A Thesis

on

Towards Efficient Recommender Systems Using Computational Intelligence

Submitted in fulfilment of the requirements for the award of the degree of

Doctor of Philosophy

by

GARIMA

(2K17/PhD/CO/05)

Under the supervision of

Prof. Rahul Katarya

Department of CSE, DTU, Delhi

Department of Computer Science and Engineering

Delhi Technological University

Delhi, India

2022

Dedicated to

My husband Gautam and My beloved Parents

CANDIDATE DECLARATION

I hereby declare that the thesis entitled "Towards Efficient Recommender Systems Using Computational Intelligence" submitted to Delhi Technological University, Delhi, in the partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy in the Department of Computer Science, is an original work and has been done by myself under the supervision of Prof. Rahul Katarya (Supervisor), Department of Computer Science and Engineering, Delhi Technological University, Delhi, India.

The interpretations presented are based on my study and understanding of the original texts. The work reported here has not been submitted to any other institute for the award of any other degree.

Garima
Roll No. 2K17/PhD/CO/05
Department of Computer Science and Engineering
Delhi Technological University
Delhi-110042, India

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
(Govt. of National Capital Territory of Delhi)
Shahbad Daulatpur, Main Bawana Road,
Delhi-110042, India

Date:_____

CERTIFICATE

This is to certify that the work incorporated in the thesis entitled "Towards Efficient Recommender Systems Using Computational Intelligence" submitted by Ms. Garima (Roll No. 2K17/PhD/CO/05) in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy, to the Delhi Technological University, Delhi, India is carried out by the candidate under my supervision and guidance at the Department of Computer Science and Engineering, Delhi Technological University, Delhi, India.

The results embodied in this thesis have not been presented to any other University or Institute for the award of any degree or diploma.

Prof. Rahul Katarya
Department of Computer Science and Engineering
Delhi Technological University
Delhi-110042, India

ACKNOWLEDGMENT

I address my sincere thanks to Almighty God for giving me the inner power to complete my thesis and guide me in every step of my life.

It is an immense pleasure to have the opportunity to express my heartiest gratitude to everyone who helped me throughout this research journey. With immense joy and heartfelt gratitude, I would like extend my indebtedness to my supervisor, Prof. Rahul Katarya (Dept. of Computer Science & Engineering), for his invaluable guidance, mentorship, encouragement, and patience. During the research, his motivation and encouragement have made me strive to work harder to achieve my goals. I am deeply humbled and indebted to my supervisor for continually motivating me to persevere and making me believe in myself during the times of hardships. His technical expertise, precise suggestions, kind nature, and detailed, timely discussions are wholeheartedly appreciated.

Also, my sincere thank goes to Delhi Technological University for considering my candidature for this course. I am also very thankful to Prof. Jai Prakash Saini, Vice-Chancellor, Delhi Technological University, Delhi, India, who has been a constant source of enthusiasm. He has always motivated young researchers like me to pursue excellence to achieve higher goals in academics and research. Also, my sincere thanks reciprocate to Dr. Vinod Kumar (HoD, Dept. of Computer Science and Engineering), Prof. Rajni Jindal (Chairperson DRC, Dept. of Computer Science and Engineering) for insightful comments and valuable suggestions. Special thanks to my seniors and colleagues of Delhi Technological University, Delhi, India. My sincere thanks to all the professors, faculty, researchers, and nonteaching staff of the Computer Science Department.

I also wish to take this opportunity to thank all my teachers who have taught me and shaped me into the person I am, aggravated me to be an academician, and have directly indirectly made me capable of succeeding in completing this research work. I am deeply thankful to all my colleagues and friends during my journey as a Ph.D. scholar. The engaging discussions, brainstorming sessions, and collaborative teamwork significantly impacted my growth as an independent researcher.

I would also like to thank my husband who always supported me in all my endeavors and believed in me and encouraged me in all the challenging times. Finally, but most importantly, I would like

to express my deepest gratitude to my parents who stood by me like a pillar of strength and always supported me to realize my goals. I will cherish their utmost love and blessings throughout my life.

Garima
(2K17/PhD/CO/05)
Department of Computer Science
Delhi Technological University,
Delhi-110042, India

ABSTRACT

Due to the nature of digital marketplaces, where the platforms are not bound by physical space, companies are adding more and more content and products and thus increasing the options available to customers manifold. To illustrate, Amazon today has around 12 million products available, and Netflix has approximately 6000 movies and shows on its platform. It is practically impossible for the user to scroll through millions of such options. This problem makes E-commerce shopping a daunting task. To solve the problem, recommender systems have emerged as an essential tool. Using recommender systems, the problem of item selection can be offloaded by implementing machine learning algorithms. These algorithms learn and predict what a user is likely to buy, hence reducing the set of products that the user has to go through to find an item that is relevant to his needs. Users often visit a platform just for window shopping, where they have no particular product or need in mind. In such a scenario, recommender systems become even more critical as they bear the entire burden of customer conversion. The system has to pick a comparatively much smaller subset of items to show the user from a much bigger list of products or services. If the system does not select relevant products, it loses potential sales despite having the right products. Thus, recommender systems are essential for working with a large pool of data. It considers the user's liking, previous purchase/like history, the social network of the user, and much more.

We faced the computational challenges of cold-start problem and data sparsity problem in implementing the recommender systems. We addressed these problems in our research work. Following are the objectives of this research work, the methodology we used to carry out these objectives which also construe the research contribution, and the results achieved after performing the research studies.

Objectives: The following four objectives have been charted out for this research study:

- To research and implement various methods and techniques of performing recommendations.
- To improve and optimize the results of recommendations by implementing evolutionary algorithms.

- To implement deep learning techniques for improving recommendations and design an algorithm which provides best results using these techniques.
- To develop a novel recommendation algorithm which could improve the accuracy of the recommendations while remediating the cold start problem and data sparsity problem.

Methodology: For achieving the mentioned objectives, this study utilizes machine learning and deep learning techniques like evolutionary algorithms, Neural Networks (NN), Natural Language Processing (NLP), and Topic Modeling approaches due to the tremendous applicability to solving the natural world problems. The following strategies are used to achieve the targeted objectives:

- For achieving the first objective, in this objective, the implementation of several benchmark machine learning and deep learning techniques in recommender systems are studied. An extensive literature survey is performed to understand and analyse the holistic application domains of recommender systems. A novel application of recommender systems in the domain of culinary science is carried out to generate efficient results.
- In the second objective, the results of recommender systems are gauged under the microscopic lens of evolutionary algorithm. To understand the proof of concept, ensemble learning is applied on benchmark machine learning techniques and the results are further optimized using Particle Swarm Optimization and after generating successful results, In another implementation of recommender systems, the results are optimized by the implementation of genetic algorithm.
- For the third objective, after performing a thorough literature survey and several implementations of recommender systems, it was deduced that deep learning produces the most optimum results. Hence, in this objective, several benchmark deep learning techniques were implemented in an ensemble learning setup to produce efficient results. In another implementation of deep learning, Recurrent Neural Network, and its application was implemented to improve the results.
- For the final objective, after performing the extensive survey and several implementations of recommender systems, it was discovered that recommender systems greatly suffer from the problem of cold-start and data sparsity. To remediate this problem, Recurrent Recommender Network, a spin-off of RNN, was implemented on real-world Point-of-Sale dataset to generate dynamic fruit recommendations.

Results: The following research outcomes were attained after performing this research study:

- An in-depth analysis of 120 research papers was performed that implemented deep learning techniques in recommender systems.
- A novel framework to solve two-fold recommendation problem of food-wine parings where novel features were extracted using text mining and sentiment analysis. Two novel datasets were created and compiled for the process of feature extraction and the results showed the resulting food-wine recommendations aligned with wine sommelier's food-wine recommendations
- An AutoML framework for Ensemble Learning Recommendations (EnPSO: Ensemble with Particle Swarm Optimization) was proposed and evolutionary algorithm Particle Swarm Optimization (PSO) for finding the best model was employed. The algorithm was analyzed on three publicly available benchmark MovieLens datasets and five benchmark machine learning techniques were implemented to create ensembles.
- A recommender system was proposed to implement AutoML framework for Ensemble Learning (En-DLR: Ensemble based Deep Learning Recommender). Genetic Algorithm was employed to identify the most optimal model in the search space and four benchmark deep learning techniques were implemented as base recommenders.
- A dynamic recommender system was implemented to dynamically incorporate the temporal changes in fruit seasonality variations and user preferences using deep learning based LSTM network.
- Alleviated the problem of data sparsity with the implementation of Recurrent Recommender Network.
- A real-world Point-Of-Sale dataset of a commercial fruit retailer was used for implementing the system.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AE	Autoencoder
AutoML	Automated Machine Learning
CF	Collaborative Filtering
CBOW	Continuous Bag-of-Words
CDL	Collaborative Deep Learning
CNN	Convolution Neural Network
DBN	Deep Belief Network
DCF	Deep Collaborative Filtering
DeepFM	Deep Factorization Machines
DL	Deep Learning
En-DLR	Ensemble Deep Learning Recommender
EnPSO	Ensemble Particle Swarm Optimization
ERM	Ensemble Recommendation Method
GRU4REC	Gated Recurrent Unit for Recommender System
IBCF	Item-Based Collaborative Filtering
IDF	Inverse Document Frequency
LSTM	Long Short-term Memory
MAE	Mean Absolute Error
MDP	Markov Decision Process
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Square Error
MV-DNN	Multi-View Deep Neural Network
NCF	Neural Collaborative Filtering
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing

NMF	Neural Matrix Factorization
NN	Neural Network
POS	Point-of-Sale
PSO	Particle Swarm Optimization
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RRN	Recurrent Recommender Network
RS	Recommender Systems
SDAE	Stacked Denoising Autoencoder
SVD	Singular Value Decomposition
TF	Term Frequency
UBCF	User-Based Collaborative Filtering
VAE	Variational Auto Encoders

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Recommender systems use machine learning and artificial intelligence algorithms to predict items to users. In today's world, the availability of such a vast pool of data makes it difficult for e-commerce websites to identify user-centric items [1]. If a user intends to select an item, he might not be able to find the item most suitable to him due to the presence of multiple options. To make this task easy, the concept of recommender systems was introduced. Recommender Systems use machine learning and artificial intelligence techniques to recommend relevant items to the user. Recommender systems are compounding to be more crucial than ever as the data available online is increasing manifold. The increasing data bestows us with an opportunity to create complex systems that can model the user interactions more precisely and extract intricate features to gneerate recommendations with improved accuracy. To create such complex models, deep learning is emerging as one of the most powerful tools. It can process large amounts of data to learn the structure and patterns that can be exploited further. It has been used in recommender systems to solve cold-start problem, better estimate the interaction functions, and extract deep feature representations, among other facets that plague the conventional recommender systems. As big data is becoming more prevalent, there is emerging an urgency to utilize tools and techniques that can make the best out of such explosive data.

Section 1.1 discusses and explains about the recommender systems. Section 1.2 elaborates upon the types of machine learning techniques and types of recommender systems. Section 1.3 containes the motivation of the study, whereas, Section 1.4 discusses in detail the research questions, and the research objectives. Section 1.5 describes the outline of the thesis and the chapter concludes with the summary in Section 1.6.

1.1. Recommender Systems (RS)

Due to the nature of digital marketplaces, where the platforms are not bound by physical space, companies are adding more and more content and products and thus increasing the options available to customers manifold. To illustrate, Amazon today has around 12 million products

available, and Netflix has approximately 6000 movies and shows on its platform. It is practically impossible for the user to scroll through millions of such options. This problem makes E-commerce shopping a daunting task. To solve the problem, recommender systems have emerged as an essential tool. Using recommender systems, the problem of item selection can be offloaded by implementing machine learning algorithms. These algorithms learn and predict what a user is likely to buy, hence reducing the set of products that the user has to go through to find an item that is relevant to his needs. Users often visit a platform just for window shopping, where they have no particular product or need in mind. In such a scenario, recommender systems become even more critical as they bear the entire burden of customer conversion. The system has to pick a comparatively much smaller subset of items to show the user from a much bigger list of products or services. If the system does not select relevant products, it loses potential sales despite having the right products. Thus, recommender systems are essential for working with a large pool of data. It considers the user's liking, previous purchase/like history, the social network of the user, and much more. After considering such parameters, the recommender system makes a systematic and sound list of products tailored to the user's likes. The recommendations depend upon several parameters and are divided the recommender systems into three types, i.e., content based, collaborative filtering based, and hybrid based. Depending upon the type of recommender system used, the process of generating recommendations vary.

The main objective of a recommender system is to study, analyse and understand the behaviour of user's past interactions which can be modelled into future likelihood of conversions. A recommender system's goal is to present items to a user, which are most likely to lead to conversion. Conversion might relate to different things in different contexts. For example, for e-commerce, it might mean purchasing the product, and for Netflix, it might mean viewing the content. To achieve this goal, recommender systems have to study the underlying data, consisting of items, users, and their interactions. To study these interactions, there is a need to extract relevant features and create a system that can learn and model such interactions. Recommender systems have become a core part of the retail experience. Retailers often rely on recommender systems to help them drive more conversions through targeted communication and advertisements. However, recommender systems are not one size fits all. Specialized retailers require specialized

recommender systems to consider various features, attributes, and dynamics about the product category.

1.2. Types of Recommender Systems

Recommender Systems use machine learning and artificial intelligence algorithms to predict items to users. In today's world, the availability of such a vast pool of data makes it difficult for e-commerce websites to identify user-centric items. If a user intends to select an item, he might not be able to find the item most suitable to him due to the presence of multiple options. To make this task easy, the concept of recommender systems was introduced. RS uses machine learning and artificial intelligence techniques to recommend relevant items to the user. The recommendations can be made in three ways, i.e., content-based filtering, collaborative filtering, and hybrid filtering, as shown in Figure 1.1.

