RECOMMENDING RECIPES BASED ON INGREDIENTS AND USER REVIEWS

by

ANIRUDH JAGITHYALA

B.Tech, Jawaharlal Nehru Technology University (JNTU), India, 2010

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Approved by:

Major Professor
Doina Caragea

Abstract

In recent years, the content volume and number of users of the Web have increased dramatically. This large amount of data has caused an information overload problem, which hinders the ability of a user to find the relevant data at the right time.

Therefore, the primary task of recommendation systems is to analyze data in order to offer users suggestions for similar data. Recommendations which use the core content are known as content-based recommendation or content filtering, and recommendations which utilize directly the user feedback are known as collaborative filtering.

This thesis presents the design, implementation, testing, and evaluation of a recommender system within the recipe domain, where various approaches for producing recommendations are utilized. More specifically, this thesis discusses approaches derived from basic recommendation algorithms, but customized to take advantage of specific data available in the *recipe* domain. The proposed approaches for recommending recipes make use of recipe ingredients and reviews. We first build ingredient vectors for both recipes and users (based on recipes they have rated highly), and recommend new recipes to users based on the similarity between user and recipe ingredient vectors. Similarly, we build recipe and user vectors based on recipe review text, and recommend new recipes based on the similarity between user and recipe review vectors. At last, we study a hybrid approach, where both ingredients and reviews are used together. Our proposed approaches are tested over an existing dataset crawled from recipes.com. Experimental results show that the recipe ingredients are more informative than the review text for making recommendations. Furthermore, when using ingredients and reviews together, the results are better than using just the reviews, but worse than using just the ingredients, suggesting that to make use of reviews, the review vocabulary needs better filtering.

Table of Contents

List of Figures

List of Tables

Acknowledgments

While this thesis is my own work, it benefited from the insights and directions of several people. I would like to thank them all for their help and support.

First and foremost, I would like to express my deepest gratitude to my advisor Dr. Doina Caragea, for her excellent guidance, caring, patience, and for providing me with an excellent atmosphere for doing research. Without her guidance and persistent help this thesis would not have been possible.

I would also like to thank Dr. Daniel Andresen, for being a member of my M.S. committee. His classes are a great source of information and also motivate students to think of latest and efficient solutions to the challenging problems.

I would like to thank Dr. Torben Amtoft for being a member of my M.S. committee, and for educating me with some of the important concepts of algorithms and about ways to tackle some of the hardest problems.

I wish to place on record my deep sense of gratitude to my Supervisor, Mr. Will Baldwin from Biosecurity Research Institute, with whom I worked throughout my Masters years at Kansas State University.

Finally, I would like to thank my family members, Mr. Jagityala Ashok Kumar, Mrs. Jagithyala Arpitha, my brother Mr. Animesh Jagithyala and my sister Mrs. Likitha Jagithyala, for their love and support at every stage of my life. I would also like thank my friends and colleagues for their support throughout my Graduate Study.

Chapter 1

Introduction

Recommender systems are prominently researched within the field of Data processing and Data Mining. Recommender systems are applied in e-commerce websites, social networking sites, and many other domains which contain high volume of data. The primary goal of recommender systems is to analyze data behavior and predict similar future relativity. Section 1.1 offers motivation for the proposal of recommender system. We state the problem addressed and emphasize on challenges of the recipe domain in order to recommend recipes in Section 1.2. The next Section 1.3 describes my contribution towards this thesis. Finally, an outline of proposed approaches is described in Section 1.4.

1.1 Motivation for Recommender Systems

The World Wide Web, a massive database system in which connected information can be accessed or manipulated through hypertext, is experiencing dramatic growth as a result of increased Internet usage. Recent statistics also indicate that the number of Internet users is high and rapidly growing. Data continues to increase with more interrelated features such as products, corresponding reviews, new products, and user preferences. Most available information help providers recommend similar available products with closely related features.

However, manually processing existing data is tedious, inefficient, and often leads to errors. In addition, it is difficult to classify, filter, and then recommend from such a huge set of data. A more efficient approach is to automatically process user opinions, features, and other related data in order to predict a new set of related products.

Recommender systems achieve this goal(i.e., to suggest products based on processing of related opinionated products) by utilizing opinions of a community of users to help individuals in that community to effectively identify content of interest from a potentially overwhelming set of choices[1] cited at p. 1, 8. Two dominant recommendation strategies: content-based and collaborative filtering. Content-based filtering relies on rich content descriptions of items that are recommended[2] cited at p. 1, while collaborative filtering recommendations are motivated by the fact that users often look to friends for recommendations[3] cited at p. 1, 14, 20.

Recommender systems are primarily applied for commercial use to analyze the process of data. Especially for popular e-commerce sites such as Amazon. Any approaches such as building ingredient networks and exploring Folksonomy and Cooking Procedures exist on recipe domain. This paper would further discuss the different approaches for a recipe domain to recommend additional accurate recipes.

1.2 Problem Addressed and Challenges

This thesis focuses on the development and evaluation of a recommender system within the recipe domain. Various approaches for computing recommendations are designed, implemented, and tested with real end-users. Evaluation was conducted by assessing system functionality and comparing recommender precision obtained by each approach.

One of the many factors for any successful user interactive systems is the ability to offer accurate recommendations. For the domain of recipe databases, recommendation of various recipes is difficult due to variations in user preferences, ingredients, cooking procedures, etc. Recipes may contain identical titles but they may differ in details of preparation procedures and ingredients.

In addition, standard strategies do not always sufficiently reflect a user's preference because preference is often context-dependent[4] cited at p. 2. User mood, allergic ingredients, or preparation procedures with/without certain ingredients or predefined preparation procedures (such as baked, fried, steamed) could contribute to user preferences. By integrating recipe ingredients, preference with recommender systems a more accurate recommender than a standard collaborative filtering.

This paper presents various approaches for building recommendations based on user preferences, and corresponding ingredients. Moreover the free-form review text is also used to identify recipe similarities for recommendations.

1.3 Thesis Contribution

The major contribution of this thesis work is to propose approaches to recommend recipes based on recipe ingredients and user reviews. All approaches mentioned in chapter 3 have been devised and the corresponding algorithms were implemented and tested.

Each approach is evaluated and compared against a standard list of recommendations generated with a generic collaborative filtering approach. The collaborative filtering approach is considered as a benchmark for all the approaches.

1.4 High-Level Overview of Proposed Approaches

The primary objective of this project was to propose various approaches that could be used to predict user recommendations on a recipe domain. User ratings are typically the only factor utilized in order to make recommendations to the user. However, other approaches specifically related to recipe domain are proposed below:

1. **Approach 1: Ingredient-based Similarity of Recipes**

   In this approach, the system is trained to consider ingredients as primary factor for recipes to be similar. This approach does not consider user profiles. The current recipe is considered to be liked by the user and recommendations are provided to an anonymous user without a profile. Most similar recipes are recommended to the user browsing/reviewing/liking the current recipe.

2. **Approach 2: User Recommendations based on Recipe Ratings**

   This approach utilizes user ratings given for various user recipes. Each user offers a rating after reviewing the recipes. Based on these ratings, similar recipes which may be related to the user are predicted. This approach is an extension to collaborative filtering.

3. **Approach 3: User Recommendations based on Recipe Ingredients**

   The main criteria for recommendations is recipe ingredients, there by demonstrating content based filtering. Recipe ingredients are considered to correspondingly determine relative recipe recommendations.

4. **Approach 4: User Recommendations based on Recipe Review Text**

   Recommendations to a user are generated based on textual information reviews offered by the user, the user profile is constructed based on reviews given by user. Textual reviews of the recipes are thus analyzed to predict user recommendations. This approach is also an extension to content based filtering.

5. **Approach 5:User Recommendations based on Recipe Ingredients and Review Text - HYBRID**

   Recommendations to a user are based on recipe ingredients and corresponding textual reviews of the recipes. The hybrid approach considers content-based filtering on recipe ingredients and review text given by users. The hybrid approach also combines the similarity measure obtained from content based filtering on ingredient and content based filtering on review text. User recommendations are generated most closes to the similarity measures.

The remainder of the thesis is organized as follows: Chapter 2 describes related work on recommender systems. Chapter 3 formulates the problem of recommender systems on the recipe domain and explains various approaches and detailed examples. Chapter 4 explains the dataset, experimental setup of various performed experiments, and the research questions addressed. Chapter 5 discusses experimental results and explains the usefulness of the proposed approaches for recommender system approaches on recipe domain. Chapter 6 concludes the work and the directions for future work are presented in Chapter 7.

Chapter 2

Related Work

This chapter introduces the concept of recommender systems and problems which recommender systems are attempting to solve by utilizing various approaches. A set of widely used basic approaches of recommender systems are also described.

2.1 The World Wide Web

The World Wide Web consortium, formed in 1994, developed standards within which computers can communicate with each other. These standards included the use of Hyper Text Transfer Protocol (HTTP), Hyper Text Markup Language (HTML), and Uniform Resource Locator(URL) in order to communicate efficiently. Since 1994, these standards have provided a simple and standard platform through which information is shared. These standards increased the number of users and hosts which share data over the Web. The number of hosts have been exponentially increasing since 1994 and is expected to continue as described in figure 2.1.

