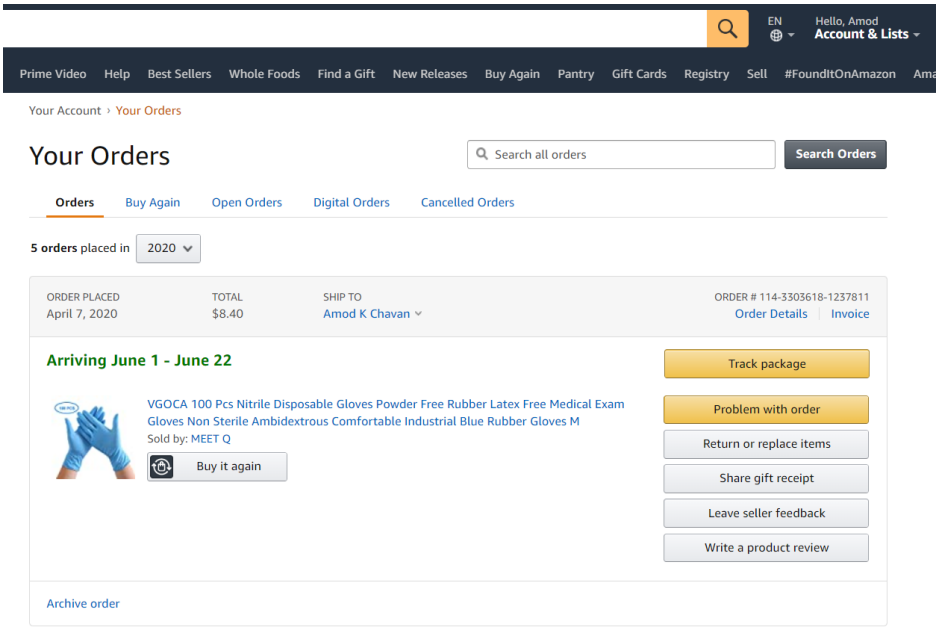


Figure 2.3: Implicit Data

Based on a user's prior purchase, Amazon is recommending other products as shown in screen-shot Figure 2.4.

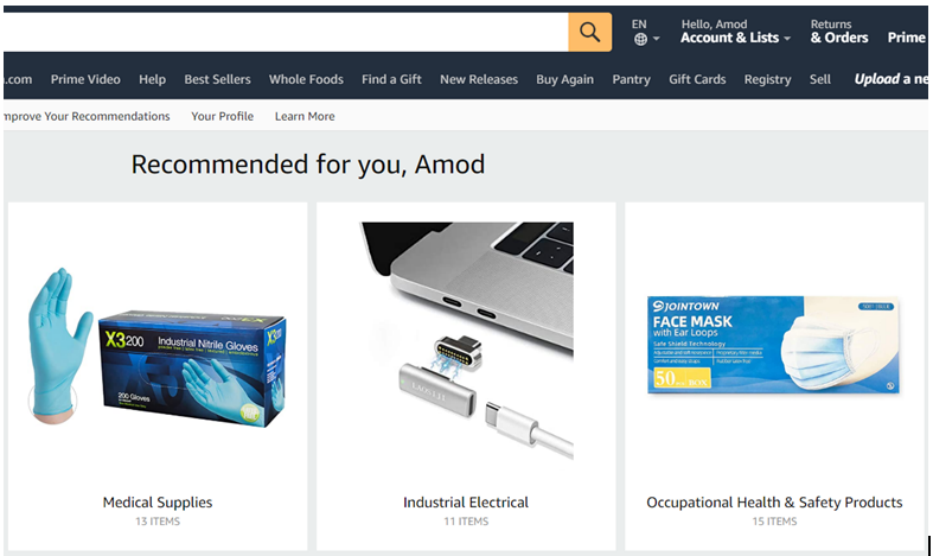


Figure 2.4: Recommendations based on Implicit Data

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 online recommendation system. The work in 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 queryprocessing. A crawler

processes all URLs via techniques like breadth-first search and depth-first search and stores the web servers 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 retrieval and processing of extracted data is quicker.