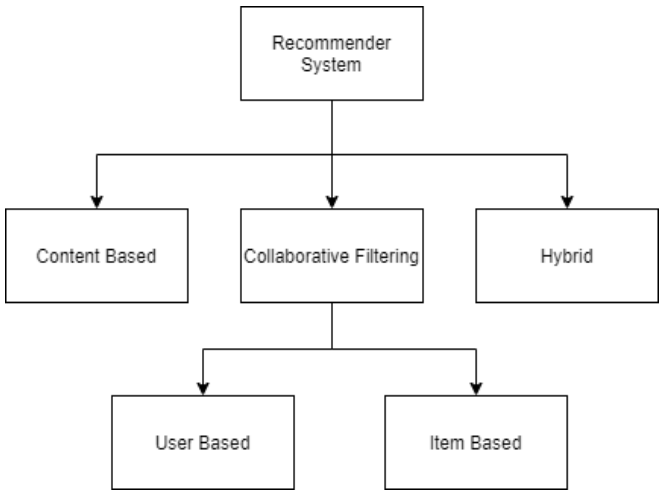


Figure 1.1 Types of Recommender Systems

To understand these, we first need to understand the different types of Machine Learning techniques.

1.2.1. Machine Learning Techniques

Machine learning is a section in computer science which deals with intelligently computing and solving problems and analyzing the results obtained. The system is presented with a dataset, and various machine learning algorithms can be applied to that dataset to obtain results. Due to the

availability of such extensive data and the need to get predictive results, machine learning algorithms are used widely. There exist three types of machine learning techniques, which can be seen in Figure 1.2.

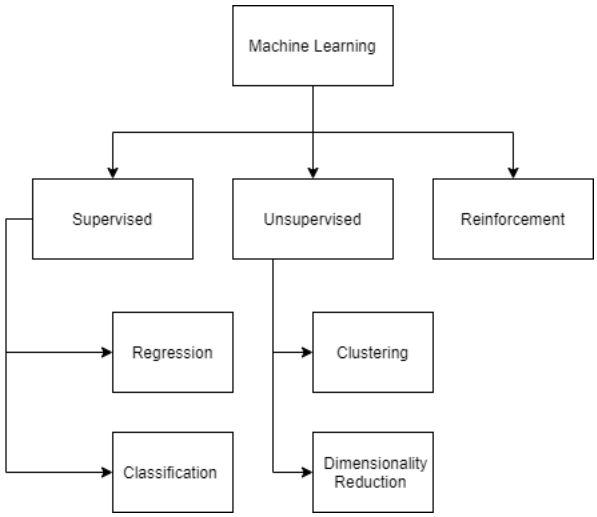


Figure 1.2 Machine Learning Architecture

1.2.1.1 Supervised Learning

In supervised machine learning algorithms, the system is presented with the dataset, and the outcome is pre-defined. For example, it is known that the outcome has to be whether a given transaction is fraudulent or genuine for identifying fraudulent bank transactions. Since the result is known beforehand, the algorithm works to guide the outcome towards the expected result [2]. Hence, the entire working is supervised. A few examples of supervised learning algorithms are Logistic Regression, Neural Network, Naïve Bayes, Decision Trees, Nearest Neighbors, and others. Semi-supervised learning is another type of technique in machine learning [3]. This technique is used when the expected result is known only for a few data points, i.e., when in a given dataset, not every data point is labeled. It helps in identifying and learning the structure of the input variables.

1.2.1.2. Unsupervised Learning

The second type of technique is unsupervised learning. In this type of technique, the data is not classified. The expected results are not known before its implementation. The system is presented with the dataset, and any of the unsupervised machine learning algorithms are applied to it, and

the results are obtained. Since there is no supervision on the desired results, this technique is called unsupervised learning [4]. The main motive behind using this technique is to identify hidden clusters and patterns in the data. Examples of such algorithms are K-means clustering, self-organizing maps (SOM), hierarchical clustering, and others.

1.2.1.3. Reinforcement Learning

In reinforcement learning, the algorithm works on the concept of rewards and penalty. None of the data points in the dataset are labeled, i.e., the expected output is unknown, but the learning takes place in such a way that the system learns the environment on-the-go. The system is given an environment, and a set of actions are defined. Depending on the type of action the system takes, it is rewarded [5]. If the system's action takes it towards the goal state, it is rewarded. However, if the system's action takes it away from the goal state, it is charged with a penalty. To carry out this task, an objective function is determined, including all the possible action and state spaces. The main aim is to maximize this objective function, and policy is defined, which is to be followed. The most widely accepted and real-life example of reinforcement learning is how a toddler learns about the positive and negative outcomes of his actions by experiencing the rewards and penalties of those actions. Machine learning algorithms can be deployed in implementing recommender systems. Based on the type of recommender system being used, the data is collected. In collaborative filtering, the data is collected by performing social network analysis and identifying the items bought or liked by neighbors of the user. These data points or choices serve as the dataset for implementing the algorithm. If content-based filtering is used, the user's past choices are identified as the data points. For example, for a movie recommender system, the movies user has watched in the past, the genre, director, actors, etc. are taken into account, and these bits of information constitute the dataset. Further, algorithms are implemented on this dataset to recommend items. Lastly, hybrid filtering can be used for collating data using both supervised and unsupervised learning and presenting with the final data points to the system.

1.2.1.3.1. MDP Model

In Reinforcement Learning, which is a machine learning technique, an agent learns the optimal behaviour in a given environment through rewards and punishment (negative reward). The agent has to learn optimal action to perform, given the state he is in. When this process is repeated over

and over again, the problem is known as Markov Decision Process (MDP). MDP is a framework that helps in deciding a set of actions to maximize reward, given the current state is in a stochastic environment. It works on the assumption that Markov property is satisfied, which means that effect of a particular action in a specific state depends only on that state and not the previous states [6].

- *S*: State space, set of all the possible states
- *A*: Action space, set of all possible actions that can be taken by an agent
- *R(S, A)*: Reward function, gives a reward for a given state S and action A.
- *T*: Transition, This tells the effect of an action on the state. T can be Deterministic or Non-Deterministic,

Deterministic means that a new state can be determined for given state S and action A as given in Equation 1.1,

$$T: S \times A \rightarrow S \tag{1.1}$$

Non Deterministic means that given an action and an initial state, the change of state is defined by probability distribution i.e. results of actions are probabilistic in nature.

$$T: S \times A \rightarrow Prob\ (S) \tag{1.2}$$

The above function, as shown by Equation 1.2, represents the probability distribution P(S|S, A)

A policy in MDP is represented by π which is a mapping from *S* to *A*. It basically maps an action *A* to be taken for every state *S*. To find the optimal set of actions to perform, there occurs a need to find a policy such that it maximizes the chosen objective function. An objective function maps an infinite sequence of rewards to a real number.

1.2.2. Types of Recommender Systems

Recommender Systems use machine learning and artificial intelligence algorithms to predict items to users. In today's world, the availability of such a vast pool of data makes it difficult for e-commerce websites to identify user-centric items [1]. If a user intends to select an item, he might

not be able to find the item most suitable to him due to the presence of multiple options. To make this task easy, the concept of recommender systems was introduced. RS uses machine learning and artificial intelligence techniques to recommend relevant items to the user.

1.2.2.1. Content-Based Filtering

In Content-Based Filtering, the recommendations are performed based on the previous choices of the user. As the name suggests, this type of filtering technique depends upon the various parameters of the user items. These parameters, when taken into account, reveal the granular level information about the user's preferences. Content-based filtering aims to exploit such a detailed analysis of the user's liking and recommends items in the present. The Recommender System studies the choices user made in the past in different domains and depending upon those choices, the user is recommended items [7]. Such choices made by the user depend on various parameters or features of the items. Since these recommendations match the preferences the user made in the past, hence they are more user-centric. To explain with the help of an example of content-based recommendations, let us say that a user likes to watch movies by the director *Quentin Tarantino*, so the movies which will be recommended to the user will feature the same director. This was content-based filtering based on just one feature. Similarly, multiple features can be taken to determine the recommendation list. Mathematically, content-based RS for multiple users can be represented using the following Equation 1.3:

$$min_{\theta^1 \ldots . \theta^n} \frac{1}{2} \sum_{j=1}^{n} \sum_{i:r(i,j)=1} \left(\left(\theta^j\right)^T x^i - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \left(\theta_k^j\right)^2 \quad (1.3)$$

Here, θ^j represents the parameter vector for user *j*

r(i,j)=1 if user *j* has rated the movie *i*, else 0,

x_i represents the feature vector for movie *i*, and

$y^{(i,j)}$ represents the rating for movie *i* by user *j*

1.2.2.2. Collaborative Filtering

The second type of RS is Collaborative Filtering. In this type of filtering, the recommendations are affected by the neighbors of the user. It is seen that if a user's friend buys a particular product,

the user is likely to buy the same or a similar product. Based on this principle, collaborative filtering adopts a neighborhood-based approach. In this technique, the user's neighbors' choices are studied and based on those choices, similar items are recommended to the user. To explain with an example of how collaborative filtering works, let us say that user *a* has a neighbor user *b*. User *b* likes a particular product, say product *p*. Now in this filtering technique, the preference of user *a* will be subjected to the preferences of user *b* and vice-versa. Hence, user *a* will be recommended product *p* and products similar to product *p* [8]. The recommendations depend upon several parameters and are divided the recommender systems into three types, i.e., content based, collaborative filtering based, and hybrid based. Apart from these broad categories, several other techniques are used to build a recommender engine, depending upon the primary parameters taken into account. Collaborative Filtering is further of two types, namely, item-based collaborative filtering and user-based collaborative filtering.

1.2.2.2.1. Item-based Collaborative Filtering

In this type of collaborative filtering technique, the recommendations are based on the similarity between the items. If a user purchases product *p* and product *q* is similar to product *p* in one or more ways, i.e., if they share some common features, the user is recommended product *q* and other products similar to product *q*. The similarity between the items is calculated using several similarity measures. It can be cosine-based similarity, which can be calculated using the following Equation 1.4. It is also known as vector-based similarity, and the similarity is determined by calculating the cosine of the angle between the two vectors [9].

$$sim(a,b) = \cos\left(\vec{a}\,,\vec{b}\right) = \frac{\vec{a}.\vec{b}}{\| \vec{a} \| * \| \vec{b} \|} \tag{1.4}$$

Here, $\vec{a}$ and $\vec{b}$ are the two item vectors.

Another similarity measure that can be used to deduce the similarity between different items is Pearson correlation-based similarity. It can be given by Equation 1.5.

$$sim(a,b) = \frac{\Sigma_{u\in U}\left(R_{u,a} - \bar{R}_a\right)\left(R_{u,b} - \bar{R}_b\right)}{\sqrt{\Sigma_{u\in U}\left(R_{u,a} - \bar{R}_a\right)^2}\sqrt{\Sigma_{u\in U}\left(R_{u,b} - \bar{R}_b\right)^2}} \quad (1.5)$$

Here,

$R_{u,a}$ denotes the rating given by user *u* for item *a*,

$\bar{R}_a$ denotes the average rating of item *a*,

Similarly, $R_{u,b}$ denotes the rating given by user *u* for item *b*, and

$\bar{R}_b$ denotes the average rating of item *b*

1.2.2.2.2. User-based Collaborative Filtering

In this type of collaborative filtering technique, the recommendations take place depending upon the preferences of the user's neighbors. If user *u* has a neighbor user *v*, and user *v* likes product *p*, then user *u* will be recommended product *p* and all such products preferred by user *v* [10].

1.2.2.3. Hybrid Filtering

The third approach is the hybrid approach. It is an amalgamation of content-based filtering and collaborative filtering. Here, first, the social network analysis of a user's neighborhood takes place. This step helps identify the neighbors' preferences, and items are kept in the account to be recommended to the user. In the second step, the user's history is studied, and depending on the items he bought previously and items having similar content or attributes as the past items; item recommendations are taken into account [11]. Finally, analyzing the results obtained in both the steps, the user is recommended the items. Authors in [2] differentiated user's preferences in generated recommendations by using a deep hybrid recommender system. In order to explain the underlying literature of hybrid filtering, let us take the example used in the previous section. Let us say that product *p* has a similar product, product *q*, along with one or several features. The system will apply content-based filtering to identify products similar to product *p* and product *q*. Upon selecting such products, say *list l*, the system will use collaborative filtering, and user *b* will now be recommended product *p*, product *q*, and other similar products, i.e., products from *list l*. Any of the three techniques can be used to recommend items to users. Depending upon the chosen

technique, the recommendations are made to the user. Hybrid Recommender Systems can be categorized into the following types:

1.2.2.3.1. Monolithic Hybrid Design

In this type, there exists a single recommendation component that aggregates multiple recommendation methods by pre-processing and combining numerous knowledge sources [12]. The architecture of monolithic hybrid design has been represented in Figure 1.3.

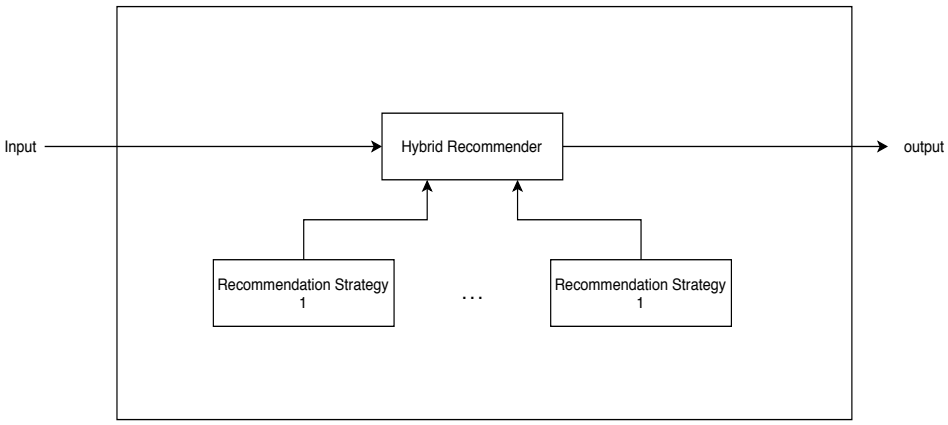


Figure 1.3 Monolithic Hybrid Design

1.2.2.3.2. Parallelized Hybrid Design

In this type, several recommender systems are run in parallel, and the output of each is combined at the later stages using an aggregation mechanism. These are further of three types, i.e., mixed, weighted, and switching [13]. The architecture of parallelized hybrid design has been shown in Figure 1.4.