Since the discovery of world wide web in the 90's, information or the required data has been growing rapidly. In 1990, it has already been accentuated by Tim Berners-Lee, the need for an information management system, to prevent the loss of information resulting

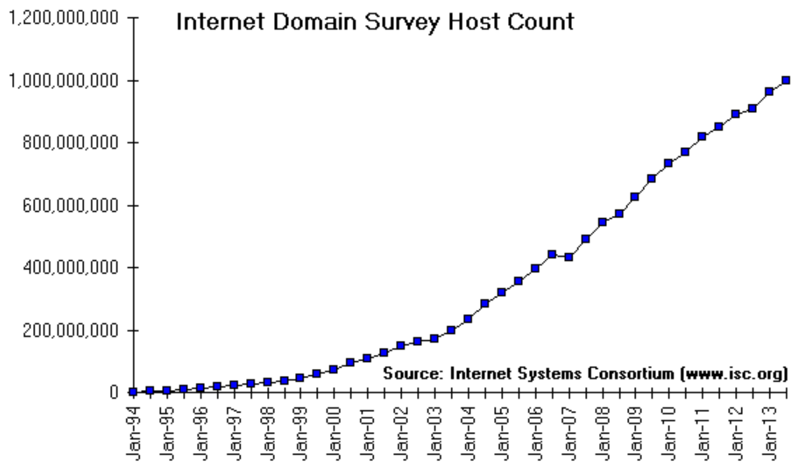


Figure 2.1: *Number of Internet Hosts*

from the growing organizational structure at CERN1[5].

Rapid expansion of web size and amount of information required application various techniques in order to find required information. These techniques are categorized as *information filtering* and *information retrieval.* Although the goal of information retrieval and information filtering is to deal with the information overload problem by examining and filtering big amounts of data, a distinction is often made between the two[6].

2.2 Information Retrieval

Information Retrieval(IR), commonly associated with data search or searching required information, involves technologies such as *web crawling, document processing, indexing and query processing..* A high level architecture of the information retrieval system is shown in 2.2. Crawling is the retrieval of various kinds of information from many diverse web servers. A web crawler processes the entire set of *URL* or links to request web servers and stores response as information. The crawler then processes the internal URL on the retrieved data using approaches such as depth-first search and breadth-first search. Then the documents are processed in order to add information like, meta data details, or remove

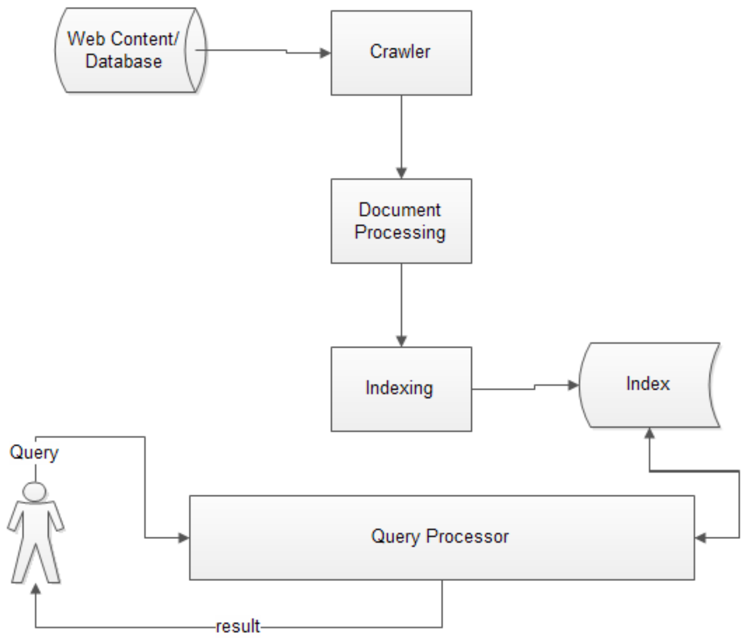


Figure 2.2: *Information Retrieval System Architecture*

noisy data.

Furthermore, nearly all information retrieval systems construct indexes from processed data. Data indexing allows fast processing and easy retrieval of the extracted data.

A request for information is a query. A typical IR system allows the user to write queries in order to retrieve the related information. The IR system interacts with the query processor to retrieve the query in the form of keywords on the indexed data. The objective is to evaluate the user's intent from the query, and then retrieve the most relevant documents.

IR depends on the users to type in the query for the required data. An IR system is efficient for the users who can query the system based on certain keywords obtained from the user's knowledge of the complete dataset.

2.3 Information Filtering

Information Filtering emphasize filtering of information specific to a user based on user preferences or user profiles. User behavior is studied by utilizing the user input or monitoring user activity. Information Filtering (IF) is automatically performed by the system to provide the user with information related to the profile. A detailed description is shown in the figure 2.3

The primary advantage of information filtering is its ability to adapt to various user profiles and to automatically perform the action by the user based on the past user profile. Information Filtering (IF) does not require the user to type in the query, but it records all user activity and filters the data to provide a suitable suggestion for future actions.

A subclass of an IF system that seeks to predict the 'rating' or 'preference' a user would give to an item is known as a recommender system.[7].

2.4 Recommender Systems

A recommender system is an Information Filtering (IF) system that provides personal preference guide based on the user profile and preferences.

2.4.1 Content-based Filtering

Information filtering (IF) differs from Information Retrieval (IR) in the way that user interests are presented. Instead of allowing user lookup information using a query, an IF system attempts to model the user's long-term interests and suggest relevant information to the user.

Content-based filtering methods, based on item description, considers user preferences according to the user profile. A content-based algorithm stores the users preferences such as interests to provide recommendations.

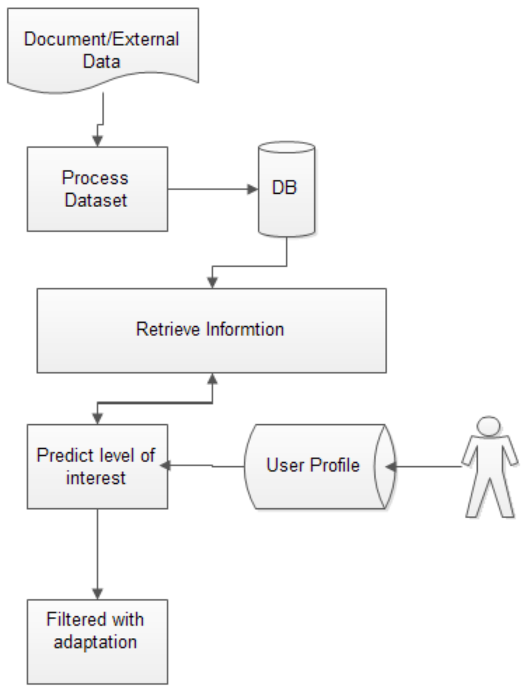


Figure 2.3: *Information Filtering in Recommender Systems*

Content-based filtering considers user history in order to match the history to the predicted future interest of the user for recommendations. Based on the algorithm considered, user preferences can be represented by weighted vectors and then compared to completed document dataset in order to retrieve most relevant documents. Bayesian classifiers, cluster analysis, decision trees, or artificial neural networks are methods to calculate weights and classify items to user preferences.

Textual information can easily be parsed and automatically categorized. For other types of information, such as multimedia data (e.g., images, music, and movies), categorization requires more complex operations performed manually by humans. However, this activity is error-prone, time-consuming, expensive, and highly subjective[8]. Therefore for an environment with variant amounts of information, dynamically increasing, content-based systems are not suitable. However, the problem with systems involving variant amounts of information can be avoided if information can be categorized without parsing.

2.4.2 Collaborative Filtering

The collaborative filtering method considers user preferences such as ratings, behaviors, or reviews to provide a filter for user preference information. Collaborative filtering systems are often classified as memory-based (user-based) or model-based (item-based). A memory-based collaborative filtering approach predicts item ratings based on all ratings given by various users for an item. A model-based approach predicts user ratings of all items from a particular set of items rated by the user. Collaborative filtering algorithms can be applied to any domain, as the algorithm considers explicit user feedback in the form of ratings. K-nearest neighbors, Pearson Correlation Coefficient are two of the approaches used to predict nearest relevance to the user.

User profile information is obtained through explicit or implicit feedback. Implicit feedback is obtained when the system automatically analyzes the user behavior based on factors such as browsing history, viewed items, and purchases made. Explicit feedback is also obtained by user ratings and reviews explicitly given by the user contributing to users feedback for constructions of the user's profile.

However, collaborative filtering has a 'cold start' problem. Similarities between users change needs to be determined when new ratings are posted. The new recommendations are determined using all old and the new modified data discarding the previously calculated recommendations. Therefore, the system must be constantly updated. Another problem with collaborative filtering is sparsity of data. A majority of items may not be rated by the user resulting diminished performance.

Chapter 3

Recommendation Approaches

Several algorithms can be applied on a dataset of recipes to compare and determine most accurate recommendation behavior. All approaches use a common experimental setup mentioned in Chapter 4. The primary algorithms for the experiments involved variations of content-based filtering, collaborative filtering, and hybrid approaches involving multiple algorithms.

3.1 Basic Approaches for Finding Similar Items

Before the algorithmic approaches applied for the recipe domain are described, the basis model used by all approaches must be defined. The **vector space model** a concept of representing documents, queries, and profiles in the form of vectors, was used extensively. The vector space model represents all basic recipe entities and corresponding features with user profiles and preferences in the form of vectors.

3.1.1 Vector Space Model

A vector space model is an algebraic model to represent text documents (and general objects) as vectors of identifiers, such as index terms[9]. A vector space model uses term occurrences as vector identifiers, so vector space model (VSM) is also known as term vector model.

Each document (an item in the source database, such as web page, images or text files etc.,), query, user profiles are represented in the form of vectors. Each vector dimension corresponds to a term, and based on term dimensions, various components have been determined to calculate corresponding weights. One way to calculate weights of the terms is to use term frequencies and inverse document frequencies (***TF-IDF***). For each orthogonal term i for a document or query j, a real valued weight W_{ij} is calculated.

$d_j = (W_{1,j}, W_{2,j}, ..., W_{t,j})$

$q = (W_{1,q}, W_{2,q}, ..., W_{t,q})$

The above vector format is represented for a document d_i with corresponding weights of each term in the form of $W_1, W_2 .., W_t$. Similarly, query 'q' is represented in the vector format. Once all the documents, queries or any other available information is in the form of vectors, vector operations are applied to compare queries with the document.

A vector space model follows three steps to retrieve documents similar to the query: (1) document cleaning and indexing, (2) term weighting, and (3) similarity measure.