2.2.3 Information Filtering

Information Filtering is associated with 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 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 helps 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.

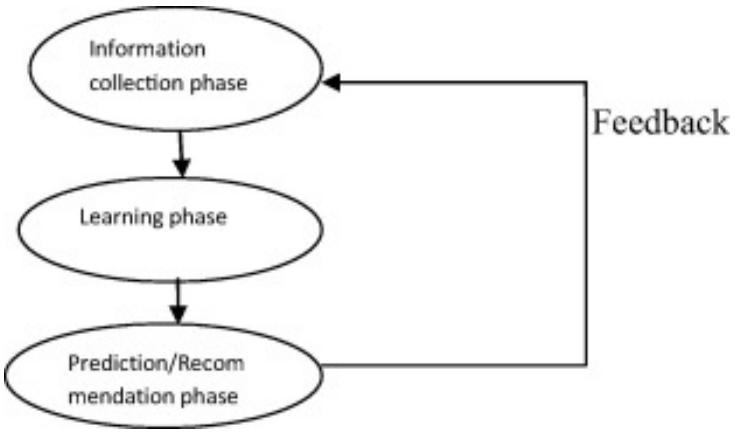


Figure 2.5: Recommendation Phases [15]

2.3 Recommender System

Recommender systems are information filtering systems that provide a solution for the problem of information overload [16]. 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. User Modeling in the figure represents the user's preferences. 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.

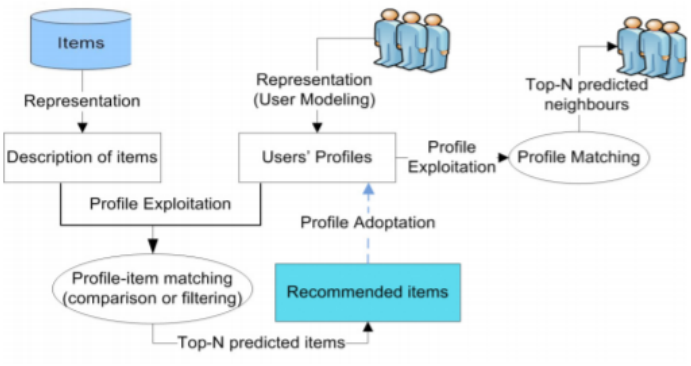


Figure 2.6: General Recommender Model [17]

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

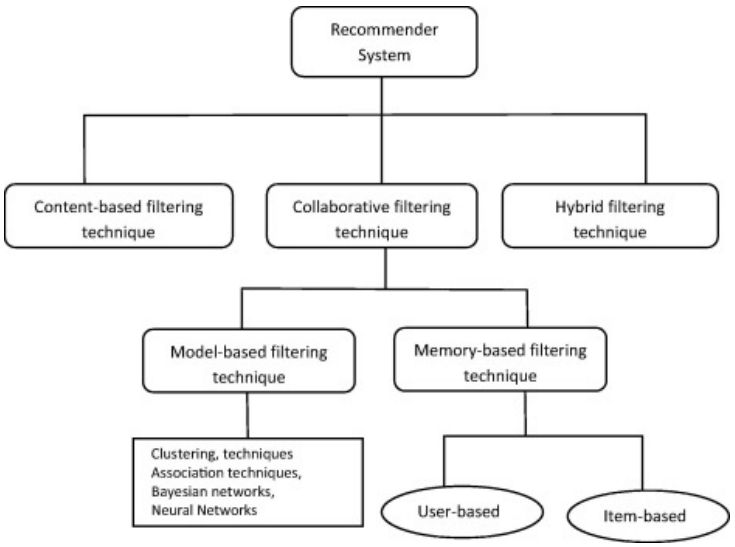


Figure 2.7: Recommendation Techniques [15]

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

- Content Based Filtering
- Collaborative Filtering
- Hybrid Filtering

In next section we will discuss commonly used methods.