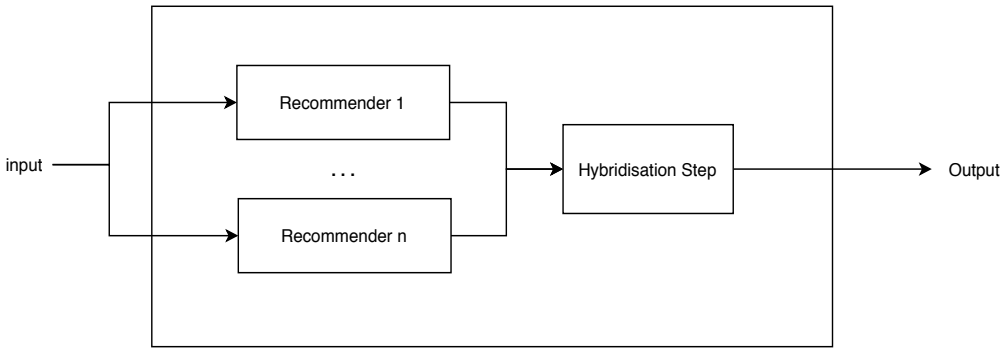


Figure 1.4 Architecture of parallelized hybrid Design

1.2.2.3.3. Pipelined Hybrid Design

In this type, every recommender system processes the input pertaining to its recommendation mechanism. The output, hence produced, is forwarded as input to subsequent recommender systems mimicking a staged process [14]. The architecture of the pipelined hybrid design is given in Figure 1.5.

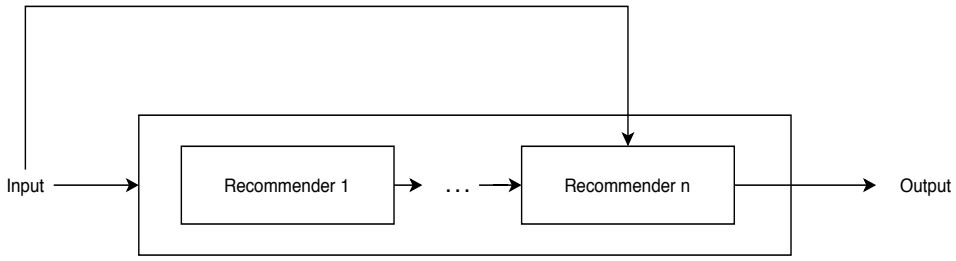


Figure 1.5 Architecture of Pipelined Hybrid Design

1.3. Motivation of Study

Recommender systems have become a primary cogwheel in most of the computational applications today. In today's world, as more and more people are connecting to the internet and using its benefits, the amount of data is increasing exponentially. In earlier times, data was managed manually and whenever required, the information was retrieved. But now due to such large availability of data, it has become a necessity to automate the process by using machines. For this task, the concept of machine learning is used where various machine learning algorithms are applied on datasets to perform the desired operations. Users often get overwhelmed with such tremendous amount of data and fail to decide what information will be suitable for them in a particular domain. So, it is important to have techniques which could let users know what information they will prefer over others pertaining to an area. The concept that is being talked about here is the Recommender Systems. These are the algorithms which when applied to a dataset, read user's choices and intelligently recommend items to them based on their past choices. There are various applications of recommender systems like if a user wishes to buy a product online [1]; there are thousands of options he could choose from. The recommender system suggests the user with products they might like on the basis of their previous choices [15]. Recommender systems have become a core part of the retail experience. Retailers often rely on recommender systems to

help them drive more conversions through targeted communication and advertisements. However, recommender systems are not one size fits all. Specialized retailers require specialized recommender systems to consider various features, attributes, and dynamics about the product category.

The performance and efficiency of a recommender system are defined by how well it can recommend items to the users. The suggested items should draw the users into buying the products and give reviews. The efficiency of a recommender system is affected by various parameters like the type of model used, type of dataset, the domain of application, type of filtering algorithm used, shopping frequency of the user, social network of the user, etc. There are various ways by which the recommendations can be improved. One is by observing the previous purchases made by the user. Even the clicks made by the user and the products he viewed are taken into account. The other way is observing let's say, the shopping trends of the people user is in contact with It is likely for the user to get influenced by the products his friends have bought. This concept does not hold true just for online shopping, there are various other applications like book recommendations in a library [16], recommendations in E-learning portals [17], tourism recommendation [15], movie recommendations [18], etc.

1.4. Research Objectives

After performing a comprehensive review of recommender system and its techniques, the following research questions surfaced:

- **RQ1**: What are the different machine learning and deep learning techniques employed in recommender systems?
- **RQ2**: How can the use of sentiment analysis contribute towards generating efficient recommendations?
- **RQ3**: What is the significance of ensemble learning in recommender systems?
- **RQ4**: How do evolutionary algorithm affect the recommender system generated predictions?
- **RQ5**: How can the problem of cold start and data sparsity be remediated in recommender systems?

To answer the above-mentioned research questions and to focus on understanding and implementing the various applications of recommender system to analyse its capabilities, following research objectives were chalked out:

Research Objective 1: To research and implement various methods and techniques of performing recommendations.

Research Objective 2: To improve and optimize the results of recommendations by implementing evolutionary algorithms.

Research Objective 3: To implement deep learning techniques for improving recommendations and design an algorithm which provides best results using these techniques.

Research Objective 4: To develop a novel recommendation algorithm which could improve the accuracy of the recommendations while remediating the cold start problem and data sparsity problem.

The detailed explanation of the research objectives has been listed as follows:

Research Objective 1: In this objective, the implementation of several benchmark machine learning and deep learning techniques in recommender systems are studied. An extensive literature survey is performed to understand and analyse the holistic application domains of recommender systems. A novel application of recommender systems in the domain of culinary science is carried out to generate efficient results.

Research Objective 2: In this objective, the results of recommender systems are gauged under the microscopic lens of evolutionary algorithm. To understand the proof of concept, ensemble learning is applied on benchmark machine learning techniques and the results are further optimized using Particle Swarm Optimization and after generating successful results, In another implementation of recommender systems, the results are optimized by the implementation of genetic algorithm.

Research Objective 3: After performing a thorough literature survey and several implementations of recommender systems, it was deduced that deep learning produces the most optimum results. Hence, in this objective, several benchmark deep learning techniques were implemented in an ensemble learning setup to produce efficient results. In another implementation of deep learning, Recurrent Neural Network, and its application was implemented to improve the results.

Research Objective 4: After performing the extensive survey and several implementations of recommender systems, it was discovered that recommender systems greatly suffer from the problem of cold-start and data sparsity. To remediate this problem, Recurrent Recommender Network, a spin-off of RNN, was implemented on real-world Point-of-Sale dataset to generate dynamic fruit recommendations.

To answer the research questions and fulfil the research objectives, the following research studies were performed as shown in Table 1.1:

Table 1.1 Aligning Research Questions, Research Objectives and Publications

ROs	RQs	Publication(s)
RO1	**RQ1** **RQ2**	Gupta, G., & Katarya, R. (2021). Research on understanding the effect of deep learning on user preferences. Arabian Journal for Science and Engineering, 46(4), 3247-3286. **[Published, SCIE, IF: 2.3]** Garima Gupta, Rahul Katarya; A Computational Approach Towards Food-Wine Recommendations. **[Submitted in SCIE journal]** Gupta, G., & Katarya, R. (2018, June). A study of recommender systems using Markov decision process. In 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 1279-1283). **IEEE**. Gupta, G., & Katarya, R. (2019, November). Recommendation analysis on item-based and user-based collaborative filtering. In 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT) (pp. 1-4). **IEEE**.

RO2	**RQ3** **RQ4**	Gupta, G., & Katarya, R. (2021). EnPSO: An AutoML technique for generating ensemble recommender system. Arabian Journal for Science and Engineering, 46(9), 8677-8695. **[Published, SCIE, IF: 2.3]** Gupta, G., & Katarya, R. , En-DLR: Generating Recommendations with AutoML and Deep Learning **(Communicated)**
RO3	**RQ1** **RQ3** **RQ4**	Gupta, G., & Katarya, R. , En-DLR: Generating Recommendations with AutoML and Deep Learning **(Communicated)** Gupta, G., & Katarya, R. (2021, May). A study of deep reinforcement learning based recommender systems. In 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC) (pp. 218-220). **IEEE**.
RO4	**RQ1** **RQ5**	Gupta, G., & Katarya, R. , A Novel Approach To Alleviate Data Sparsity And Generate Dynamic Fruit Recommendations From Point-Of-Sale Data **(Major Revision, SCIE, IF: 1.5)**

1.5. Outline of the Thesis

The thesis comprises of seven chapers and a detailed summary of all the chapters have been summarised below:

Chapter 2: In this chapter, a thorough literature review has been performed in the field of recommender systems using computational intelligence. Here we present a detailed understanding of state-of-the-art computational intelligence techniques used for employing recommender systems. This chapter also entails the various parameters across which the recommnder systems have been categorized and analysed.

Chapter 3: After identifying the research gaps from the syrvey, this chapter presents a food-wine recommender system that uses a novel framework designed to tackle the problem of handling two layers of recommendation. The underlying data used by this recommender system is derived by text mining and sentiment analysis operations on Yummly and Winemag datasets used in

conjuncture with two datasets that are self-created. Computational Intelligence techniques like word2vec allowed the system to model abstract features required for the recommender system. A system is presented that was able to apply the principles of food-wine pairing to thousands of dishes and wines to instantly generate pairings consistent with the principles while considering user preference.

Chapter 4: It was deduced from the previous studies and implementation, that optimizing the results is crucial in recommender systems and ensemble elarning is a great tool for that task. Hence, in this chapter, we implemented EnPSO (Ensemble Particle Swarm Optimization), an ensemble learning technique that further optimized the recommendation results using Particle Swarm Optimization. We used the benchmark MovieLens dataset for its implementation.

Chapter 5: This work proposes an AutoML (Automated Machine Learning) framework wherein the algorithm will optimize and pick out a well-performing ensemble from the search space by giving the base recommender. The framework En-DLR (Ensemble Deep Learning Recommender) can be extended to any domain, any number of base algorithms, and customized with various ensemble techniques in the framework. The framework optimizes a hierarchical ensemble and returns a well-performing ensemble from the search space. Genetic algorithm is used for our optimization flow. With this technique, not only a well-performing recommender system can be created but also adding a new recommendation technique to the framework is algorithmically trivial.

Chapter 6: A novel fruit recommender system is proposed that generates dynamic recommendations while remediating the problem of data sparsity. A novel fruit recommender system is developed that considers the temporal dynamics in the fruit market, like price fluctuations, fruit seasonality, and quality variations that occur throughout the year. To perform this task, Recurrent Recommender Network (RRN) is used which uses the deep learning method Long Short-Term Memory (LSTM) to implement the system model. To ensure that the work and results obtained are practical, the system has worked in a real-world setting, by tying up with a specialty fruit retailer based in New Delhi to get the real-world Point-of-Sale (POS) data of

consumers. The result of the study suggests the proposed algorithm performs better than other benchmark algorithms along NDCG and RMSE metrics.

Chapter 7: This chapter presents a comprehensive summary of the research work done. It includes the research summary of the work done, and the limitations of the research study identified. The chapter also presents the future aspects of the research work performed and how the study can help the future researchers in the said domain.

1.6. Chapter summary

This chapter covers the overview of the recommender systems, its types and the computational intelligence techniques which are employed to implement such recommender systems. This chapter discusses the motivation of the study and the research objectives of the research work. It also comprises of a brief summarization of the thesis chapters to give readers an overview of the research work performed.

Chapter 2

METHODICAL LITERATURE REVIEW

In this study, various parameters were considered for identifying the relevant research papers to be included in the study. The broad area of consideration was deep learning in recommender systems. In this section, the research methodology adopted in this study has been explored. Broadly, the research methods have been based upon ten attributes, i.e., publisher of the paper, year of publication, number of citations, and location of performing the study.

In this chapter, a thoroughly extensive literature survey in the field of recommender systems has been performed. Section 2.1 discusses the several application domains of recommender systems. Section 2.2 describes the major benchmark and self-compiled datasets used in various im plementations of recommender systems. Section 2.3 enlists different studies working in different types of recommender systems. Section 2.4 gives an overview of the deep learning techniques employed in recommender systems. Section 2.5 enlists various metrics used in several research studies and Section 2.6 discusses the literature review in detail where, Section 2.7 describe sthe different deep learning techniques in detail. The chapter concludes with the summary in Section 2.8.

2.1. Application Domain

There is a need for a dataset to apply machine learning or artificial intelligence algorithms to implement a recommender system. The dataset depends on the application domain of the recommender system. Table 2.1 presents all the application domains of the referred papers and several studies performed in that domain.