Document Cleaning and Indexing

The first step of a vector space model is to extract particular content which bears terms from the document. It is evident that the document would contain some unrelated data such as is, the, a etc., which do not contribute to describe the contents of the document. Therefore, most of such stop words are removed in order to represent the document by the content bearing terms ([10]). In addition, terms can be stemmed to hold root term of the word, such as cats for catlike and catty, thus relating more terms and reducing the vocabulary. Next, *inverted index* with the terms and corresponding occurrences in the documents are constructed. Indexing can be based on term frequencies of the term in the document, as

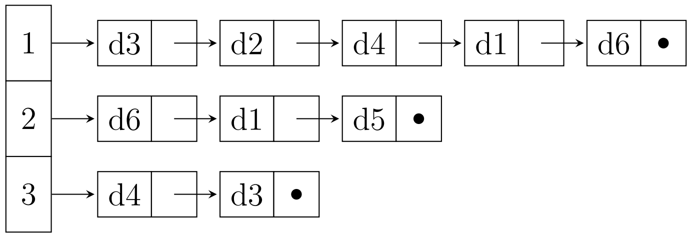


Figure 3.1: *Inverted Index Example*

discussed in the Section 3.1.1. Vocabulary dimension is determined by *orthogonal* terms formed after cleaning and indexing.

Figure 3.1 represents a sample inverted index with terms 1, 2 and 3 and term occurrences in various documents d1, d2, d3, d4, d5, and d6. In addition to indexing the documents, corresponding weights described in Section 3.1.1, of the term document can be calculated and stored in a similar inverted index.

Term Weighting

The second step of the vector space model is to weight indexed terms. Appropriate term weighting must be chosen in order to improve the retrieval of relevant document to the user. The most popular and efficient term weighting is use of term frequencies, inverse document frequencies[11] **TF-IDF**.

Most frequent terms in the document are considered more indicative of the document topic. Similarly, terms that appear in many documents are less indicative of the document topic.

To determine most frequent terms in the document:

$$\text{frequency of term i in document j : } \mathrm{f_{ij}}$$
$$\text{normalizing term frequencies : } \mathrm{tf_{ij}}=\mathrm{f_{ij}}/\mathrm{max_i}(\mathrm{f_{ij}})$$

To determine document frequencies for terms, such as to recognizing less indicative terms within many documents:

$$\left(\begin{array}{ccccc} & T_1 & T_2 & \cdots & T_t \\ D_1 & w_{11} & w_{21} & \cdots & w_{t1} \\ D_2 & w_{12} & w_{22} & \cdots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \cdots & w_{tn} \end{array}\right)$$

Figure 3.2: *Document-Item Matrix*

number of documents containing term i : df_i

Inverse Document Frequencies of term i : $idf_i = \log_2 N/df_i$

where N is the total number of documents.

A term occurring frequently in the document but rarely in the rest of the collection is given high weight. A typical tf-idf weighting which combines the occurrence of frequent terms in the document and less frequent in the remaining documents is given by:

$$W_{ij} = tf_{ij} * idf_i$$
$$W_{ij} = tf_{ij} * (\log_2 N/df_i)$$

At the end of term weighting step, weights can be directly stored in the inverted index for fast retrieval. A collection of documents can be represented in matrix format. An element in the matrix corresponds to a weight of the term in the particular document, as shown in figure 3.2.

Similarity Measures

The last step of VSM, is to rank the document with respect to the query or user profile according to a similarity measure. The similarity measure computes the degree of similarity

$$CosSim(x,y) = \frac{\sum_i (x_i)(y_i)}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}} \tag{3.1}$$

$$= \frac{\langle x, y \rangle}{||x||\ ||y||} \tag{3.2}$$

Figure 3.3: *Cosine Similarity*

between query and the document. The final document can be ordered based on the relevance when similarity with the query is determined.

Similarity in vector space models is determined by the use of associative coefficients based on inner product of the document vector and query vector, where word overlap indicates similarity. The inner product is typically normalized. The most popular similarity measures is the cosine coefficient and Pearson correlation coefficient. Cosine coefficient measures the angle between a document vector and query vector; Pearson correlation coefficient is a measure of linear correlation or dependency. Both correlation values lie between [-1,1].

Cosine similarity between two vectors can be represented as demonstrated in figure 3.3, where x $_i$ and y $_i$ represent two vectors. These vectors are the pairs of query and every document in the dataset.

Pearson correlation coefficient can be explained as a centered cosine; or normalized covariance. Pearson correlation coefficient is an invariant to shift; therefore, it considers the mean to normalize in the form, as shown in figure 3.4, where x_i and y_i represent two vectors and $\bar{x}, \bar{y}$ represents the respective mean of the vectors from the collection already calculated. Pearson correlation coefficient is represented as centered average inner product. Therefore, for the vector space model, the correlation measure mentioned in Figure 3.4 is efficient than cosine similarity measure.

$$Corr(x,y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2}\sqrt{\sum (y_i - \bar{y})^2}} \tag{3.3}$$

$$= \frac{\langle x - \bar{x},\ y - \bar{y} \rangle}{||x - \bar{x}||\ ||y - \bar{y}||} \tag{3.4}$$

$$= CosSim(x - \bar{x}, y - \bar{y}) \tag{3.5}$$

Figure 3.4: *Pearson Correlation Coefficient*

3.1.2 Extended Vector Space Model for Content-based Filtering

No explicit queries are given by the users in recommender system so the vector space model is extended for recommendations. Various stages of a vector space model described in Section 3.1.1 are used according to required recommendations to the user. Generically all user profiles are constructed as queries to the system.

Steps followed to recommend documents to users based on the user's profile are listed below

- Clean and Build Inverted Index, based on relevancy of terms to be considered. Terms vary based on domain. For example, ingredients can be considered terms for a recipe document.

- Construct the document-term matrix and store values in the inverted index for faster computation.

- Construct a user profile similar to vectors of the documents. For example, construct user profiles with terms considered for inverted index such that user profiles and documents are comparable. User profile can be constructed based on the documents user has already considered in his profile.

- Calculate similarities of all user profile vectors with document vectors and sort order according to relevance. Documents previously considered by the user (i.e., documents already on the user profile) should be omitted to calculate relevance.

- Most relevant documents not already considered by the user can be offered as recommendations

3.2 Proposed Approaches

The following are various approaches used in this paper to recommend related recipes.

3.2.1 User Recommendations based on Recipe Ratings

A memory-based collaborative filtering for the user and recipes was applied. This approach utilizes information related to user ratings on the recipes. An item-based collaborative filtering with user profiles was applied to discern the recommendations.

The following steps were followed to obtain recommendations for a user:

- Extract all user profiles and recipes, such that every user and recipe can be represented by a unique key value.
- Obtain all ratings given by the user for each recipe. If no feedback is provided by the user, the corresponding rating input is blank or not used in the algorithm.
- From all obtained user ratings, prepare a preference matrix on the recipe vectors.
- Calculate user that are similar to each other with the row similarity job.
- For any user 'u', the rating given to the item 'i' is predicted using an aggregate function.

$$\mathrm{r}_{\mathrm{u,i}}=\mathrm{aggr}_{\mathrm{u'} \in \mathrm{U}}\ \mathrm{r}_{\mathrm{u',i}}$$

where 'U' is the top N users similar to the user 'u', and the aggregate function 'aggr' can be denoted in the similarity measure.

$$\mathrm{r}_{\mathrm{u,i}}=\sum_{u' \in U} Simil(u,u')\ \mathrm{r}_{\mathrm{u',i}}$$

The SIMIL(u,u') uses the similarity measure mentioned in Figures 3.3 and 3.4.

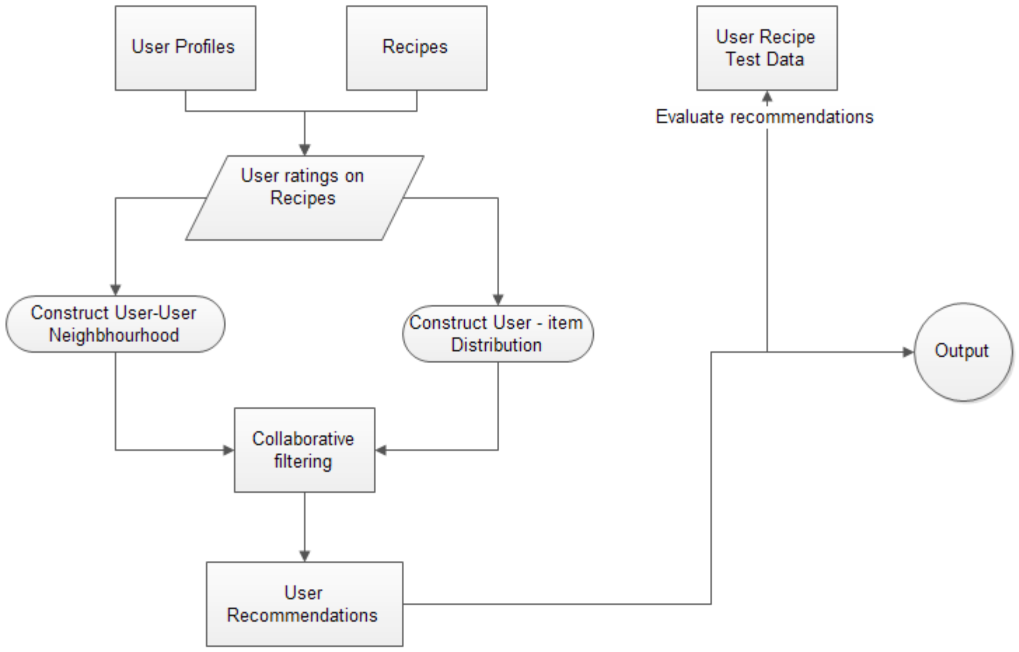


Figure 3.5: *Workflow for Rating-based Recommendation Approach*

- Predicted ratings of the recipes are used to retrieve recommendations for a user. The total number of recommendations requested are extracted as the recipes with highest predicted ratings. Predicted ratings with certain thresholds (a rating greater than or equal to 3 out of 5) are only considered for recommendations.