2.3.1 Content Based Filtering

In Content based method algorithm, user preference is considered based on 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 reflects a user's taste [18].

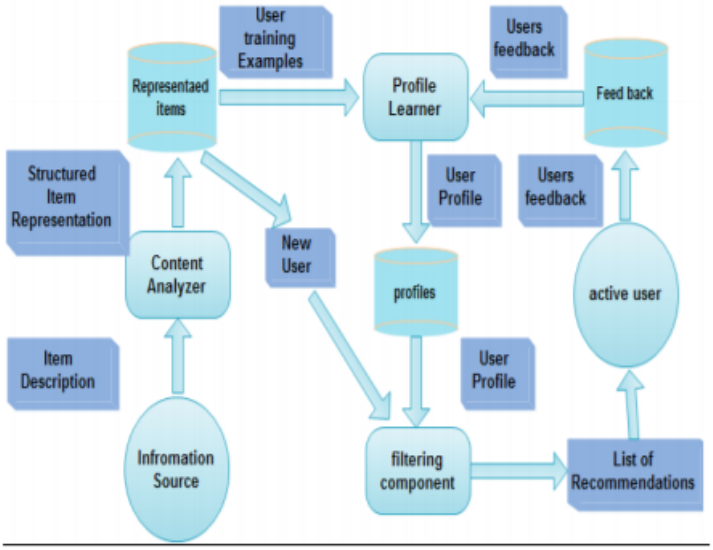


Figure 2.8: Content Based filtering Architecture [19]

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

- **Content Analyzer**

  When information has no exact structure like a document that time

pre-processing step is necessary to extract relevant features from that document. Content analyzer analyzes such items 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**

  Filtering component uses user's profile to find matching items from items data set. The items are matching with a user's data profile is

decided by calculating similarity between a user's data profile and item vectors. A list of potential interesting items is recommended by filtering component.

In content based algorithm each user's information can be stored in vector form which contains past behavior of the user. This vector is known as profile vector or user profile. All the information about item is stored in item vector or item profile which contains all the details about item specific attributes. Based on similarity score between user profile and item profile most relevant items are recommended to user. The calculation of user or item vector is discussed in subsubsection 2.3.1.

Vector Space Model

Vector Space Model (VSM) for information retrieval represents documents as queries as vectors of weight [20]. It is also known as term vector model as it uses term occurrences as vector identifier.

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

	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

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.

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 binary representation of rating considers all terms with same weight. To consider only important terms, various components are combined to calculate weight of the term. One way to calculate weight of term is to user term frequencies and inverse document frequencies (TF-IDF).

Machine Learning	Databases	Cyber Security
1	1	1

Table 2.3: User Vector for Book's Genres

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 respect to document corpus. TF-IDF weighting negates the effect of high-frequency words in order to understand the importance of the term [21]. 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 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

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

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

$$TF - IDF = TF * IDF \tag{2.3}$$

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' $= 0.1 * 1.60 = 0.160$

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 [18].

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.

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 user 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 of the system decreases [18].

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 metadata. 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 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 [22].

The main source of input for this 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 3. The system finds similar user profiles in different ways but here we can see that user 4 likes the same items which are rated high by user 1. From this analysis, we can consider that user 1 and user 4 are quite similar. Based on this matrix 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 system can predict that user 4 may like item 4 because item 4 is rated high by user 1. 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.