Table 2.1 Application domains of studies

Application Domain	Studies
Item recommendation	[19], [20], [21], [22], [23], [24], [25], [26], [27], [28]
News Recommendation	[29], [30], [31], [32], [33], [34]
Movie recommendation	[35], [36], [37], [38], [39], [40]
Cold-Start Problem	[41], [42], [43], [44], [45]
Session-based recommendation	[46], [47], [48], [49], [50]
Music Recommendation	[51], [52], [53], [54]
Text Recommendation	[55], [56], [57], [58], [59]
Image Recommendations	[60], [61], [62]
social network-based recommendation	[63], [64], [65]
Hashtag recommendation	[66], [67], [68]
POI recommendation	[69], [70], [71]
Citation Recommendation	[57], [72], [73]
Content-based Recommendations	[73], [74], [75]
Video recommendation	[76], [77]
Quote recommendation	[78], [79]

E-Learning Recommendation	[80], [81]
Rating prediction	[82], [56]
Job recommendation	[42], [83]
Fashion Recommendation	[84], [85]
Blog Recommendation	[86]
Venue Recommendation	[87]
Consumer Preferences	[88]
Co-evolutionary Latent Feature Processes	[89]
Artwork Recommendation	[90]
Adaptive user-interfaces	[91]
Audience Activity Recommendation	[92]
Data Sparsity	[93]
Healthcare service recommendation	[94]
Treatment recommendations	[95]

It can be observed from the table that maximum work has been done in the domain of item recommendations, news recommendations, and movie recommendations.

2.2. Datasets

In this section, the datasets used in the research papers have been listed. In some papers, authors used publicly available datasets to perform the recommendation process using their proposed model. In other papers, authors scraped data from websites or applications using APIs or manually collect data for implementing recommender systems. Table 2.2 enlists the external datasets used in previous studies, and Table 2.3 enlists self-generated datasets.

Table 2.2 Dataset from external sources

Dataset	Studies
MovieLens	[24], [27], [32], [35], [36], [37], [38], [96], [57], [73], [56], [88], [92], [97], [89], [98], [98],[99], [100], [81], [101], [102]
Amazon	[19], [21], [24], [28], [45], [57], [82], [56], [93], [100], [103], [104], [105], [106]
Yelp	[21], [24], [25], [28], [69], [70], [82], [89], [93], [104]
IMDb	[38], [107], [44], [56], [101], [50]
NetFlix	[108], [107], [44], [76], [101]
CiteULike	[108], [58], [109], [110], [111]
Epinions	[61], [64], [94], [99]
Last.fm	[22], [50], [67], [101]
Delicious	[43], [67], [98], [101]
Million Song Dataset	[52], [53], [54], [106]
BookCrossing	[92], [106], [102]
FourSquare	[69], [70], [87]
The Echo Nest Taste Profile Subset	[51], [53]
Dbbook	[55]
Wikiquote website	[78], [79]
Flixter	[63], [65]

Ciao	[64], [65]
YooChoose	[74], [106]
FilmTrust	[65], [92]
Twitter	[26]
Google News	[66]
Tumblr	[86]
Flickr	[87]
Picasa	[87]
Oxford Concise Dictionary of Proverbs	[78]
Google Play	[112]
IPTV	[89]
YouTube	[77]
UGallery	[90]
Rossmann	[98]
Frappe	[113]
CLEF NewsREEL	[31]
Penn Treebank	[107]
CADE web directory	[106]
RefSeer	[57]
Reddit	[50]
Economics thesaurus (STW)	[72]
EconBizRecSys evaluation dataset	[72]
Jester	[23]
PubMed	[73]
EqGraph	[91]

MSWeb	[91]
Assistments	[91]
Beer	[24]
Douban	[99]
NAVER News	[33]
Instagram	[84]
Zalando	[84]
Trip.com	[26]
Facebook	[26]
Exact Street2Shop	[85]
Taobao advertising dataset	[27]
Meetup	[101]
DeepSurv	[95]

It is evident from Table 2.3 that the MovieLens dataset is extensively used for analyzing the effectiveness of implementing deep learning in recommender systems. Other widely used datasets are Amazon and Yelp.

Table 2.3 Self-generated dataset

Self-Generated Dataset	Studies
1. Google Chrome Extension "Daum News Tracker," 2. Android application "KECI News.	[30]
1. Logs scraped from search engine Bing Web vertical 2 News article browsing history from Bing News vertical 3 App download logs from Windows AppStore 4 Movie/TV view logs from Xbox.	[20]
User's news click history between 04/01/2014 and 09/30/2014.	[30]
The images and users' information in this dataset were crawled from Flickr through its API	[60]
1. VIDXL - was collected over a 2-month period from a video site	[47]

2. CLASS - product view events of an online classified site.	
"starc" platform which is based on Open edx for self-development which serves the fundamental education field	[80]
Collected from real life users' browsing history of an online European department store over two weeks at the beginning of January 2016.	[48]
Scraped Ukiyo-e images from the "the Ukiyo-e search service" and the Ukiyo-e pages of Ritsumeikan University Art Research Center	[61]
Wanted and missing persons' application of the Police of the Czech Republic.	[62]
Collected 419,509 check-in records published by 49,823 users among 18,899 locations from August 2012 to July 2013 in Manhattan via the API of Foursquare	[71]
A large collection of quality article selection demonstration with an average length of 900 characters over six months, manually created by professional editors.	[59]
1314 resumes which came in as a part of summer research intern application at IBM Research Labs	[83]
Sampled offline dataset collected from a commercial news recommendation application	[114]

2.3. Types of Recommender Systems

There are various types of recommender systems, depending on the recommendation technique used. In this section, the types of recommender systems used in the studied papers have been listed along with the studies they were used in.

Table 2.4 Types of Recommender Systems

Type of Recommender System	Studies
Content Based	[20], [22], [29], [30], [33], [34], [36], [39], [47], [54], [54], [59], [62], [73], [78], [79], [80], [85], [86], [87], [95], [105], [106]
Collaborative Filtering	[19], [24], [26], [27], [28], [31], [32], [35], [40], [107], [43], [44], [48], [57], [58], [61], [63], [64], [65], [67], [68], [69], [70], [76], [77], [81], [82], [42], [88], [91], [94], [109], [98], [112], [110], [99], [100], [99], [101], [104], [50], [111]
Hybrid	[108], [25], [37], [38], [51], [60], [75], [84], [102]
Context-Aware	[41], [71], [115], [74], [56], [92], [93], [103], [112]

It is evident from Table 2.4 that most of the studies have been done in content-based and collaborative filtering algorithms. This presents the readers an opportunity to work in other hybrids, context-aware, and other similar recommendation techniques.

2.4. Metrics

The result obtained by implementing the recommender systems is analyzed by using various metrics. Different papers deploy different metrics to get a multi-dimensional interpretation of the results. Table 2.5 enlists the metrics used in the papers. Figure 2.1 graphically analyzes the metrics used.

Table 2.5 Analysis Metrics used in the papers

	Metric
A	Time
B	Accuracy
C	Mean Square Error (MSE)
D	Root Mean Square Error (RMSE)
E	Recall
F	Mean Average Precision (MAP)
G	Precision
H	Mean Reciprocal Rank (MRR)
I	Mean Absolute Error (MAE)
J	Rank Score
K	Area Under Curve (AUC)
L	Normalized Discounted Cumulative Gain (nDCG)
M	Pearson Correlation
N	Hit Ratio
O	F1-score
P	Mean Average Rank (MAR)
Q	Coverage

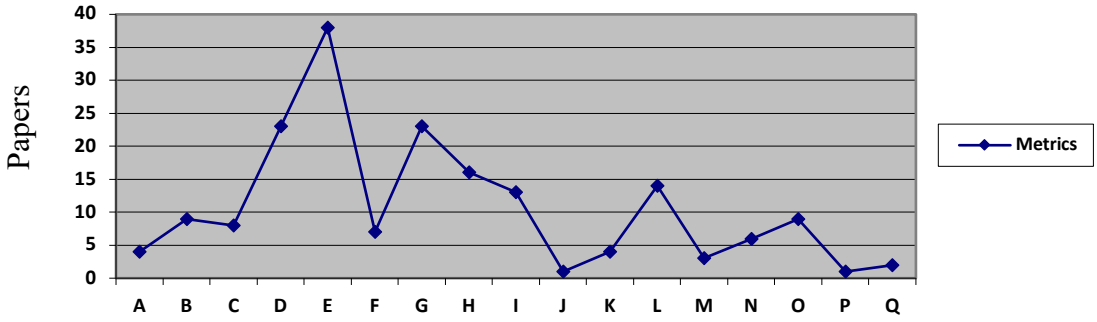


Figure 2.1 Graphical analysis of usage of analysis metrics

This analysis of metrics shows that the recall metric is used by maximum researchers to analyze their results. The graph also helps in analyzing the usage gaps between all the metrics.

2.5. Literature Review

For performing the research papers' complete study, various aspects and attributes of these studies were divided into columns, and all such columns were collated together to form a spreadsheet. Such information clusters included the problem statement, the dataset, the proposed model, deep learning technique, type of recommender system used, and others. Later in the Conclusions section, the significant research gaps were identified and reported, extracted from the studied research papers. In this section, a holistic study of all the research papers has been presented.

2.6. Deep Learning Techniques

To efficiently implement the recommendation process, several deep learning techniques were used in the referred research work. Table 2.6 lists all the techniques used in the papers and hence, classifies the research work.

Table 2.6 Classification of Deep Learning Techniques applied in Recommender Systems

Deep Learning Technique	Studies
Recurrent Neural Network (RNN)	[19], [22], [30], [33], [46], [47], [48], [49], [50], [54], [55], [58], [69], [115], [73], [74], [78], [79], [81], [82], [56], [89], [91], [92], [105], [50]
Deep Neural Network	[23], [25], [26], [27], [31], [36], [53], [62], [65], [68], [77], [90], [93], [98], [112], [103], [99], [106], [116]

Convolutional Neural Network (CNN)	[21], [24], [28], [33], [108], [57], [59], [61], [69], [78], [83], [85], [86], [87], [99], [100], [104]
Deep Belief Network (DBN)	[29], [37], [45], [51], [70], [76], [80], [88]
Deep Feed Forward Neural Network	[38], [40], [64], [66], [95], [112]
Stacked Denoising Autoencoder (SdA)	[32], [107], [44], [100], [102]
Collaborative Deep Learning (CDL)	[108], [60], [42], [97]
Sparse Autoencoder	[67]
Multi-view Deep Neural Network (MV-DNN)	[20]
Supervised Neural Network	[35]
Deep Learning Matcher (DLM)	[41]
Variational Autoencoder	[111]
Generative Adversarial Network (GAN)	[110]
Denoising Autoencoder	[43]
Deep Density Networks (DDN)	[75]
Neural Matrix Factorization (NeuMF)	[101]

It can be seen from the table that the Recurrent Neural Network (RNN) is the most used deep learning technique in recommender systems. Hence, users can try to implement other deep learning techniques and perform a comparative study. In this subsection, the problems addressed in different papers have been categorized into eleven clusters of deep learning techniques as presented henceforth. Every subsection expounds a detailed research work done in recommender systems for each deep learning technique. At the end of every subsection, a detailed description of the findings and open issues have been presented.

2.6.1. Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP) is a feed-forward network with one or several computation layers and has non-linear activations. It has at least one layer that is connected in a feed-forward manner [117]. It can also be used to transform the linear methods of recommender systems into non-linear models. Convolutional Neural Networks (CNN) are also a kind of feed-forward network. In a deep forward neural network, the information flows in one direction across multiple layers, as shown in Figure 2.2. The output from one layer becomes the input into the next layer. This architecture does not consist of any cycles, and hence it is called "feed-forward."

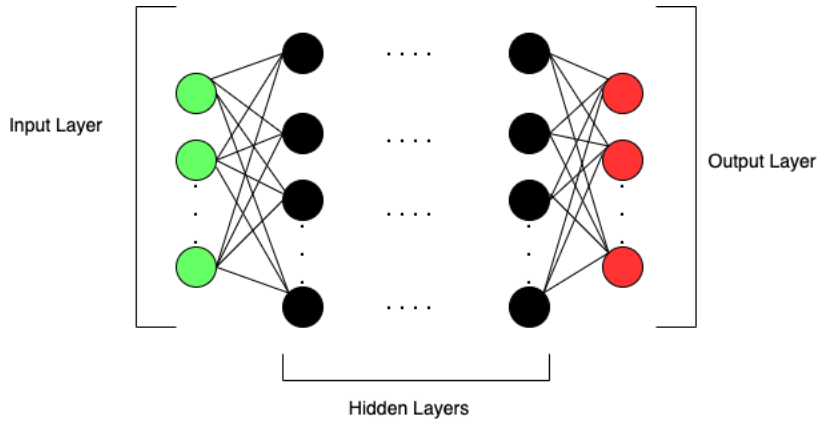


Figure 2.2 Architecture of Deep Feed-Forward Neural Network

For a feed-forward neural network with D inputs, $x = [x_1, \ldots\ldots, x_D]$, a layer with K hidden nodes $h = [h_1, \ldots\ldots, h_K]$, then the output node y is given by Equation 2.1.

$$y = v^T h = v^T f(w^T x) \tag{2.1}$$

where,

$v = [v_1, \ldots\ldots, v_K] \in \mathbb{R}^K, W = [w_1, \ldots\ldots, w_K] \in \mathbb{R}^{D \times K}$

f represents the nonlinear activation function.

The given Equation 2.2 mathematically represents the value of every hidden node.

$$h_k = f\left(w_k^T x\right) = f\left(\sum_{d=1}^{D} w_{dk} x_d\right) \tag{2.2}$$

2.6.1.1 Wide & Deep Learning

The wide and deep learning model is capable of solving both regression and classification problems [118]. The wide learning element is a generalized linear model consisting of a single layer perceptron, whereas the deep learning element consists of the multilayer perceptron. The blend of the two above stated elements leads to the inclusion of both memorization and generalization. The ability of wide learning element to capture prominent features from historical data results in memorization. Whereas the ability of deep learning element to create general and abstract representations results in generalization. The amalgamation of the two results in diverse results and better performance of the recommender system. The architecture of this model is given in Figure 2.3.