  If the number of recommendations requested is more than the total predicted ratings on recipes for the user, all the recipes predicted above the threshold can be given as recommendations.

For example, the rating of the user for corresponding recipes needs to be predicted and then all similar users are predicted with the user-user distribution and user-item distribution. From this distribution of ratings, the blank rating is predicted and recommended if the result is liked.

3.2.2 User Recommendations based on Recipe Ingredients

Recipe Ingredients are used to calculate user recommendations in recipe ingredient based recommendations approach. The user and recipe vectors are constructed based on ingredients as orthogonal terms, or the dimension of the vectors.

A high-level overview of recipe ingredient based recommendations approach is stated in the Figure 3.6.

- Extract all recipe ingredients from recipe data.
- Obtain unique ingredient terms with corresponding quantities. Multiple ingredients as a single unit should be split into unit set ingredients. For example, ingredients separated with 'and'/'or'/'instead' are split into independent or separate ingredients. Quantity associated with ingredients for the recipes are considered as the frequency of ingredient occurrence in the recipe.
- Construct inverted index with ingredients as terms and quantities as frequencies from all recipes. Common ingredients such as salt (to taste) are considered for a quantity of 1.
- Calculate maximum ingredient frequencies of the recipe, and the total number of recipes.
- Calculate the term frequencies and inverted document frequencies of all terms as described in Section 3.1.1. Then construct the inverted index with corresponding weights.
- Construct all recipe vectors for each recipe in the dataset.
- Build user vectors based on user-liked ingredients or user liked recipes. For all recipes rated by the user with a threshold (rating greater than or equal to 3 for a maximum rating of 5) acceptable rating, extract all ingredients and construct an ingredient dimension vector from the inverted index previously constructed.

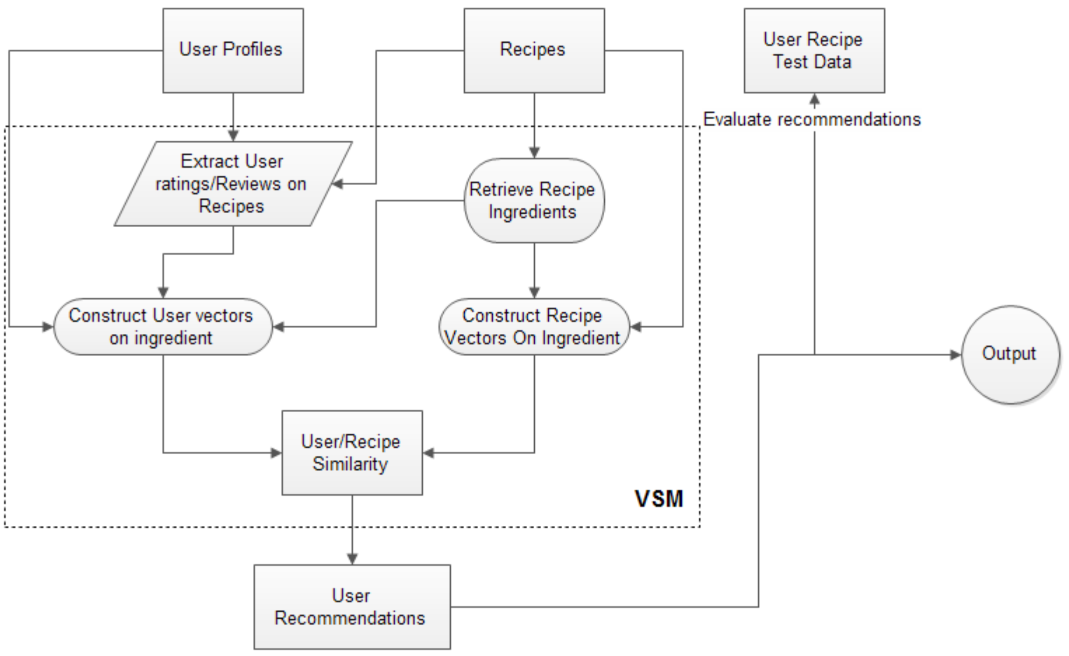

User Profiles
Recipes
User Recipe Test Data
Evaluate recommendations
Extract User ratings/Reviews on Recipes
Retrieve Recipe Ingredients
Construct User vectors on ingredient
Construct Recipe Vectors On Ingredient
Output
User/Recipe Similarity
VSM
User Recommendations

Figure 3.6: *Workflow for Ingredient-based Recommendation Approach*

- User recommendations are calculated from user vectors and ingredient vectors. User vector and all recipe vectors, represented in the ingredient dimension, are evaluated against similarity measures as cosine (Figure 3.3), Pearson correlation coefficient (Figure 3.4) and the closest relevant recipes with highest similarity are given as recommendations.

Ingredient-based Similarity of Recipes

A primary approach to determine recipe similarity among the recipes themselves involves the use of vector space model with queries as documents. In ingredient-based similarity, recipe ingredients are used to calculate similarity between recipes. For each ingredient as terms and the recipes as documents, a pairwise recipe similarity matrix is constructed. No user profile is required for this approach, and the recommendations are given to any anonymous user. Recipes similar to the current recipe are given as recommendations.

This approach follows the following steps:

- Extract all recipe ingredients from the recipe data.
- Obtain unique ingredient terms with corresponding quantities. Multiple ingredients as a single unit should be split into unit set ingredients. For example, ingredients separated with 'and'/'or'/'instead' are split into different and independent ingredients. Quantity associated with recipe ingredients are considered as the frequency of ingredient occurrence in the recipe.
- Construct inverted index with ingredients as terms and quantities as frequencies from all recipes. Common ingredients such as salt (to taste) are considered as a quantity of 1.
- Calculate maximum ingredient frequencies of the recipe, and the total number of recipes.
- Calculate the term frequencies and inverted document frequencies of all the terms as described in Section 3.1.1. Then construct the inverted index with corresponding weights.
- Construct all recipe vectors for each recipe in the dataset.
- Build a **pairwise similarity matrix** for all recipes to store the similarity correlation. Similarity measures could be cosine 3.3, Pearson correlation coefficient 3.4, or any other similarity measure obtained from the constructed recipe vectors.
- When a user requests recommendations on a recipe, the most closest relevant recipes from the *pairwise similarity matrix* are given as recommendations.

The ingredient based similarity on recipes approach is difficult to evaluate as construction of a test set of recommendations to compare resulting recommendations is difficult. The

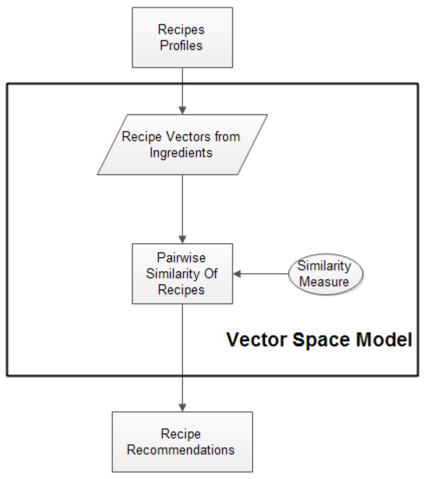


Figure 3.7: *Workflow for Recommending Similar Recipes based on Ingredients*

test set is difficult to construct due to lack of user profile and also because recommendations are specific to each recipe. However, the algorithm approach is the basis for all approaches mentioned in Sections 3.2.2, 3.2.3, 3.2.4

3.2.3 User Recommendations based on Recipe Review Text

User recommendations for recipes are computed based on textual review given by the user. The user recommendations based on review text approach also considers the explicit feedback offered by the user as part of the review text given for the document. User recommendations based on review text approach can be compared to a document search model with the vector dimension acting as vocabulary of the review text. A high-level overview of this approach is stated in the Figure 3.8

- Clean the review text by removing stop words such as a, the, is etc., which do not contribute to review content. Stem all words such that terms are orthogonal forming the dimension of the vectors to be constructed.

- Construct inverted index with terms and corresponding frequencies from the reviews of all recipes.

- Calculate maximum ingredient frequencies of the recipe, and the total number of recipes.
- Calculate the term frequencies and inverted document frequencies of all terms as described in Section 3.1.1. Then construct the inverted index with corresponding weights.
- Construct all recipe vectors for each recipe in the dataset using the inverted index of the review text.
- Build user vectors based on user reviews. For every review of the user, extract all terms and build a vector with the inverted index previously constructed. To obtain user vectors more inclined to the user profile, consider reviews for recipes which the user liked.
- User recommendations are calculated from user vectors and ingredient vectors. User vector and all recipe vectors represented in the ingredient dimension are evaluated against the similarity measures as cosine (Figure 3.3), Pearson correlation coefficient (Figure 3.4), and the closest and most relevant recipes with highest similarity are given as recommendations.

3.2.4 User Recommendations based on Recipe Ingredients and Review Text

User recommendations based on ingredients and review text approach uses similarity measures of multiple methods to provide recommendations. It also utilizes the results of both the methods Ingredient vector approach (Section 3.2.2), review text approach (Section 3.2.3) to compute recommendations for the user.