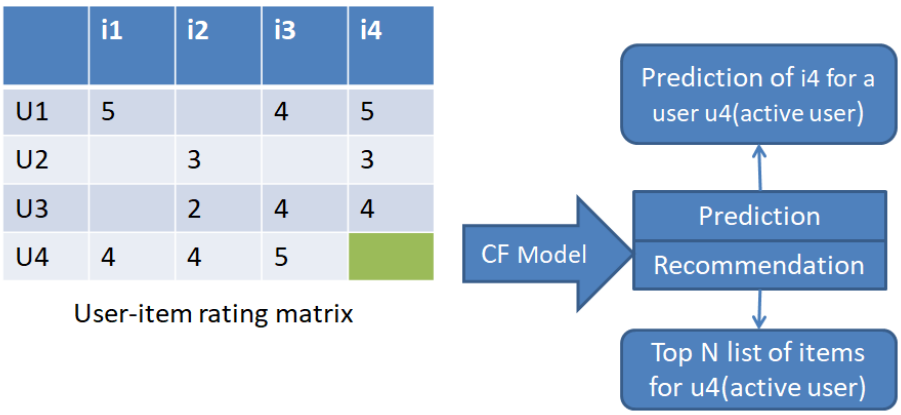


Figure 2.9: Collaborative Filtering Process [15]

Collaborative Filtering technique is broadly divided into 2 categories [23].

- Memory-based CF

  A memory-based collaborative filtering approach predicts item ratings based on ratings given by different users for an item. There are primary two forms of memory-based collaborative filtering.

- 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 [24, 25, 26].

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

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

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 the section of 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.

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 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.

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 [27]. 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

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

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 and eliminate lower entries. So, $(r-k)$ columns from U and $(r-k)$ rows from V^T are discarded to gnerate 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

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

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

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

With SVD we can predict ratings with good accuracy.

Advantages and Disadvantages of collaborative filtering techniques

Collaborative Filtering has major advantages over content-based in domains where there is selection of many features is not necessary. Collaborative filtering has ability to recommend a new item such that even if content of that item is not stored as user's preference in user's profile. It is very successful algorithm but it suffers from some drawbacks [28].

- Data Sparsity Problem

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

- Cold Start Problem

  Cold-start is a situation where system dose 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 user has not rated any item so, the preference of user will be unknown to the system.

- 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

Pure recommendation systems have some limitations such as cold start problem, data sparsity as discussed in Advantages and Disadvantages of collaborative filtering techniques. Hybrid filtering techniques use combination of different recommendation techniques to overcome limitations of pure recommendation techniques to improve performance [29, 30]. Many researchers have combined content-based and collaborative filtering techniques to gain better results. The idea behind combining different recommendation techniques is that resultant algorithm will provide more accurate and effective recommendations than any single algorithm [31]. Burk [32] has categorized hybrid techniques in different types. One of them is weighted hybridization.

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, [33] P-Tango system combines content-based and collaborative filtering techniques. At start, both techniques were equally weighted but based on performance and user ratings the weight for different techniques has gradually adjusted. In this paper, we will be using weighted hybrid technique.

2.4 Similarity Methods

There are several methods available to calculate similarity score.

2.4.1 Cosine Similarity

In cosine similarity measure [34], the result is the cosine of the angle between two vectors. It is measured by the cosine angle between two vectors and discovers the direction between two vectors is same or not. Often it 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, the similarity between them can be calculated as per Equation 2.9.

$$sim(A, B) = cos(\theta) = \frac{A.B}{\parallel A \parallel \parallel B \parallel} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.9}$$

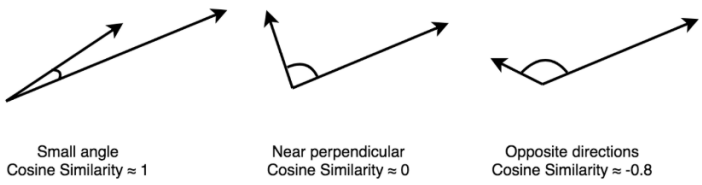


Figure 2.10: Cosine Similarity

Using subsection 2.4.1 if we compute cosine similarity between A and B vectors, then we get:

$$\| A \| \| B \| = (5x3 + 0x0 + 3x2 + 0x0 + 2x2) = 25$$

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

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

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

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 the angle, less distance between vectors hence vectors considered as more similar to each other.