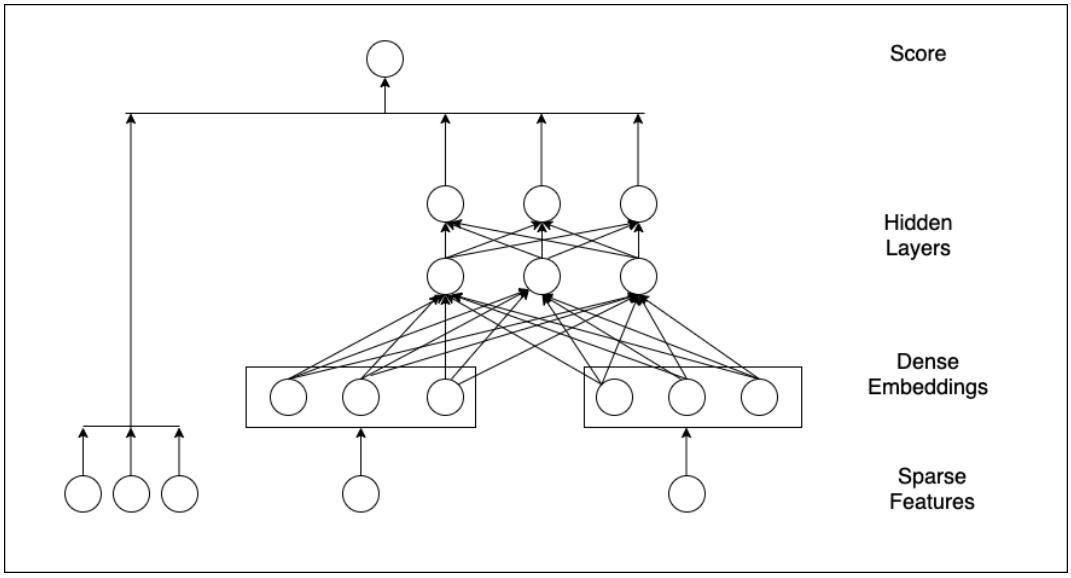


Figure 2.3 Architecture of Wide & Deep Network

This concept can be represented mathematically using Equation 2.3, which represents the wide learning element, and Equation 2.4, which represents the deep learning element, respectively. The final wide & deep learning model is represented by Equation 2.5:

$$y = W_{wide}^{T}\{x, \phi(x)\} + b \tag{2.3}$$

here,

W_{wide}^{T} and b represent the model parameters,

x represents the raw input feature, and

$\phi(x)$ represents the transformed feature

$$a^{l+1} = f\left(W_{deep}^{l}\, a^{l} + b^{l}\right) \tag{2.4}$$

here,

l represents the l^{th} layer,

$f(\cdot)$ represents the activation function,

W_{deep}^{l} represents the weight term, and

b^{l} represents the bias term

$$P(\hat{r}_{ui} = 1|x) = \sigma\left(W_{wide}^{T}\{x, \phi(x)\} + W_{deep}^{T}\, a^{lf} + bias\right) \tag{2.5}$$

here,

$\sigma(\cdot)$ represents the sigmoid function,

$\hat{r}_{ui}$ represents the binary rating label,

a^{lf} represents the final activation

Authors in [119] used a feed-forward multilayer neural network for collaborative filtering recommender systems. Once the model learned the hidden factors and features, the system feed-forwarded the latent features to identify <user,item> pairs having NULL ratings. The authors [66] realized that the usage of hashtags in social media required additional efforts by the user. Hence, they used the deep forward neural network to recommend relevant hashtags to the users. To carry out this task, first, they performed tweet collection and data pre-processing, followed by feature engineering by vector generation. In the end, they performed training and evaluation. In another work, the authors addressed the sparsity of data by using Wide & Deep learning by training

comprehensive linear models with feed-forward deep neural networks [112]. It was found out that rating prediction techniques often rely on user's private information that is a threat to privacy. Hence, the authors came up with a feed-forward neural network-based dTrust model that used the topology of anonymous user-item interactions that combined user's trust relations with user rating scores [64]. In another implementation of the feed-forward deep neural network, the researchers addressed the 0/1 recommendation problem by combining the ratings given by users and Natural Language Processing (NLP) of texts [38]. For using hashtags, user-defined tags usually suffer from various problems like data sparsity, redundancy, ambiguity, and others. To solve this problem, the authors [67] abstracted features that were used to make recommendations instead of raw data. The authors also addressed this problem, and they solved it by using deep neural network [68]. In another work, the authors observed that since recommender systems influence both content and user interactions to create recommendations that adapt to the users' preferences, this can be used as a leverage to improve recommendations [36]. They proposed a Deep Space model that learned a user-independent high dimensional based on their substitutability, semantic space items were positioned, and according to the user's past preferences, it learned user-specific transformation function to convert this space into a ranking. In news recommender systems, modeling temporal behavior, the cost of estimating the parameters also increases, making the recommendations costly. The authors in [31] used the deep neural network to address this issue. It was observed by some researchers [65] that the preference of choosing friends in social media did not always match. Hence it became challenging to recommend friends on online social networks. Thus, they introduced a deep learning approach to learn about both user preferences and the social influence of friends for an effective recommendation. In the paper [53], as a solution to the cold-start problem in music recommenders, the authors combined text and audio information with user feedback data using deep neural network frameworks.

Some authors in [62] observed that sometimes erroneous police photo lineups resulted in the conviction of innocent suspects. To avoid this, they came up with a two leveled approach. The first one was based on the visual descriptors of the deep neural network, and the other was based on the content-based features of persons. The authors [23] proposed a model based on the deep neural network for product recommendations, which required only ratings for making recommendations. In another work [27], the authors realized that the amount of calculation for the learned model to

predict all user-item pairs' preferences was humungous. To solve this problem, they proposed a TDM attention-DNN (tree-based deep model using attention network). For an appropriate research paper recommendation to the scholars, a study [72] used the deep neural network with the paragraph vectors re-ranking method for adequate recommendations. It was observed by some researchers that side information written for business reviews were seldom taken into account for the recommendation [25]. Hence, they used an artificial neural network in a hybrid recommender with the inclusion of side information. Some authors [26] addressed the issue of cross-domain in social recommendation. To solve this problem, they introduced the model Neural Social Collaborative Ranking (NSCR), which immaculately integrated user-item interactions of the information sector and user-user social relations of the social sector. In a study, it was observed that the existing recommender systems did not take into account the effect of distrust among users [94]. They generate recommendations pertaining to the trust relations among users. As a solution, items were recommended for both the trust and distrust relations among active users. In this paper [120], the authors proposed a novel recommender system RecDNNing that combined the user and item embeddings using a deep neural network. First, the authors created deep embedding for users and items, and later the average and concatenated values of those embeddings are given as input to the system. The deep layers generated recommendations using the forward propagation method. The authors in [121] used the hybrid of content-based and collaborative filtering approaches to create a deep classification model for generating efficient music recommendations. To carry out this implementation, they came up with the Tunes Recommendation Systems (T-RecSys) algorithm. Authors in [122] proposed a deep learning-based novel collaborative filtering algorithm. The input of the system were the normalized values of user and item rating vectors. This resulted in decreasing the time complexity as the system need not learn the features of users and items. In paper [123], the researchers applied deep learning in the domain of agriculture. They used the Twitter platform to scrape agriculture tweets and applied sentiment analysis on those tweets. This helped the authors to predict the sentiment range of agriculture tweets.

2.6.1.2 Findings and Open Issues

Deep feed-forward networks Multilayer Perceptron is used to estimate any calculatable function to any given measure of accuracy. It also acts as a foundation for other novel techniques and is adopted in numerous domains. Implementing Multilayer Perceptron for feature representation is

very elementary and remarkably efficient, although it might not be as demonstrative as autoencoders, CNNs, and RNNs. However, one major limitation of deep feed-forward neural networks is that they do not have memories or loops for remembering preceding computations [124].

One of the primary reasons for employing Deep Neural Networks is that they are synthesized so that multiple neural networks can be merged into one big differentiable function and trained end-to-end. This application becomes a necessity because of the abundant availability of multi-modal data. The DNN framework for recommender systems typically extracts user and item feature vectors or latent and explicit features. Another significant advantage of Deep Neural Network is that it can effectively extract essential features from raw data automatically. The effect of upper-case letters in hashtags used for hashtag recommendations can be explored, and its effect on the recommendations can be analyzed in detail. Another scope in the future is to analyze the effect of language modeling on prediction performance. Another scope is to remediate the cold-start problem for user-based Collaborative Filtering by learning the similarities between user-to-user and user-to-job. This can be implemented on a multimodal document embedding.

2.6.2. Multi-View Deep Neural Network (MV-DNN)

Multi-View Deep Neural Network (MV-DNN) is excellent in modeling domain recommendations [118]. It considers users as the main view and every other domain, say Z, as a secondary view. For every secondary or auxiliary user-domain pair, there exists a specific similarity score. The loss function of MV-DNN can be computed using the following Equation 2.6. The architecture of MV-DNN has been represented in Figure 2.4.

$$\mathcal{L} = argmin_{\theta} \sum_{j=1}^{Z} \frac{exp\left(\gamma \,.\, cos\left(Y_u, Y_{a,j}\right)\right)}{\sum_{X' \in R^{da}} exp\left(\gamma \,.\, cos\left(Y_u, f_a(X')\right)\right)} \tag{2.6}$$

here,

θ represents the model parameters,

γ represents the smoothing factor,

Y_u represents the user's output view,

a represents active view's index, and

R_{da} represents view a's input domain

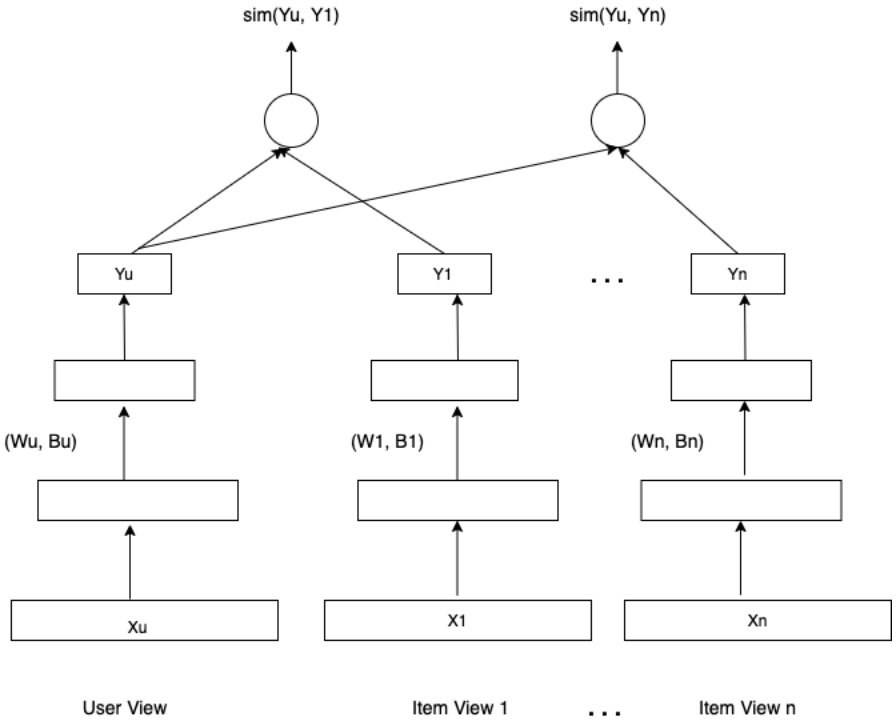


Figure 2.4 Architecture of MV-DNN

Figure 2.5 showcases the structure of the Deep Structured Semantic Model (DSSM) [20]. The crude textual features are fed as input in the form of a high dimensional vector. The DSSM forwards these inputs to two neural networks separately and maps them into semantic vectors into a joint semantic space.

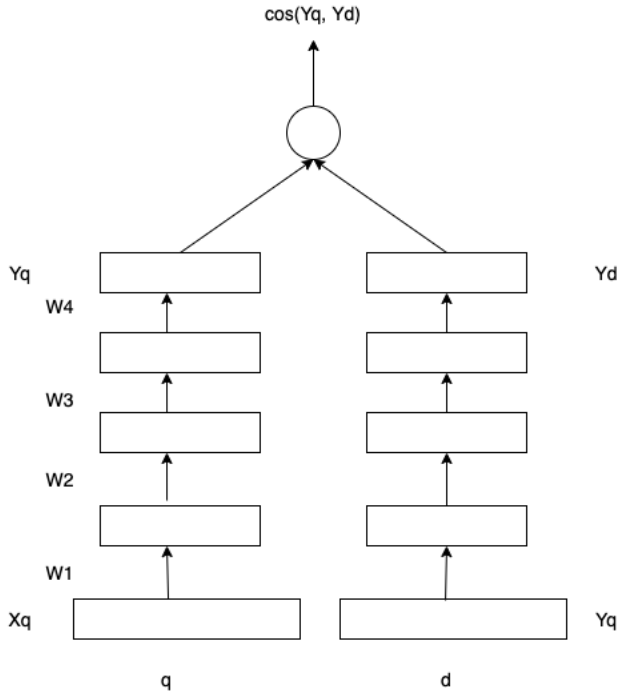


Figure 2.5 Architecture of DSSM

In Equation 2.7, 2.8, and 2.9 , x represents the input vector, y represents the output vector, Wi represents the ith weight matrix, and li represents the hidden layers, such that $\forall i \in \{1,.....,N\text{-}1\}$. q represents the query and d represents the document.

$$l_1 = W_1 x \tag{2.7}$$

$$l_i = f(W_i l_{i-1} + b_i), \qquad i \in \{2, \ldots., N-1\} \tag{2.8}$$

$$y = f(W_N l_{N-1} + b_N) \tag{2.9}$$

The activation function is given by the following Equation 2.10:

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.10}$$

The semantic relevance score between a query Q and a document D is given by the following Equation 2.11:

$$R(q,d) = \cos\left(Y_q, Y_d\right) = \frac{Y_q^T\, Y_d}{\| Y_q \| \cdot \| Y_d \|} \tag{2.11}$$

MV-DNN can be implemented in several domains. Conforming to the concepts of user-based collaborative filtering, users having similar preferences in one domain tend to have similar preferences in other domains as well. However, in many cases, this assumption may be rendered ineffective. Hence, the basic knowledge of the correlations between different domains is an essential aspect of MV-DNN. It is based on a Deep Structure-based Semantic Model (DSSM). In MV-DNN, the architecture of DSSM contains multiple views to map sparse features in high dimensions into the low dimensional dense matrix. The authors observed that the online services present humungous content to the users, making this content user-centric [20]. Hence, they came up with the Deep Structured Semantic Model (DSSM), which used content-based filtering to improve the recommendations. The authors on [39] observed that current CF-based techniques could only comprehend a single type of relations. RBM, for example, considers either user-user or item-item relations. The matrix factorization approach considers use-item relations only. To fix this problem, the authors propose a framework that first learns low dimension user and item vectors. This captures the user-user and item-item relations. This is then passed to a multi-view deep neural net, which models the user-item interaction.