A high-level overview of hybrid approach is presented in Figure 3.9

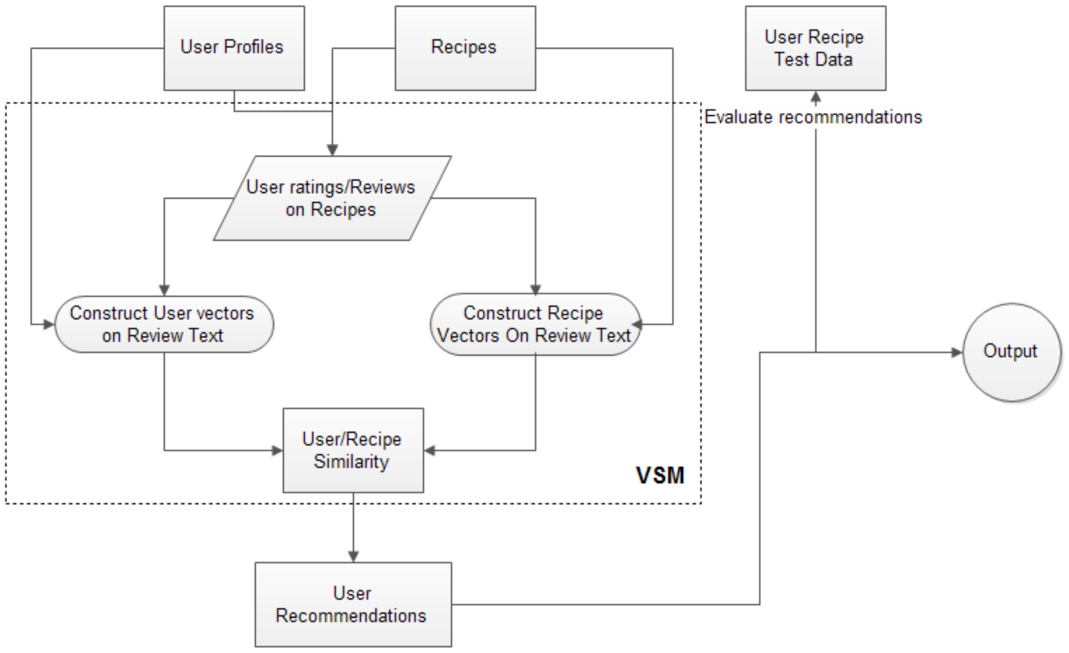


Figure 3.8: *Workflow for Review Text-based Recommendation Approach*

- Similarity measure was calculated as described in ingredient vector approach in Section 3.2.2.
- Similarity measure was calculated as described in review text approach in Section 3.2.3.
- Based on the considered similarity, measure considered cosine (Figure 3.3), Pearson correlation coefficient (Figure3.4) the average of similarity values for all combinations are computed.
- The most similar recipes identified from the average of the two approaches are given as recommendations to the user.

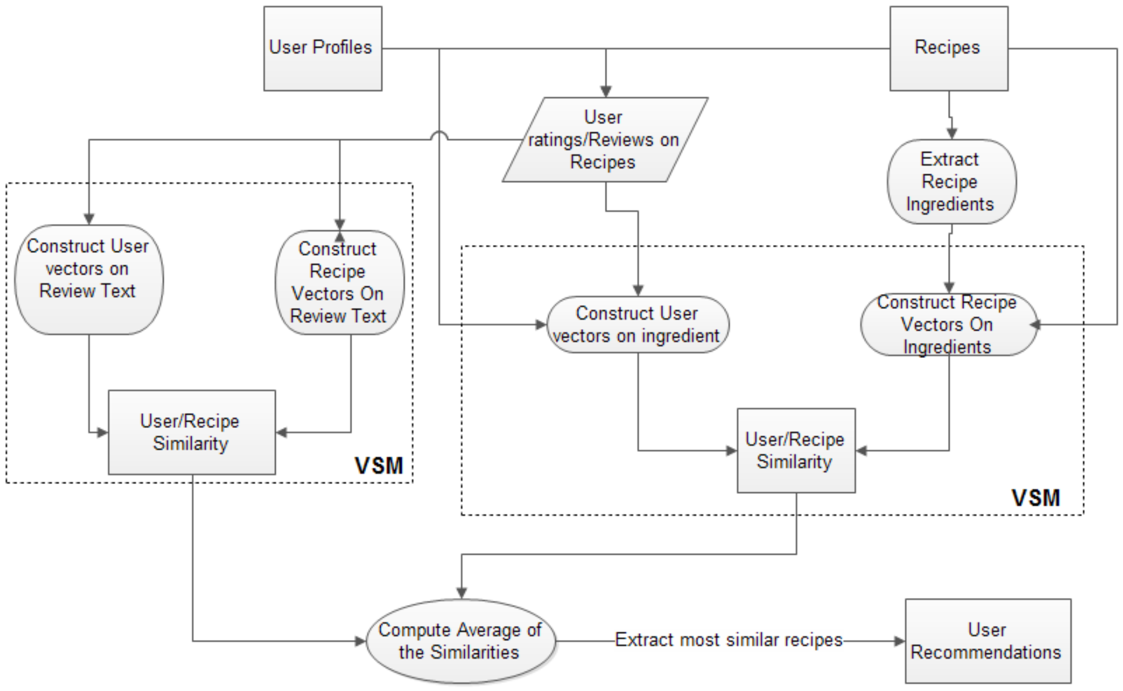

User Profiles
Recipes
User ratings/Reviews on Recipes
Extract Recipe Ingredients
Construct User vectors on Review Text
Construct Recipe Vectors On Review Text
Construct User vectors on ingredient
Construct Recipe Vectors On Ingredients
User/Recipe Similarity
VSM
User/Recipe Similarity
VSM
Compute Average of the Similarities
Extract most similar recipes
User Recommendations

Figure 3.9: *Workflow for Ingredient and Review Text-based Recommendation Approach*

Chapter 4

Experimental Setup

This chapter explains the dataset used and the experiments designed to evaluate this study's approach. Experiments were conducted with various approaches mentioned in Chapter 3. Section 4.5 lists the set of experiments performed for this study. Section 4.2 describes the dataset used.

4.1 Overview

A high-level overview of the experimental setup is shown in Figure 4.1. Data from www.allrecipes.com was crawled and stored in the local dataset in order to perform the experiments. Details of the dataset are described in Section 4.2. The complete dataset was randomly split into two parts such that 70% of the data was in the training set and 30% was in the test. The training set was given to the algorithm to provide recommendations, and then the training set was evaluated across the test set to calculate the Mean Average Precision (MAP) for the experiment. The experiment was repeated for datasets of five iterations generated from the training and test folds. Training and test fold setup and corresponding datasets are described in Section 4.3 and a high-level overview of the experiments performed is described in

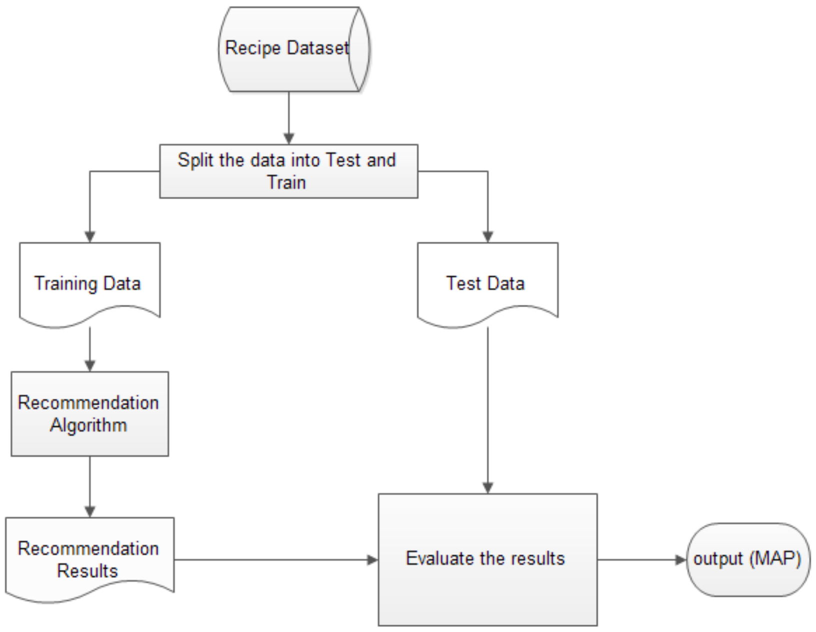
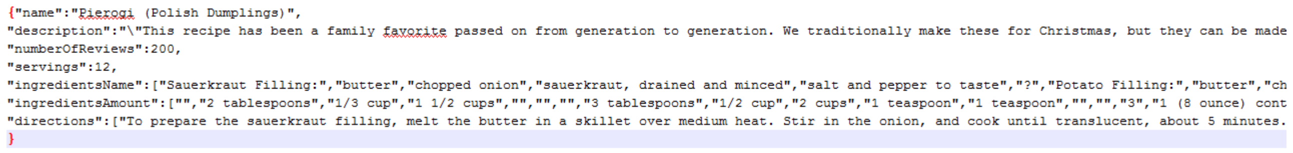


Figure 4.1: *Experimental Setup*

Section 4.5. The evaluation measure used was MAP which is explained in detail in Section 4.4.

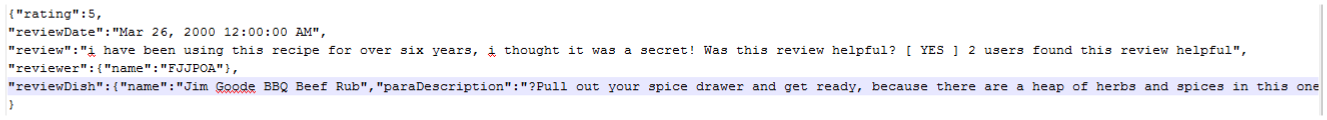
4.2 Data Description

Two separate files stored a set of recipes and reviews of corresponding recipes. Recipe data and review data were stored in JSON (JavaScript Object Notation) format after being extracted from a website known as www.allrecipes.com. The data set consisted of 45,668 recipes with 2,845,167 reviews reviewed by 585,700 users. One input file, JSONRecipes, consists of all recipes including ingredient details, cook details, preparation details. All of these were in JSON format. Similarly, reviews that included reviewer, ratings, and reviews of a recipes was a JSONReviews file. Both the files were processed to extract unique recipe names and user names in order to generate unique ID's for each of the unique user names. Furthermore, reviews were processed to have only the set for which every user had atleast



```
{"name":"Pierogi (Polish Dumplings)",
"description":"\"This recipe has been a family favorite passed on from generation to generation. We traditionally make these for Christmas, but they can be made
"numberOfReviews":200,
"servings":12,
"ingredientsName":["Sauerkraut Filling:","butter","chopped onion","sauerkraut, drained and minced","salt and pepper to taste","?","Potato Filling:","butter","ch
"ingredientsAmount":["","2 tablespoons","1/3 cup","1 1/2 cups","","","","3 tablespoons","1/2 cup","2 cups","1 teaspoon","1 teaspoon","","","3","1 (8 ounce) cont
"directions":["To prepare the sauerkraut filling, melt the butter in a skillet over medium heat. Stir in the onion, and cook until translucent, about 5 minutes.
}
```

Figure 4.2: *JSON Recipes*



```
{"rating":5,
"reviewDate":"Mar 26, 2000 12:00:00 AM",
"review":"i have been using this recipe for over six years, i thought it was a secret! Was this review helpful? [ YES ] 2 users found this review helpful",
"reviewer":{"name":"FJJPOA"},
"reviewDish":{"name":"Jim Goode BBQ Beef Rub","paraDescription":"?Pull out your spice drawer and get ready, because there are a heap of herbs and spices in this one
}
```

Figure 4.3: *JSON Reviews*

3 reviews to properly split the data into training and test folds. Considering users with at least 3 reviews negligibly reduced the total number of reviews. A sample recipe and review data are shown in Figures 4.2 and 4.3

Figure 4.4 shows a class diagram used to read/write the input files in JsonReview and JsonRecipe formats.