2.4.2 Euclidean Distance Similarity

The Euclidean distance between two points is the length which is connecting those two points. If we plot n dimensional space and plot similar items,

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 [23]

then they will fall under close proximity. Consider a example of a positive quadrant of space and we plot items on the axis which are rated by user. The Table 2.4 illustrates the relationship between users and movies with ratings. The points drawn on graph in Figure 2.11 represents rating given by the user to those particular movies.

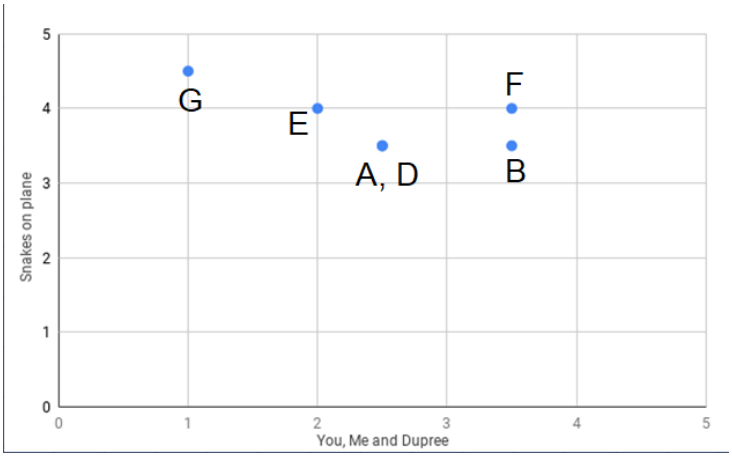


Figure 2.11: Movie Rating Plots [23]

In that case, we can calculate distance between items with Euclidean distance formula which is given by Equation 2.10

$$\text{Euclidean Distance } = \sqrt{(x_1 - y_1)^2 + ... + (x_n - y_n)^2} \tag{2.10}$$

From above example, if we calculate Euclidean's distance between user A and user B then we get :

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

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 [35]. Correlation on higher side implies more similarity. Equation 2.11 gives correlation between two users r_u and r_v.

$$sim(u, v) = \frac{\sum(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{(\sum(r_{ui} - \bar{r}_u))^2}\sqrt{(\sum(r_{vi} - \bar{r}_v)^2}} \tag{2.11}$$

Where r_{ui} and r_{vi} are rating scores for from two users. $\bar{r_u}$ 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 that no correlation. Hence a correlation value can capture a rating similarity between two users.

	Recommended	Not Recommended
Relevant	TP	FN
Irrelevant	FP	TN

Table 2.5: Confusion Matrix [40]

2.5 Evaluation Metrics for Recommendation Systems

In this section evaluation metrics used to evaluate recommender systems algorithms are explored. Evaluation can be done by two ways, offline and online evaluation [36, 37]. In offline evaluation methods an analysis of collected data is divided into training sets and testing sets in the proportion of 80% training to 20% testing. The model of recommender system is trained on the training dataset and testing dataset is hidden from engine. To understand the quality of recommendation engine, one or combination of evaluation metrics are used. There are several methods available to evaluate the performance of the recommender systems [36, 38].

2.5.1 Recall and Precision

Recall and precision are most commonly used metrics to evaluate recommendation engines [39]. These metrics can be explained by a confusion matrix [40] as shown in Table 2.5.

Where

TP denotes True Positive represents all relevant items are recommended by the system.

TN denotes True Negative represents all irrelevant items are correctly not recommended by the system.

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

FN denotes False Negative represents relevant items but 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 2.12.

$$Precision = \frac{TP}{TP + FP} \tag{2.12}$$

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 2.13.

$$Recall = \frac{TP}{TP + FN} \tag{2.13}$$

Larger value of recall and precision implies better recommendations.