2.6.2.1 Findings and Open Issues

Initially, Multi-View Deep Neural Network (MV-DNN) was incepted for performing cross-domain recommendations. However, the understanding of cross-domain recommendations may not be fruitful for MV-DNN. This is said essentially because the underlying literature of generating recommendations states that if a user likes an item *a* and there exists an item *b* similar to item *a*, then the user will like item *b* as well. However, this is not always true. For the times this hypothesis fails, this assumption becomes obstructive for the implementation of MV-DNN.

2.6.3. Convolution Neural Network (CNN)

For bio-inspired Multilayer Perceptrons and Computer Vision, CNN are the most used deep learning models. The essential elements of CNN are the convolutional layers and the subsampling layers. The convolutional layers work as a filter for the output from previous layers. These convolutional layers hence produce filtered outputs. The subsampling layers subsample the convolution output based on their activations [125]. To create a deep CNN model, the convolutional layers and the subsample layers are added alternatively. Hence, such a deep model of CNN can learn a hierarchy of complex features. Another massive advantage of CNN is that it has fewer parameters than the traditional feed-forward neural networks. This method is based on the feed-forward deep neural network. To reduce the preprocessing, multilayered perceptrons are used in this model. The architecture of CNN has been shown in Figure 2.6.

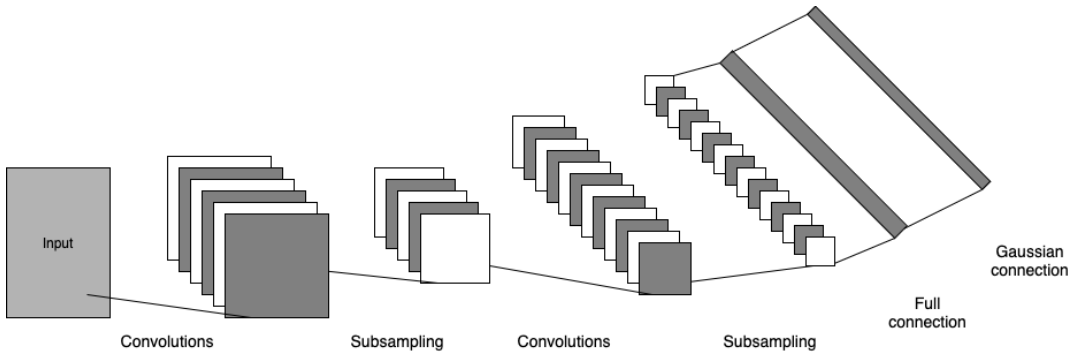


Figure 2.6 Architecture of CNN

Cooperative Neural Network (CoNN) is a model where two neural networks work in tandem [24]. To discover novel user and item features, user and item reviews are given as input to the system. A common shared layer is added on the top of the two neural networks which couples them and the user and item features are mapped together into a common feature vector space to enable the interaction amongst them. Figure 2.7 represents the architecture of CoNN. Another latent layer is added to the architecture to enable the interaction of user and item latent features. The two neural networks, i.e., neural network for items (*NNi*) and neural network for users (*NNu*), run parallel. The item and user ratings are fed as inputs to the two neural networks, respectively. In the lookup layer, the user and item review texts are placed as matrices of word embeddings. The subsequent

layers perform the functions of convolution, max pooling, and full connection, respectively. This layer also acts as a platform for computing the objective function for calculating the rating prediction error. One major drawback of Cooperative Neural Network is that it is incapable of addressing users and items which do not have ratings.

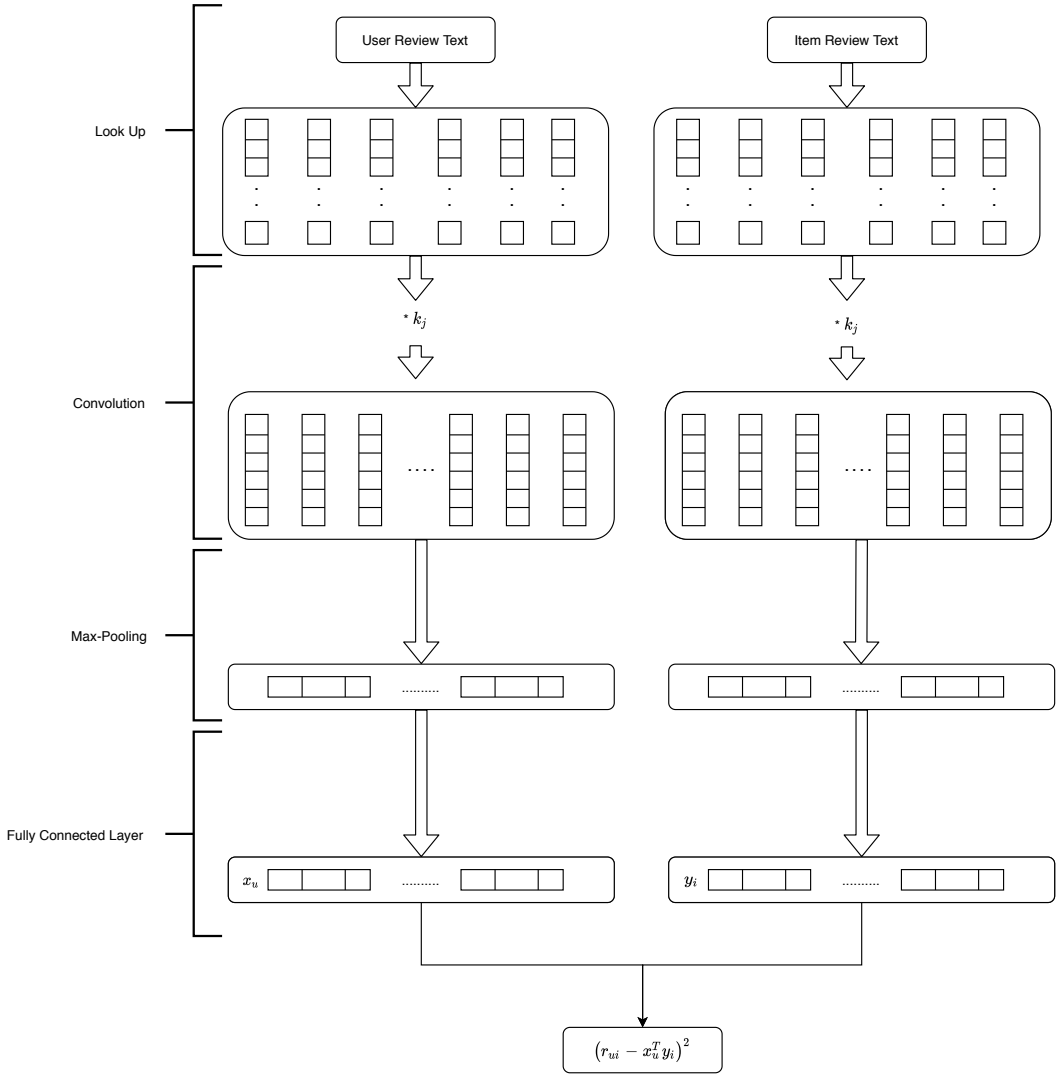


Figure 2.7 Architecture of CoNN

Many researchers addressed data sparsity, and they used CNN in their papers to solve this problem. The authors in [24] observed that maximum recommender systems ignored the reviews leading to increased data sparsity problem. Thus, they proposed Deep Cooperative Neural Networks

(DeepCoNN) model to learn item properties and user behaviors in the review text. Similarly, to solve the issue of the sparsity of data, authors integrated convolutional neural network (CNN) into probabilistic matrix factorization (PMF), resulting in convolutional matrix factorization (ConvMF) [57]. In the research work [110], the researchers solve the data sparsity problem by including tag and user information. First in nearest neighbors set, the most significant user impact is found using similarity metric, to process item information CNN is used. To get the results, the prediction matrix is decomposed by the probability matrix. The profound use of CNN in the fashion industry was acknowledged in some research papers. The authors in [84] observed that not much work had been done on complicated recommendation scenarios involving knowledge transfer across multiple domains in fashion recommendations. Similarly, for online clothing shopping, current methods did not address the challenges in the cross-domain clothing retrieval scenario completely. The intra-domain and cross-domain data relations were considered together, and the numbers of matched and unmatched cross-domain pairs were imbalanced. Hence, the authors [85] proposed a deep cross-triplet embedding algorithm and a cross-triplet sampling technique to provide improved recommendations.

Some other applications of CNN have been included in the survey. For efficient venue recommendation, the authors proposed a City Melange framework that matched the interacting user to social media platforms with similar interests [87]. As a result, this approach could recommend both on- and off-the-beaten-track locations to the users. In another work, the authors proposed a hybrid music recommender system that used CNN to model real-world users' information and high-level rendering of audio data [108]. The authors observed in [61] that finding appropriate Ukiyo (a Japanese firm) e-prints that intrigue a novice is challenging. Thus, they proposed Ukiyo-e recommendation using a deep learning-based CNN model to present recommendations. The input to the recommender system is an image provided by the user. The CNN model is used to create a classifier that takes input images and other information related to the image and outputs whether the user will like or dislike it. The authors in [99] used tag information and user information and improved the recommendation by obtaining the nearest neighbor set that significantly impacted on the target user. They named their model Convolution Deep Learning model on Label Weight Nearest neighbors (LWNCDL). Traditionally, candidate articles were handpicked from a vast pool of articles, and hence, its recommendation was manual

and laborious. To solve this problem, the Dynamic Attention Deep Model (DADM) was proposed, which used multiple neural networks [59]. As a result, the authors used the Convolutional Neural Network to study the connection between visual and textual inputs and examine the possibility of knowledge transfer from complex domains for expert recommendations. The researchers in [83] addressed the problem of job recommendation by analyzing semi-structured resumes of candidates. To recommend a job, they used CNN to process the entire input at once, which is important since any part of the text in a resume can affect its semantics. The authors in [126] proposed an AODR model that considers user and item rating texts to infer item aspects and user opinions by applying deep learning. Such item aspects and user opinions were later merged using collaborative filtering to learn insightful information. In paper [117], the researchers introduced an algorithm, Aspect-Based Opinion Mining (ABOM), which analyzed the user reviews extensively to learn hidden features that further improve the recommendation accuracy. To carry out this task, they used multichannel deep CNN (MCNN) and Tensor Factorization (TF) machines. In paper [127], the authors used decentralized knowledge graphs in deep recommender systems to validate the efficiency of the system. The knowledge graph was constructed by crowdsourcing. The authors in [128] created a passenger hunting recommender system. The proposed system consists of two components, i.e., offline training and online implementation. In the former component, the authors applied deep CNN, and in the latter component, the authors proposed the DL-PHRec method. Hence, the authors could generate a personalized ranking list of destinations regions for each taxi driver.

2.6.3.1 Findings and Open Issues

CNNs are efficient in handling unstructured multimedia data with convolution and pooling functions. The majority of CNN based recommender models use CNNs for performing feature extraction. CNN based models are instrumental in learning deep features to model user and item latent factors. The significant advantage of using CNN is its implementation of pooling operation for reducing training data dimensions. CNN has been used for feature extraction to efficiently model user and item representations for Recommender Systems. It is also useful in image processing. Another new scope is to embed the user's rating information in a matrix. A Convolution Neural Network can be trained on this matrix, treating it like an image where each data point represents a particular feature of the user. Exploiting Convolution Neural Networks for

prediction of risk with medical imaging can be explored. One major limitation is that it requires massive hyperparameter tuning to extract optimal features. In addition to this, it is also challenging to support intricate activity details. Although CNN uses feed-forwarding, it has fewer parameters than traditional deep feed-forward networks [125].

2.6.4. Auto Encoders (AE)

Autoencoders are typically unsupervised neural networks that are trained to mirror its input as output. It is a tri-layered neural network, consisting of the input layer, hidden layer, and the output layer. The input layer is fed with the complex representations of the dataset, and in the hidden layer, such complex representations are transformed into low-dimensional representations. It essentially mirrors the working of an encoder, which encodes the complex, high dimensional representations into low dimensional representations. Mirroring the operations in a reverse manner, the low dimensional representations are converted into high dimensional representations as the data travels from hidden layer to output layer. This can also be termed as the working of a decoder. The architecture of Denoising AutoEncoder has been given in Figure 2.8.