Data statistics include :

- Total number of recipes = 45,668
- Total number of users = 585,700
- Total number of reviews = 2,394,505

4.3 Dataset Folds, Training and Test Datasets

Input data was divided into five folds and each fold was randomly divided in order to independently perform experiments on all folds. Each fold was further split into a training set and test dataset. Seventy percent of the data (or reviews of the user) was in the training set and 30% was given to the test set. For each fold, experiments were performed on the

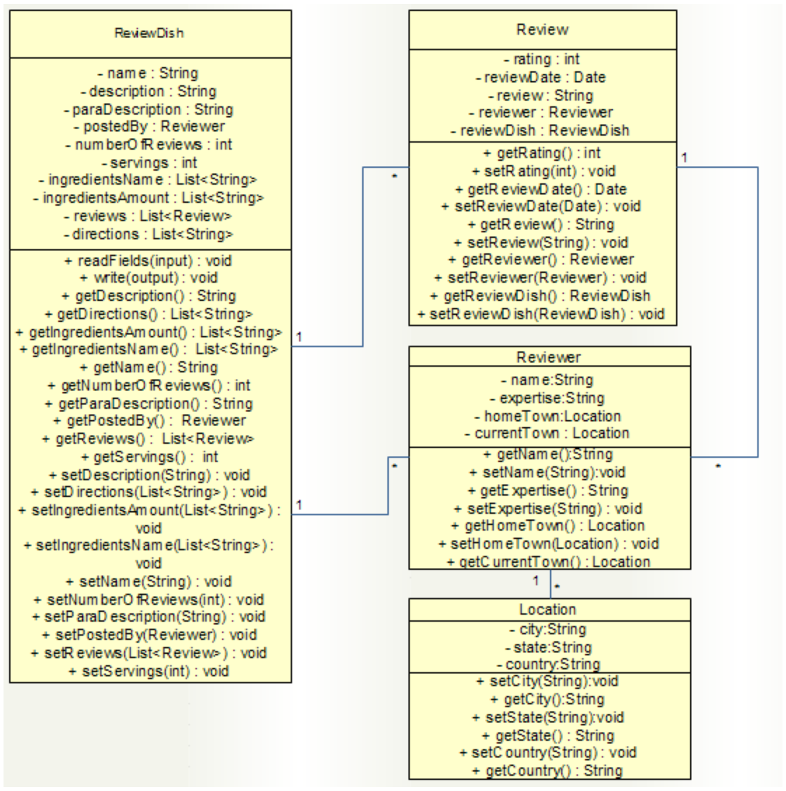

ReviewDish
- name : String
- description : String
- paraDescription : String
- postedBy : Reviewer
- numberOfReviews : int
- servings : int
- ingredientsName : List<String>
- ingredientsAmount : List<String>
- reviews : List<Review>
- directions : List<String>
+ readFields(input) : void
+ write(output) : void
+ getDescription() : String
+ getDirections() : List<String>
+ getIngredientsAmount() : List<String>
+ getIngredientsName() : List<String>
+ getName() : String
+ getNumberOfReviews() : int
+ getParaDescription() : String
+ getPostedBy() : Reviewer
+ getReviews() : List<Review>
+ getServings() : int
+ setDescription(String) : void
+ setDirections(List<String>) : void
+ setIngredientsAmount(List<String>) : void
+ setIngredientsName(List<String>) : void
+ setName(String) : void
+ setNumberOfReviews(int) : void
+ setParaDescription(String) : void
+ setPostedBy(Reviewer) : void
+ setReviews(List<Review>) : void
+ setServings(int) : void
Review
- rating : int
- reviewDate : Date
- review : String
- reviewer : Reviewer
- reviewDish : ReviewDish
+ getRating() : int
+ setRating(int) : void
+ getReviewDate() : Date
+ setReviewDate(Date) : void
+ getReview() : String
+ setReview(String) : void
+ getReviewer() : Reviewer
+ setReviewer(Reviewer) : void
+ getReviewDish() : ReviewDish
+ setReviewDish(ReviewDish) : void
Reviewer
- name:String
- expertise:String
- homeTown:Location
- currentTown : Location
+ getName():String
+ setName(String):void
+ getExpertise() : String
+ setExpertise(String) : void
+ getHomeTown() : Location
+ setHomeTown(Location) : void
+ getCurrentTown() : Location
Location
- city:String
- state:String
- country:String
+ setCity(String):void
+ getCity():String
+ setState(String):void
+ getState() : String
+ setCountry(String) : void
+ getCountry() : String
1
*
1
*
1
*
1
*

Figure 4.4: *Class Diagram to Read Input Files*

training dataset and evaluated across the corresponding test dataset. All folds were built on random input; however, the training set had 1,652,625 reviews and the test set had 741,880 reviews for all five folds.

A review with a rating greater than or equal to 3 was considered a user-liked recipe. Those rated recipes were considered as output for the test set, and the algorithm used the training set to generate comparable recommendations to the test set.

Table 4.1: *Statistics for All Training Folds*

Folds	Recipes	Users	Reviews(Training)	Reviews Considered (Ratings≥3)
Fold 1	45668	585700	1652625	121684
Fold 2	45668	585700	1652625	121502
Fold 3	45668	585700	1652625	121728
Fold 4	45668	585700	1652625	121771
Fold 5	45668	585700	1652625	121798

Table 4.2: *Statistics for All Test Folds*

Folds	Recipes	Users	Reviews(Test)	Reviews Considered (Ratings≥3)
Fold 1	45668	585700	741880	54160
Fold 2	45668	585700	741880	54342
Fold 3	45668	585700	741880	54116
Fold 4	45668	585700	741880	54073
Fold 5	45668	585700	741880	54046

4.4 Evaluation Metric: Mean Average Precision

For all approaches, recommendations were evaluated as a measure of MAP. Recommendations generated by algorithms applied on the training set were evaluated with the test set in order to find corresponding positions and determine precision points of each recommendation. The MAP is the average of precision points.

Precision of a recommended recipe/ingredient is calculated as follows:

Precision = (A/(A+C))*100

C: Number of irrelevant documents retrieved

A: Number of relevant documents retrieved

All precision points were evaluated for user recommendations. For precision values of all user outcomes, the mean was computed to determine the Mean Average Precision (MAP) for the corresponding approach.

4.5 Experiments

Various sets of experiments were performed for each approach mentioned in Chapter 3. All experiments evaluated the approaches on a test data set for every user profile. However, the algorithm in Section 3.2.2 does not require any user profile and no test set is available to evaluate this approach, but the approach based on ingredient similarity mentioned in Section 3.2.2 forms the basis for other algorithms based on ingredients such as user similarities on ingredients as described in Sections 3.2.2, 3.2.4.

Every algorithm was evaluated with a similarity measure of cosine and Pearson correlation coefficient for a set of 10 and 20 recommendations each thus dividing all experiments into sets of two for each algorithm. The following description offers additional detail as to the type of experiments performed for all algorithms in Chapter 3.

4.5.1 Evaluate Recommendations based on Ratings

Ratings given to recipes were extracted from the dataset as described in Section 4.2 and unrated combinations of user and recipes were predicted as mentioned in Section 3.2.1. Experiments 1 and 2 evaluate the same algorithm approach but by using different the similarity measure.

Experiment 1: The purpose of this experiment was to evaluate the user recommendations on Ratings algorithm mentioned in Section 3.2.1 using *cosine similarity measure*. Experiment 1 determined the MAP of the results when compared across a randomly generated test set. The dataset used for this experiment is described in Section 4.2. Experiments were performed for an output set of 10 and 20 recommendations.

Experiment 2: The purpose of this experiment was to evaluate the User recommendations on Ratings algorithm mentioned in Section 3.2.1 comparing with *Pearson correlation coefficient* similarity measure. Experiment 2 determined the MAP of the results when compared across a randomly generated test set. The dataset for this experiment is described in Section 4.2. Experiments were performed for an output set of 10 and 20 recommendations.

4.5.2 Evaluate Recommendations based on Ingredients

Experiment 3: The purpose of this experiment was to evaluate the user recommendations on Ingredients algorithm mentioned in Section 3.2.2 utilizing *cosine similarity measure*. Experiment 3 determined the MAP of the results when compared across a randomly generated test set. The dataset for this experiment is described in Section 4.2. Experiments were performed for an output set of 10 and 20 recommendations.

Experiment 4: The purpose of this experiment was to evaluate the user recommendations on Ingredients algorithm mentioned in Section 3.2.2 using *Pearson correlation coefficient similarity measure*. Experiment 4 determined the MAP of the results when compared across a randomly generated test set. The dataset for this experiment is described in Section 4.2. Experiments were performed for an output set of 10 and 20 recommendations.