2.5.2 Mean Absolute Error (MAE)

Mean absolute error used to calculate the average deviation or error generated from predicted ratings and actual ratings [41].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Predicted_i - Actual_i| \tag{2.14}$$

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 2.15.

$$NMAE = \frac{MAE}{Rating_{max} - Rating_{min}} \tag{2.15}$$

2.5.3 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 2.16. RMSE can be defined as a square root of the average of squared deviations

between predicted ratings and real ratings.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Predicted_i - Actual_i)^2} \tag{2.16}$$

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.

Normalized RMSE is given in Equation 2.17

$$NRMSE = \frac{RMSE}{Rating_{max} - Rating_{min}} \tag{2.17}$$

2.5.4 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 [42]. 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 2.18

$$MRR(O,U) = \frac{1}{|U|}\sum_{u\in\epsilon}\frac{1}{k_u} \tag{2.18}$$

Where,

k_u denotes the first relevant recommended item for user u.

U denotes the number of users.

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

2.5.5 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. Mathematically it is show in Equation 2.19

$$MAP_i = \frac{1}{|R_i|} \sum_{k=1}^{|R_i|} P(R_i[k]) \tag{2.19}$$

Where,

R_i denotes rating for item i.

k denotes rank from 1 to k.

P denotes precision of first relevant item.

Higher mean precision value implies correct recommendations.

Chapter 3

Implementation

3.1 Environment Setup

In this thesis, different techniques of recommendation algorithms have 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 performance in terms of time and space 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 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.

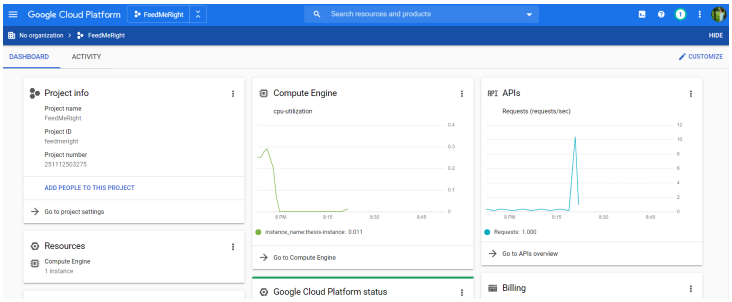


Figure 3.1: GCP Dashboard

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.