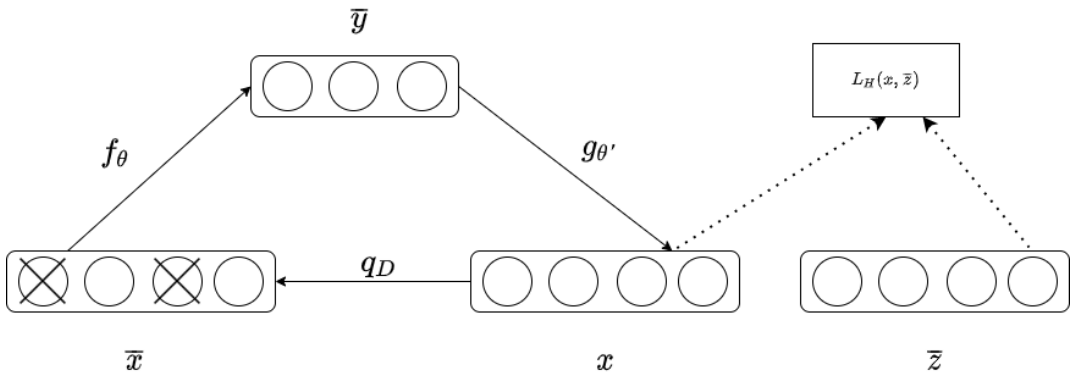


Figure 2.8 Architecture of Denoising AutoEncoder

The feature extraction performed as the encoding is not robust enough. It was observed that the addition of Gaussian noise improved the problem described above [129]. Hence, to make the system more robust and training the hidden layers to identify hidden data features, denoising autoencoders were introduced. The structure of such autoencoders encodes the input while preserving the information about the input, and reduces the effect of the alteration process, applied

stochastically to the input of the autoencoder. The authors in this work attempted to improve the automatic recommendations [35]. This was done by extracting the features of input data and reconstituting the input to perform recommendations. Authors in [130] extended generative models for the task of Collaborative filtering. To do this, they used Wasserstein autoencoders.

2.6.4.1 Stacked Denoising AutoEncoders (SDAE)

A stacked auto-encoder is a neural network having several layers of sparse autoencoders wherein the output of each layer is forwarded as the input of the successive layer. The hidden layers in a denoising auto-encoders can be colluded to create a deep network by forwarding the output of the previous layer as the input to the next layer. Such a strong feature extraction capability makes stacked denoising autoencoder aptly suited for recommender systems. The architecture of a Stacked Denoising AutoEncoder has been given in Figure 2.9.

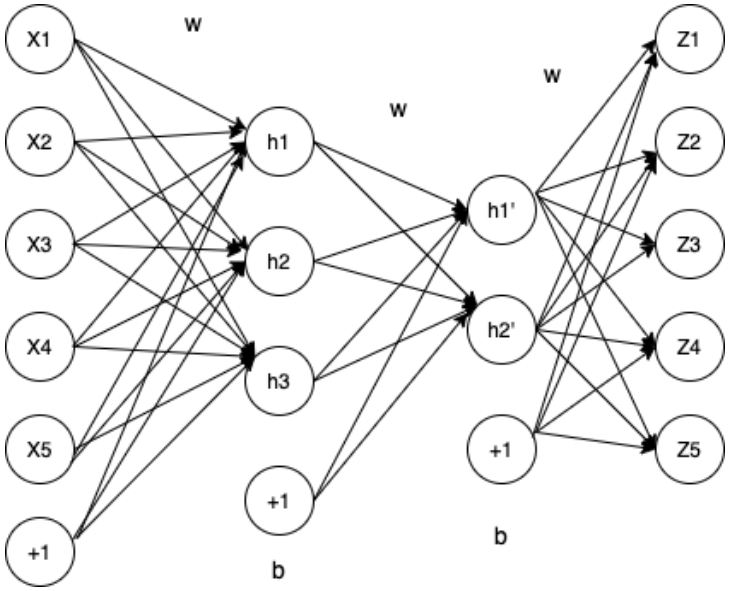


Figure 2.9 Architecture of Stacked Denoising AutoEncoder

The method of generating top-N recommendations using Stacked denoising Autoencoder starts with selecting a dataset with user reviews [32]. Then the similarity between the items liked by the user is calculated for each user. Afterward, the nearest data points with a defined length, say K, are selected. Such K-nearest datasets are combined, and the user reviewed movies are excluded

from such datasets to form another dataset. The similarity between the two datasets is calculated, and the top-N similar movies among the two datasets are selected and recommended to the user.

Upon stacking the denoising autoencoder, the deep model hence created increases the capability of feature extraction. To make this task robust, Gaussian noise is added to the system. The authors proposed a Hybrid Recommendation system with CF and Deep learning (HRCD) [107]. It explored the content features of the items learned from a deep learning neural network and applied later to the timeSVD++ CF model. The authors in [32] addressed the problem of information overloading and data sparsity. As a solution, stacked autoencoders were used with denoising to extract low dimensional features from the sparse user-item matrix. It was observed that personalized recommendations often led to sparse observations of users' adoption of items. As a solution to the problem, the authors used Collaborative Topic Regression with Denoising AutoEncoder or (CTR-DAE) [43]. Here, the user's community-based preference was bridged with his topic-based preference. Similarly, some other researchers addressed data sparsity by proposing a hybrid model that executed deep learning of users and items' latent factors from side information and CF obtained from the rating matrix [102]. According to some authors, CF recommender systems suffer from Complete Cold Start (CCS) problem where zero rating records are present and Incomplete Cold Start (ICS) problems where significantly less rating records are available for new items or users in the system [44]. Hence, they used timeSVD++ models and used temporal dynamics of user preferences and item features for improvised recommendations.

2.6.4.2 Variational Autoencoders (VAE)

Due to the drawbacks of collaborative based filtering, the authors in [111] used variational autoencoders to include rating and content information for the recommendation. The novelty in the framework is that the latent distribution for content is learned in latent space rather than observational space. In another paper, the authors demonstrated the enhancement of modeling in CF with side information by using Variational Auto Encoders-based Collaborative Filtering (VAE-CF) [110]. The authors [131] proposed a novel healthcare recommender system collaborative variational deep learning (CVDL) to eliminate the limitations of data sparsity and cold start problem from recommender systems to generate useful recommendations. The paper [132] proposes a deep autoencoder model to learn low and high-dimensional features to remediate data

sparsity and data imbalance problems in recommender systems. To make the task of feature selection simpler and more efficient, this paper [133] proposed a fuzzy entropy-based deep learning recommender system to decrease data dimensionality and eliminating unnecessary noise from data. This paper [134] proposed an end-to-end recommender model by taking several content sources, for example, textual content, graphical content, and others. The authors used a stacked autoencoder to carry out this implementation. In this paper [135], the authors identified the limitations of side information in RS. Apart from having extensive detail about the rating, side information also contains a tremendous amount of noise. Hence, the process of feature extraction from such side information becomes challenging. So, the authors proposed a Stacked Discriminative Denoising Auto-Encoder by merging deep learning with matrix Factorization based RS.

2.6.4.3 Findings and Open Issues

Autoencoder is an efficient feature representation learning method that learns feature representations from user-item content features. Autoencoder are also implemented in recommender systems by learning low-dimensional feature representations at the outer layer or by building the interaction matrix crafted in the reconstruction layer. Autoencoder models have a high capability to work with noisy data to learn the complicated and hierarchical structure from the input data. Sparse-autoencoders are very efficient for low-dimensional feature extraction from input data using a supervised learning technique. Denoising autoencoders are trained by initializing layers wherein each layer produces input data for the next layer. One of its significant advantages lies in implementing recommender systems as denoising autoencoders can be stacked to decrease the processing errors. Autoencoders suffer from certain limitations. Autoencoder based Collaborative Filtering (ACF) is incapable of incorporating non-integer ratings, and partially observed features result in low recommendation accuracy. Another major disadvantage of using deep autoencoders is that they are incapable of searching for an optimal solution, and due to high parameter tuning, the time complexity of the training process increases manifold.

2.6.5 Neural Matrix Factorization (NMF)

Matrix Factorization (MF) is one of the most widely and commonly used Collaborative Filtering approaches. Such MF models learn the low-dimensional embeddings of items and users in

common latent factor space. The system generates a user-item matrix where the ratings provided for every item by the users are captured. However, due to the problem of data sparsity, the user-item matrix is sparse. Such a sparse matrix is factorized into a dense matrix to make computations simple. This is done by taking the product of two low-rank matrices consisting of user-item embeddings. Upon learning the latent factors, the similarity between user and item is computed. In addition to this, the newly discovered preferences are considered by analyzing the user and item latent factor representations. To learn such latent user-item factors, loss function has to be calculated as given in Equation 2.11:

$$min_{v_u v_i} \sum_{(u,i)\in S} (r_{ui} - \langle v_u, v_i \rangle)^2 + \lambda(\| v_u \|^2 + \| v_i \|^2) \tag{2.11}$$

here,

v_i represents the latent factor of item i,

v_u represents the latent factor of user u,

r_{ui} represents the rating given to item i by user u, and

λ represents the item bias to prevent overfitting

Factorization Machines (FM) are supervised learning models that are used in various prediction problems like regression, classification, and others. These models map random real features into low-dimensional latent feature vector space. They can determine the parameters of the model precisely given a sparse user-item matrix and train with linear complexity. Such characteristic makes FMs ideal for implementing real-world recommender systems.

2.6.5.1 Deep Factorization Machines (DeepFM)

DeepFM is a model wherein the capabilities of Factorization Machines are combined with deep feature learning to form a novel neural architecture, wherein Factorization Machines and Deep Neural Network (DNN) are integrated to model both low dimensional and high-dimensional feature vectors [136]. As the name suggests. DeepFM is a dual component model, i.e., the factorization machine component and deep learning component. These components share the same

input. These two components are jointly trained for the prediction model computed using the following Equation 2.12.

$$\hat{y} = sigmoid(y_{FM} + Y_{DNN}) \tag{2.12}$$

here,

$\hat{y} \in (0,1)$,

Y_{FM} represents the output of the FM component, and

Y_{DNN} represents the output of the deep learning component

The main advantages of using this model are no requirement for pre-training the model, its capability to learn both low-dimensional and high-dimensional features. The shared feature embedding of FM and deep components enables the system to avoid extensive feature engineering. Rather than conventional push-pull methodology amongst the user and item pairs, the authors proposed LRML (Latent Relational Metric Learning) model to learn latent relations that described every user-item interaction [96]. The authors observed in [125] that although Weighted Matrix Factorization (WMF) can learn latent features from implicit feedback, these features are still not good enough to train a CNN model. The authors in [103] proposed a novel matrix factorization method with neural network architecture and novel use of binary cross-entropy loss function. In another work, the authors identified that Factorization Machines (FM) are incapable of modeling the non-linear aspect of the real-world data [112]. In [63], the authors observed that the recommendation systems primarily relied on initializing the user and item latent feature vectors. In this way, they used deep learning to estimate the initialization in MF for trust-aware social recommendations and to differentiate the neighborhood effect in the user's trusted social circle. This was achieved with the help of the Deep Learning-based Matrix Factorization (DLMF) model.

2.6.5.2 Findings and Open Issues

Neural Matrix Factorization provides exemplary results for latent feature models; however, it is influenced by methods involving local graph structure. One limitation of Neural Matrix Factorization is that to investigate the scope of system architectures, activation functions, regularization techniques, and cross-validation strategies. There is a risk of overfitting, which may lead to erroneous or insignificant insights.

2.6.6 Neural Collaborative Filtering (NCF)

In recommender systems, the users and items collate together to create a mapping that helps to under the similarity between the two and further recommend items to the user. Since the phenomenon of generating recommendations in a dual process, which involves both user latent factors and item latent factors. Hence, to model this two-way interaction between user and item, Neural Collaborative Filtering (NCF) is used [137]. The scoring function of NCF can be computed using the following Equation 2.13:

$$\hat{r}_{ui} = f\left(U^T \cdot s_u^{user}, V^T \cdot s_i^{item} \mid U, V, \theta\right) \tag{2.13}$$

here,

$f(\cdot)$ represents the multilayer perceptron,

s_u^{user} represents the side information of the user profile,

s_i^{item} represents the side information of the item features, and

θ represents the parameters of the network

It becomes easier to combine the neural representation of MF with Multilayer Perceptron to create a model that incorporates the linear characteristics of matrix factorization and non-linear characteristic of MLP. This results in improved performance of the recommender engine. The cross-entropy loss for implicit and explicit feedback is given by the following Equation 2.14:

$$\mathcal{L} = -\sum r_{ui} log\, \hat{r}_{ui} + (1 - r_{ui})\, log(1 - \hat{r}_{ui}) \tag{2.14}$$

here,

u represents the user,

i represents the item, and

r_{ui} represents the rating of item i given by user u

2.6.6.1 Collaborative Deep Learning (CDL)

This technique is an amalgamation of deep representation of content and collaborative filtering for the ratings. CDL is a deep learning recommender model that learns from the text of the reviews given by the users [125]. It combines the autoencoder and Click Through Rate (CTR) to model ratings. To learn from such review texts, CDL implements Bag-Of-Words (BOW) technique. This model has certain limitations, such as it models only item review texts, limiting the capabilities of a recommender system, and due to its usage of the BOW method, it only considers the frequency of words in a text, and not the similarities between different texts. Another limitation of the CDL model is that it considers the order of words in a text which sometimes contradicts the semantics of the text. Figure 2.10 represents the architecture of SDAE, upon which the SDAE is built. Figure 2.11 represents the architecture of CDL.