For Experiments 3 and 4, recipe ingredients, and customer ratings were the core components used in the dataset. Recipe ingredients and customer ratings on recipes were extracted from the large dataset mentioned in Section 4.2.

4.5.3 Evaluate Recommendations based on Review Text

Experiment 5: The purpose of this experiment was to evaluate the user recommendations on review text algorithm mentioned in Section 3.2.3 using *cosine similarity measure*. Experiment 5 determines the MAP of the results when compared across a randomly generated test set. The dataset used for this experiment is described in Section 4.2. Experiments were performed for an output set of 10 and 20 recommendations.

Experiment 6: The purpose of this experiment was to evaluate the user recommendations on review text algorithm mentioned in Section 3.2.3 utilizing *Pearson correlation coefficient similarity measure*. Experiment 6 determined the MAP of the results when compared across a randomly generated test set. The dataset that used for this experiment is described in Section 4.2. Experiments were performed for an output set of 10 and 20 recommendations.

For Experiments 5 and 6 the user reviews and customer ratings were the core components used in the dataset. Reviews in text of recipes and customer ratings with corresponding recipe reviews were extracted from the large dataset mentioned in Section 4.2. This dataset was cleaned to handle review text experiments. Experiments 5 and 6 evaluate the same algorithm approach but for a different similarity measure.

4.5.4 Evaluate Recommendations based on Ingredients and Review Text

The complete dataset described in Section 4.2 was utilized in these experiments. Ratings were extracted from the dataset in order to construct the test dataset. Recipe ingredients and review text of the user on recipes were also extracted from the same dataset. Experiments 7 and 8 evaluated the same algorithm approach but for a different similarity measure.

Experiment 7: The purpose of this experiment was to evaluate the user recommendations on review text and the recipe ingredient algorithm mentioned in Section 3.2.4 utilizing *cosine similarity measure*. Experiment 7 determined the MAP of the results when compared across a randomly generated test set. The dataset used for this experiment is described in Section 4.2. Experiments were performed for an output set of 10 and 20 recommendations.

Experiment 8: The purpose of this experiment was to evaluate the user recommendations on review text and the recipe ingredient algorithm mentioned in Section 3.2.4 using *Pearson correlation coefficient* similarity measure. Experiment 8 determined the MAP of the results when compared across a randomly generated test set. The dataset used for this experiment is described in Section 4.2. Experiments were performed for an output set of 10 and 20 recommendations.

Chapter 5

Results

5.1 Experiment 1 Results

Table 5.1 shows the average of MAP (Mean Average Precision) for the algorithm in order to generate recommendations based on ratings described in Section 3.2.1 over the dataset described in Section 4.2.

Table 5.1: *Results for Experiment 1: Using Cosine Similarity to Make Recommendations based on Ratings*

	Number of Recommendations	
Fold	10	20
Fold 1	0.00144074965126068	0.00198493215185491
Fold 2	0.0010925630563132	0.00115491027416314
Fold 3	0.000962693863675622	0.00122997865361737
Fold 4	0.00094435526647306	0.00137860567666071
Fold 5	0.00106470022445386	0.00160853316591987
Average of all Folds	0.0011010124	0.0014713920

Experimental results show that recommendations generated based on ratings had a relatively good MAP. However, a change in the number of recommendations resulted in a negligible change in resulting values. The cosine similarity measure for Section 3.2.1 is better than the Pearson correlation coefficient measure.

5.2 Experiment 2 Results

Table 5.2 shows the average of MAP for the algorithm in order to generate recommendations based on ratings described in Section 3.2.1 over the dataset described in Section 4.2 using Pearson correlation coefficient similarity measure.

Table 5.2: *Results for Experiment 2: Evaluate User Recommendations on Ratings by Pearson correlation Coefficient Measure*

	Number of Recommendations	
Fold	10	20
Fold 1	0.000718911837151497	0.000858195410660199
Fold 2	0.00061512709103312	0.000654555627984316
Fold 3	0.000607430663641943	0.0010303553457882
Fold 4	0.000370247538390318	0.000568127243207782
Fold 5	0.000918472060501275	0.000697737097567657
Average of all Folds	0.0006460378	0.0007617941

Experimental results show that recommendations generated based on ratings had a relatively good MAP. However, an increase in MAP occurred with an increase in number of recommendations. This behavior is unique, since the expected result may have many false positives, thereby reducing MAP. The Pearson correlation coefficient similarity measure for Section 3.2.1 was not better than the cosine similarity measure.

5.3 Experiment 3 Results

Table 5.3 shows the average of MAP for the algorithm in order to generate recommendations based on ratings described in Section 3.2.2 over the dataset described in 4.2 with cosine similarity measure.

Table 5.3: *Results for Experiment 3: Evaluate User Recommendations on Recipe Ingredients by Cosine Similarity Measure*

	Number of Recommendations	
Fold	10	20
Fold 1	0.000153401676285802	0.000153401676285802
Fold 2	0.000157027162947017	0.000157027162947017
Fold 3	0.000131198746515271	0.000131198746515271
Fold 4	0.000192523525856859	0.000192523525856859
Fold 5	0.000158055039571627	0.000158055039571627
Average of all Folds	0.0001584412	0.0001584412

MAP calculated using the ingredients with cosine similarity measure projected relatively good results. However, the MAP was constant even though the number of recommendations increased because the number of recommendations produced less than the minimum required number of recommendations (i.e., ¡10 recommendations) for both cases. Therefore, even though the results were good, the produced recommendations were highly satisfactory due to the constant MAP.

The cosine similarity measure for ingredient similarity was better than the Pearson correlation similarity measure for less recommendations but the scenario changed with an increased number of recommendations.

5.4 Experiment 4 Results

Table 5.4 shows the average of MAP for the algorithm in order to generate recommendations based on ratings described in Section 3.2.2 over the dataset described in Section 4.2 with Pearson correlation coefficient similarity measure.

The MAP calculated using the ingredients with Pearson correlation coefficient similarity measure was better than the cosine similarity measure for less recommendations. However,

Table 5.4: *Results for Experiment 4: Evaluate User Recommendations on Recipe Ingredients by Pearson Correlation Coefficient Measure*

	Number of Recommendations	
Fold	10	20
Fold 1	0.0000253789685120157	0.000130076793442712
Fold 2	0.000112294516800954	0.000126089082223149
Fold 3	0.0000402128186092551	0.000277762368938264
Fold 4	0.000195847362514029	0.000298647746016167
Fold 5	0.000107823042104364	0.000171012123356473
Average of all Folds	0.0000963113	0.0002007176

an increased number of recommendations resulted in increased result value for MAP, but with a significantly lower increase compared to cosine similarity measure.

5.5 Experiment 5 Results

Table 5.5 shows the average of MAP for the algorithm in order to generate recommendations based on ratings described in Section 3.2.3 over the dataset described in Section 4.2 with cosine similarity measure.

Table 5.5: *Results for Experiment 5: Evaluate User Recommendations on User Review Text by Cosine Similarity Measure*

	Number of Recommendations	
Fold	10	20
Fold 1	0.0000362394229435427	0.0000758836807525946
Fold 2	0.0000281282996659223	0.0000304887164211046
Fold 3	0.0000228538038671232	0.0000233286489489153
Fold 4	0.0000237518037518037	0.0000238548752834467
Fold 5	0.0000157750895844891	0.0000708431965278243
Average of all Folds	0.0000253497	0.0000448798

The MAP for the approach with review text was relatively low compared to ingredients as vectors due to large vector dimension for review text. An increase in vector dimensions may reduce similarity values. Moreover, similar reviewed recipes may not always be similar. However, the use of cosine measure improves the MAP with increased number of recommendations.

5.6 Experiment 6 Results

Table 5.6 shows the average of MAP for the algorithm in order to generate recommendations based on ratings described in Section 3.2.3 over the dataset described in Section 4.2 with Pearson correlation coefficient similarity measure.

Table 5.6: *Results for Experiment 6: Evaluate User Recommendations on User Review Text by Pearson Correlation Coefficient Measure*

	Number of Recommendations	
Fold	10	20
Fold 1	0.000312754489734687	0.0000103135313531353
Fold 2	0.00114399545257033	0.000187245993405413
Fold 3	0.00200040597389549	0.0000130154815728181
Fold 4	0.000201643378623576	0.00000603864734299516
Fold 5	0.0010328843414622	0.0000378630225675175
Average of all Folds	0.0009383367	0.0000508953

The MAP for the approach with review text was relatively high compared to ingredients with Pearson correlation coefficient similarity measure. However, use of the Pearson correlation coefficient measure significantly decreased the MAP when the number of recommendations increased.

5.7 Experiment 7 Results

Table 5.7 shows the average of MAP for the algorithm in order to generate recommendations based on ratings as described in Section 3.2.4 over the dataset described in Section 4.2 with cosine similarity measure.

Table 5.7: *Results for Experiment 7: Evaluate User Recommendations with Hybrid approach by Cosine Similarity Measure*

	Number of Recommendations	
Fold	10	20
Fold 1	0.0000464157075769714	0.0000979736374572268
Fold 2	0.0000458487465403449	0.000055824367114407
Fold 3	0.0000493205942618071	0.0000565927292571694
Fold 4	0.0000382213332877469	0.0000550880925364684
Fold 5	0.000052284230526673	0.000118948750280643
Average of all Folds	0.0000464181	0.0000768855

The hybrid approach with cosine similarity measure falls like an intermediate path for both the ingredient similarity approach and review text similarity approach. The MAP value significantly improved with an increased number of recommendations using cosine similarity measure.

5.8 Experiment 8 Results

Table 5.8 shows the average of MAP for the algorithm in order to generate recommendations based on ratings as described in Section 3.2.4 over the dataset described in Section 4.2 with Pearson Correlation Coefficient similarity measure.