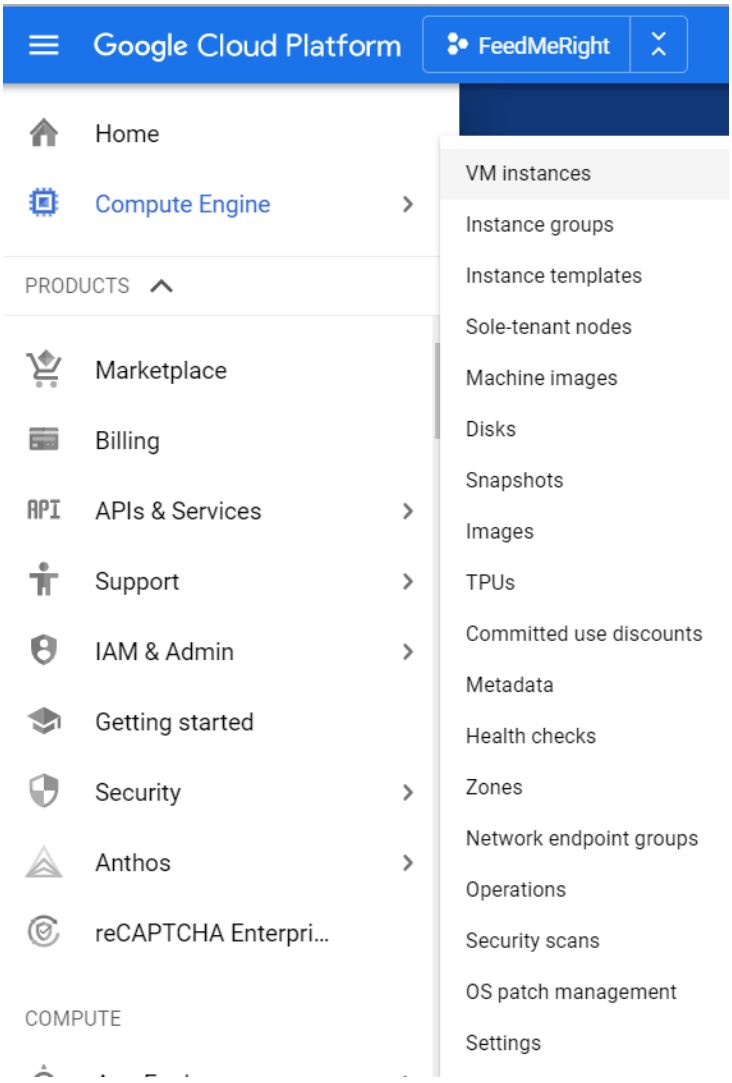


Figure 3.2: GCP Service List

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.

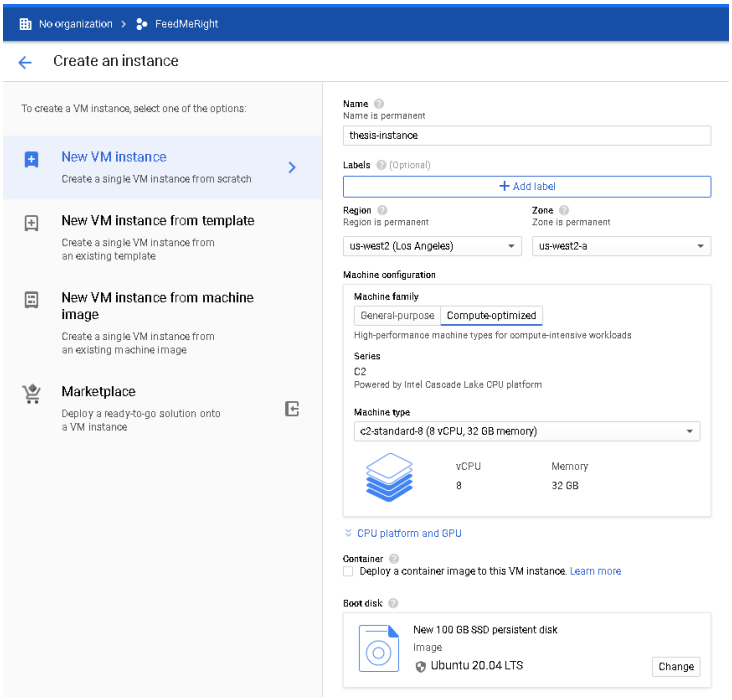


Figure 3.3: VM Configuration for FeedMeRight

3.1.3 Python Scripts Deployment

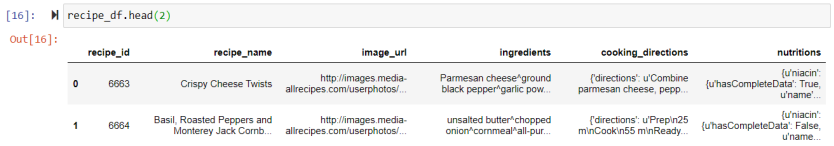
After successful creation of VM instance, the application can be configured on this instance. To get all scripts created for 'FeedMeRight' Git software has been used. Git is a version control system to keep track of changes in the source code. The source code for the project is stored on GitHub. GitHub 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 [43]. This dataset 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 as per Figure 3.5.

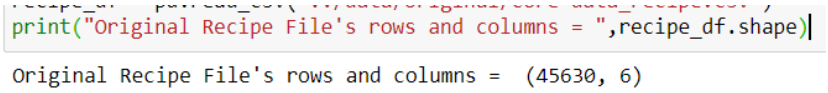


```
[16]: recipe_df.head(2)
```

Out[16]:

	recipe_id	recipe_name	image_url	ingredients	cooking_directions	nutritions
0	6663	Crispy Cheese Twists	http://images.media-allrecipes.com/userphotos/...	Parmesan cheese^ground black pepper^garlic pow...	{'directions': u'Combine parmesan cheese, pepp...	{u'niacin': {u'hasCompleteData': True, u'name':...
1	6664	Basil, Roasted Peppers and Monterey Jack Cornb...	http://images.media-allrecipes.com/userphotos/...	unsalted butter^chopped onion^cornmeal^all-pur...	{'directions': u'Prep\n25 m\nCook\n55 m\nReady...	{u'niacin': {u'hasCompleteData': False, u'name...

Figure 3.4: Raw Recipe Data



```
print("Original Recipe File's rows and columns = ",recipe_df.shape)
```

```
Original Recipe File's rows and columns =  (45630, 6)
```

Figure 3.5: Rows and Columns of original recipe's file

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 such as a user has given a rating to a recipe as depicted in Figure 3.6. These two files are combined and formed a single file that has all user interactions. It has 960386 users-recipes interactions and 4 columns, 'user_id', 'recipe_id', 'rating', 'dateLastModified' as shown in Figure 3.7.

```
In [17]: core_ratings_df.head(2)
```

Out[17]:

	user_id	recipe_id	rating	dateLastModified
0	5215572	17991	5	2010-08-25T14:38:53.84\n
1	5215572	170724	4	2010-09-09T14:04:45.733\n

Figure 3.6: Raw User-Interaction Data

```
print("Original ratings file's rows and columns = ",train_test_ratings_df.shape)
Original ratings file's rows and columns =  (960386, 4)
```

Figure 3.7: Rows and Columns of original user-recipes interactions

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', 'Cooking Method', 'Calories' and 'Diet Labels' features to find similarities between recipes.

Ingredients Extraction

The recipe's data received from Kaggle has the ingredients column in a clean format at 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 words that are irrelevant in predicting similarity be-

tween ingredients such as 'white' from 'white eggs', 'frozen', 'thawed', 'piece'. Such custom keywords are added in a recipe_stopwords list and removed them from ingredients to make ingredients very specific. The difference between raw ingredients and cleaned ingredients is illustrated in Figure 3.8 Figure 3.9. Irrelevant words such as 'thawed' and 'white' present in the first column of the pre-processed ingredients column have been removed in post-processed ingredients.

```
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 3.8: 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 3.9: Post-processed Ingredients

Bibliography

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

[2] Paul Resnick and Guest Editors Hal R. Varian. "RecommenderSystems". In: (1997).

[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 International 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] 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.

[8] 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.

[9] 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.

[10] 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.

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

[16] 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.

[17] Ullah I. Khusro S. Ali Z. "(2016) Recommender Systems: Issues, Challenges, and Research Opportunities". In: (2016).

[18] Harry Zisopoulos et al. “Content-Based Recommendation Systems”. In: (Nov. 2008).

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

[20] 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.

[21] 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.

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

[23] Aggarwal C.C. “An Introduction to Recommender Systems. In: Recommender Systems”. In: (2016).

[24] 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.

[25] 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.

[26] 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.

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

[28] Joseph Konstan Badrul Sarwar George Karypis and John Riedl. "Item-Based Collaborative Filtering RecommendationAlgorithms". In: (2001).

[29] 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.

[30] 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.

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

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

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

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

[35] 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.

[36] LOREN G. TERVEEN JONATHAN L. HERLOCKER JOSEPH A. KONSTAN and JOHN T. RIEDL. “Evaluating Collaborative FilteringRecommender Systems”. In: (2004).

[37] 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.

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

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

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

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

[42] 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.

[43] Kaggle. *Allrecipes Dataset*. Kaggle.