The hybrid approach with Pearson correlation coefficient similarity measure provided intermediate results for ingredient similarity approach and review text similarity approach. The MAP value decreased with an increased number of recommendations using Pearson correlation coefficient similarity measure.

Table 5.8: *Results for Experiment 8: Evaluate User Recommendations with Hybrid approach by Pearson Correlation Coefficient Measure*

	Number of Recommendations	
Fold	10	20
Fold 1	0.000138484106528316	0.0000779907451667064
Fold 2	0.000523594188190461	0.000150355511552621
Fold 3	0.00040867093105899	0.000187143048058312
Fold 4	0.000123530982905982	0.000193891813346098
Fold 5	0.0000629447181171319	0.000119258431341885
Average of all Folds	0.0002514450	0.0001457279

5.9 Summary of Results

Table 5.9 shows the results of all the approaches described in Chapter 3 applied on the dataset described in Section 4.2 with the cosine similarity measure described in Section 3.3 and Pearson correlation coefficient described in Section 3.4. The Similarity columns identifies the similarity measure used, 'Reco' is the number of recommendations, and the other columns are resultant MAPs for approaches mentioned in Chapter 3. Recommendations generated based on rating described in Section 3.2.1 stands-out from all the other approaches and it is used as a benc- mark to compare with the approaches described in Sections 3.2.2, 3.2.3, and 3.2.4. Recommendations based on ratings were expected to perform the best because the test set to evaluate approaches considered only highly rated recipes as recommendations. Therefore, the approach to find recommendations using ratings was not comparable to other approaches.

Ingredient-based similarity recommendation looks better than other approaches for cosine similarity measure even with a range of number of recommendations. The review text similarity performed better with Pearson correlation coefficient similarity measure with less number of recommendations. However, as the number of recommendations increased ingredients-based similarity approach performed better. The hybrid approach took the intermediate path of ingredient-based approach and review text approach due to its average

Table 5.9: *Experiment: Evaluation Results of All approaches by Cosine Similarity Measure*

		Recommendation based on			
Similarity	Reco	Ratings	Ingredients	Review Texts	Hybrid
Cosine	10	0.0011010124	0.0001584412	0.0000253497	0.0000464181
Cosine	20	0.0014713920	0.0001584412	0.0000448798	0.0000768855
Pearson Correlation Coefficient	10	0.0006460378	0.0000963113	0.0009383367	0.0002514450
Pearson Correlation Coefficient	20	0.0007617941	0.0002007176	0.0000508953	0.0001457279

of results of both approaches.

Chapter 6

Discussion and Conclusions

This chapter discusses the optimization of all approaches, draws conclusions, and also addresses limitations of these approaches.

Based on results, the argument can be made that the algorithm to find recommendations of ratings given by the user on recipes perform better. However, the user rating approach extends collaborative filtering and should perform better because the test set was built based on user liked recipes with good ratings. Therefore, recommendations on ratings approach are comparable to other approaches as a baseline and among other approaches, performance varies based on similarity measure and change in number of recommendations.

For cosine similarity measure, ingredient-based similarity performed better and looks to be constant with an increased number of recommendations. When an increase in number of recommendations occurred, a slight increase in the review text similarity approach was observed, but that increase became static after a certain point. The hybrid approach falls in between ingredient similarity and review text similarity as per its definition.

However, with the use of Pearson correlation coefficient as the similarity measure, recommendations on review text offered better results with less number of recommendations. The scenarios changes to a normal expected behavior with increase in number of recommendations.

The recipe dataset was sparse. For example, a recipe with few common ingredients such as salt and pepper were found in almost all recipes and some ingredients were used in relatively very less recipes. Therefore, the performance to be predicted becomes difficult when the dataset increases.

Chapter 7

Future Work

This chapter discusses improvements and possible future directions for the approaches described in this paper. The following work can be performed to extend future work of this project.

1. The approaches can be extended to study the behavior of recommendations on recipe domain with varying similarity coefficients such as log likelihood and maximum likelihood

2. The algorithms can be applied on denser datasets to test whether or not it yields better results.

3. In addition to the above experiments, other classification approaches such as Naive classifier, Bayes classifier, or support vector machine (SVM) can be implemented to study trend changes.

4. Furthermore, the approach could be comparable with more advanced algorithms such as adsorptions involving recommendations from graphs of the heterogeneous network.

Bibliography

[1] Resnick and Varian. Acm press, recommender systems, volume 40., 1997. URL http://doi.acm.org/10.1145/245108.245121.

[2] R. Mooney "P. Melville and R. Nagarajan.". "content-boosted collaborative filtering". *"Recomender System"*, 40, 2001. URL citeseer.ist.psu.edu/melville01contentboosted.html.

[3] Joseph A. Konstan Badrul M. Sarwar, George Karypis and John Reidl. Item-based collaborative filtering recommendation algorithms. 2001. URL citeseer.ist.psu.edu/sarwar01itembased.html.

[4] Fourth workshop on the evaluation of adaptive systems in conjunction with um'05. 2005. URL http://www.easy-hub.org/hub/workshops/um2005/challenge.html.

[5] T. Berners-Lee. Information management: A proposal. cern. world wide web consortium (w3c). 1989.

[6] N. Belkin and B. W. Croft. Information filtering and information retrieval: two sides of the same coin? *Recomender System*, 1992. URL http://doi.acm.org/10.1145/138859.138861.

[7] *"Recommender Systems Handbook"*. "Springer", 2011.

[8] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. *Recomender System*, 2004. URL citeseer.ist.psu.edu/article/massa04trustaware.html.

[9] Wikipedia. Vector space model, 2013. URL http://en.wikipedia.org/wiki/Vector_Space_Model.

[10] Gerard. Salton. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[11] Buckley Salton, Gerard and Chris. *Technical Report TR87-881.*

[12] Wikipedia. Mapreduce, 2013.

[13] The Apache Software Foundation. Mahout, 2014. URL https://mahout.apache.org/.

Appendix A

Technologies

A.1 Apache Hadoop

Apache Hadoop is an open-source framework for run applications on large cluster. Apache Hadoop processes large amounts of data by dividing data into independent small fragments and processing them individually over a distributed environment. Hadoop Distributed File System (HDFS) stores and retrieves data over a distributed environment. Apache Hadoop manages the cluster for any failures, nodes of the cluster and provides relentless highly available service.

Hadoop implements a computational paradigm known as MapReduce [[12]]. A majority of the research involved writing MapReduce programs in Hadoop framework

A.2 Mahout

Apache Software Foundation provides an open-source scalable machine learning library, known as Mahout[13]. Mahout has a set of predefined algorithms implemented in hadoop framework. Partial implementation of collaborative filtering by item-based recommendations was in this study for the recommendations approach based on user ratings.

A.3 Pig

Pig is a tool to create MapReduce programs using Hadoop using SQL approach. Pig scripts are written in Pig Latin script which provides syntax and semantics for processing or retrieving data through MapReduce Hadoop programs. Pig Latin scripts were written for this study to evaluate results of all approaches in order to calculate MAP.

A.4 Google GSON API

Google GSON API converts model objects to JSON strings and vice versa. Data from the website allrecipes.com is parsed and stored in model objects and these model objects are converted to JSON strings which are stored in a text file. The text file is the input file that is sent to the algorithm for processing.

A beocat cluster provided by CIS department of Kansas State University to process all my jobs for this study. A lot of shell scripts had to be written in order to submit jobs to beocat. Only Java programming language in Eclipse IDE was used to code this project.

Appendix B

Additional Experiments

An additional set of experiments were performed by dividing the dataset into different format of training and test folds. This chapter explains the dataset division and describes corresponding results.

The complete data set of reviews were divided into three independent folds of data. For each experiment, a pair of folds was given to an algorithm as a training set and the other fold was used as a test set to evaluate results. This division of train and test was conducted so the algorithm might process all data rather than a randomly generated fold.

Experiments were applied for all approaches described in Chapter 3 for a combination of the three folds using cosine and Pearson correlation coefficient similarity measures. A standard of ten (10) recommendations were used to evaluate results.

B.1 Algorithm Results Using Cosine Similarity over a 3-fold Dataset

MAP values of all algorithms with cosine similarity measure over three fold combinations is shown in Section B.1. Surprisingly, these results demonstrate improvement in the evaluation pattern with this kind of fold division on the dataset. However, when all approaches are compared, it follows identical results as mentioned in Chapter 6.

Table B.1: *Experiment: Evaluation of All approaches in three fold dataset by cosine similarity measure*

Test	Training		Recommendation based on			
TestSet	Training Sets		Ratings	Ingredients	Review Texts	Hybrid
Fold 1	Fold2	Fold3	0.0006588469	0.0001331084	0.0000542784	0.0000732641
Fold 2	Fold3	Fold1	0.0011485206	0.0001331084	0.0001209689	0.0001024186
Fold 3	Fold1	Fold2	0.0008088946	0.0001331084	0.0001213160	0.0001034884
Average of all Folds			0.0008720874	0.0001331084	0.0000988545	0.0000930570

B.2 Algorithm Results Using Pearson Correlation Coefficient Similarity over a 3-fold Dataset

MAP values of all algorithms with Pearson correlation coefficient similarity measure over three fold combination is shown in Section B.2. These results show similar kind of behavior as mentioned in Chapter 6 and consistently there is no variation of results based on number of recommendations is present. However, the Pearson correlation coefficient result and cosine coefficient result are in compliance with each other.

Table B.2: *Experiment: Evaluation of All approaches in three fold dataset by Pearson correlation similarity measure*

Test	Training		Recommendation based on			
TestSet	Training Sets		Ratings	Ingredients	Review Texts	Hybrid
Fold 1	Fold2	Fold3	0.0004931997	0.0001411597	0.0000879158	0.0000829122
Fold 2	Fold3	Fold1	0.0005721238	0.0000821126	0.0001312880	0.0001066105
Fold 3	Fold1	Fold2	0.0005218508	0.0000821126	0.0000000550	0.0000537270
Average of all Folds			0.0005290581	0.0001017950	0.0000730863	0.0000810832