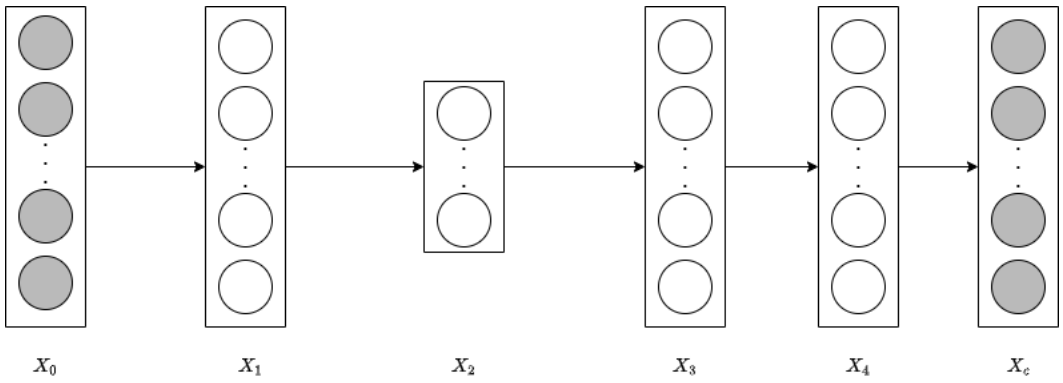


Figure 2.10 Architecture of SDAE

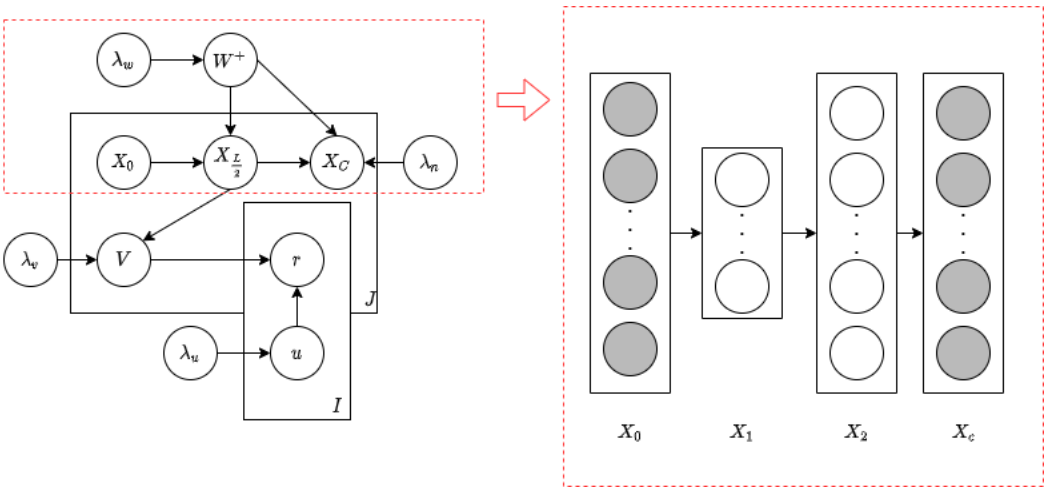


Figure 2.11 Architecture of CDL

The authors argued that the sole criteria for recommending items were user reviews [108]. However, this often led to data sparsity, hence degrading the recommendation quality. To avoid this, the authors came up with a Collaborative Deep Learning (CDL) model, which simultaneously performed deep learning of content and collaborative filtering for the ratings. In another work, the authors used CDL based Bayesian Deep Learning for Recommender Systems model to achieve integrated intelligence that involved both perception and inference in a principled probabilistic framework [109]. In [77], the researchers attempted to improve YouTube recommendations by using a deep collaborative filtering model that efficiently integrated various signals and added layers of depth to the interaction. Some authors in their paper [90] compared the Deep Neural Network (DNN) features with visual features for artwork recommendation and concluded that DNN features outperformed the other. The authors in [98] presented a novel neural network model, Neighborhood-based Neural Collaborative Filtering (NNCF), which jointly characterized both user-item interactions and neighborhood information for the recommendation. The authors in [138] indicated that the Collaborative Filtering algorithms primarily deal with low dimensional and linear interactions between users and items. To address this issue, the authors proposed a novel deep CF recommender algorithm for service recommendations. The authors in [139] used MLP to learn the interaction function to implement the recommender system. The authors [96] proposed a deep multi-criteria collaborative filtering RS to generate accurate multi-criteria recommendations.

2.6.6.2 Findings and Open Issues

Collaborative Deep Learning (CDL) can perform deep learning of content and collaborative filtering for the ratings of items. This allows the model to balance the effects of side information and interaction history. One significant limitation of the CDL model is that it cannot apprehend contextual information with word embeddings and convolutional filters.

2.6.7 Deep Belief Network (DBN)

It is a class of deep learning where several layers of hidden units are stacked together to form a deep net. The hidden units present in each layer are not directly connected. They can be viewed as a class of unsupervised networks like Restricted Boltzmann Machines (RBM) [140]. Authors [141] observed that classic neural networks suffer from the problem of optimization. DBN was

introduced to solve this problem [142]. This technique integrates supervised and unsupervised learning by implementing a local search to get an optimized result and learn the data distribution without prior knowledge [141]. DBN employs a stacked RBM structure for extracting deep features of data [76]. Thus, it can be viewed as a generative probabilistic model [45]. In DBN, the hidden layers are trained in a bottom-up manner. This pre-training of the layers is termed as the pre-train stage, and upon the addition of the objective layer, the training of the model becomes supervised [143]. Figure 2.12 represents the architecture of the Deep Belief Network.

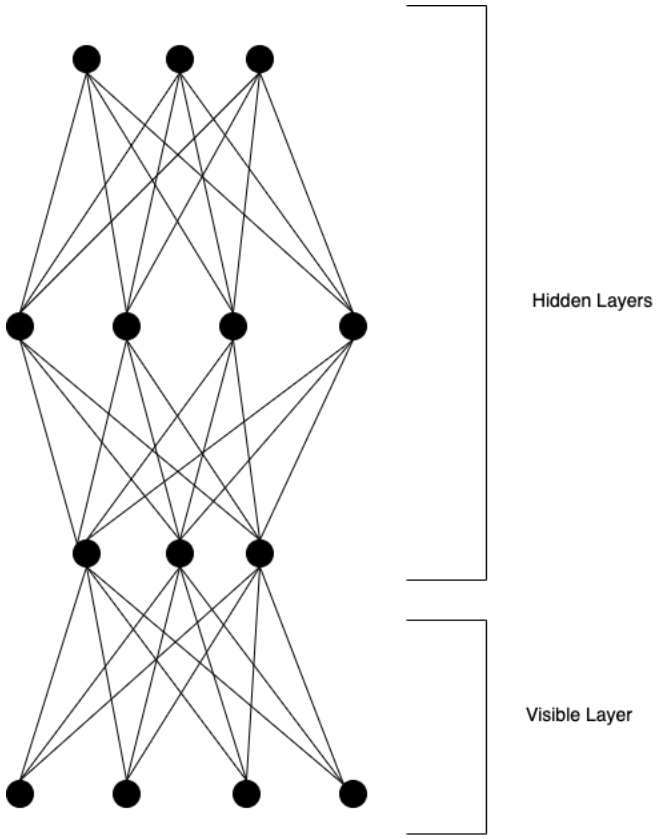


Figure 2.12 Architecture of Deep Belief Network (DBN)

The standard RBM is given by Equation 2.15

$$E(\vec{u}, v) = -\sum_{i=1}^{p}\sum_{j=1}^{q} w_{i,j} u_i v_j - \sum_{i=1}^{p} x_i u_i - \sum_{j=1}^{q} y v_j \tag{2.15}$$

where,

$\vec{u}$ is the visible unit vector and $\vec{v}$ is the hidden unit vector.

w represents the weight matrix and $\vec{x}$ is the visible bias vector and $\vec{y}$ is the hidden bias vector.

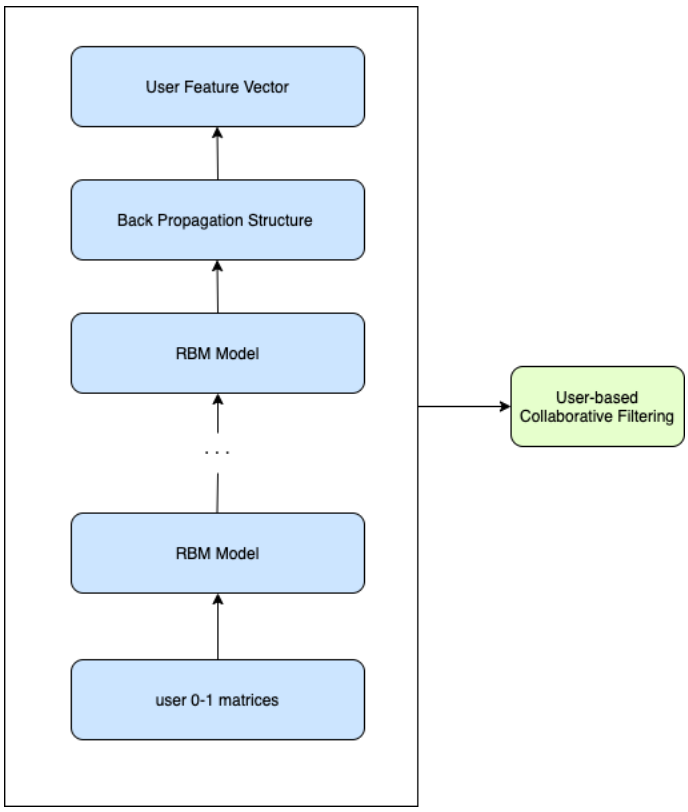


Figure 2.13 Video Recommender system implementing DBN and CF

In video recommender systems, generating recommendations for videos becomes complicated because extracting intricate features from graphics is difficult. The workflow of such a recommender system has been presented in Figure 2.13. To make this task simpler, authors in [76] divided the user-item matrix into 0-1 matrices and feed them as input to the DBN model. Hence, post-training, the difference between trained features and raw features becomes prominent. Afterward, UBCF calculates the similarity between neighboring users to generate recommendations.

2.6.7.1 Findings and Open Issues

One of the most significant advantages of DBN is that Deep Belief Network has been successfully employed to perform extensive feature engineering. It is useful in extracting hidden and useful features from audio data [117]. Another significant advantage of using DBN is that it is also used for dimensionality reduction. DBN has also proved to help solve the data sparsity problem, which leads to the cold start problem and generation of poor-quality recommendations. However, DBN comes with several limitations as well. Due to extensive parameter initialization, the DBN model is challenging to train. Another major disadvantage of DBN is that it cannot learn the representation of features from labeled and unlabeled data [144]. The training of the Deep Belief Network model can be further optimized to improve the recommendations.

2.6.8 Recurrent Neural Network (RNN)

Unlike feed-forward deep neural network, the layers present in a recurrent neural network form a graph, and hence the layers interact with each other. The graphs thus formed can either be cyclic or acyclic [118]. Essentially RNN captures the sequential data temporally, which is used for training the model [145]. The RNN takes the input and computes the weighted sums as given in the following Equation 2.16 and Equation 2.17. The architecture of the Recurrent Neural Network has been presented in Figure 2.14.

$$z_{hj} = \sum_{i=1}^{n_x} x_i w_{ij}^{xh} + \sum_{i=1}^{n_h} h_i w_{ij}^{hh} \tag{2.16}$$

where, n_x are the input nodes and n_h are the hidden nodes.

$$h_j = \tanh(z_{hj}) = \tanh\left(\sum_{i=1}^{n_x} x_i w_{ij}^{xh} + \sum_{i=1}^{n_h} h_i w_{ij}^{hh}\right) \tag{2.17}$$

The weighted sums of the output layer can be calculated as given in Equation 2.18:

$$z_{yj} = \sum_{i=1}^{n_h} h_i w_{ij}^{hy} \tag{2.18}$$

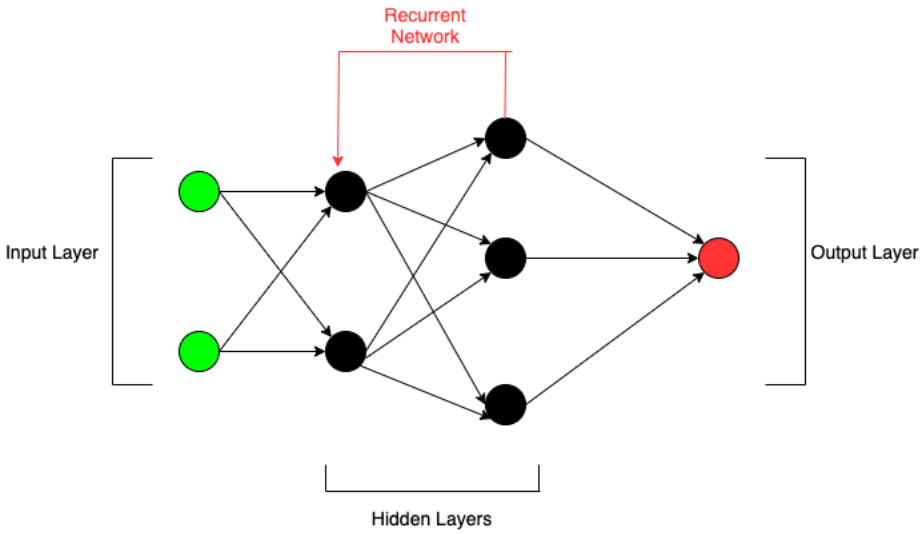


Figure 2.14 Architecture of Recurrent Neural Network

2.6.8.1 Gated Recurrent Unit for Recommender System (GRU4REC)

GRU4REC is a session-based recommender system proposed by [146]. The input given to the model is an encoding of 1 of N, where N represents the number of items. The output of the system determines the possibility of an item to move to the subsequent session. If the item is active in the current session, the coordinate is assigned value 1, otherwise 0. The architecture of GRU4REC has been given in Figure 2.15. The ranking loss function for this model is computed using the following Equation 2.19:

$$\mathcal{L}_s = \frac{1}{S}\sum_{j=1}^{S} \sigma(\hat{r}_{sj} - \hat{r}_{si}) + \sigma(\hat{r}_{sj}^2) \tag{2.19}$$

here,

S represents the sample size,

$\hat{r}_{si}$ represents the scores on negative item i at session s,

$\hat{r}_{sj}$ represents the scores on positive item j at session s, and

σ represents the logistic sigmoid function

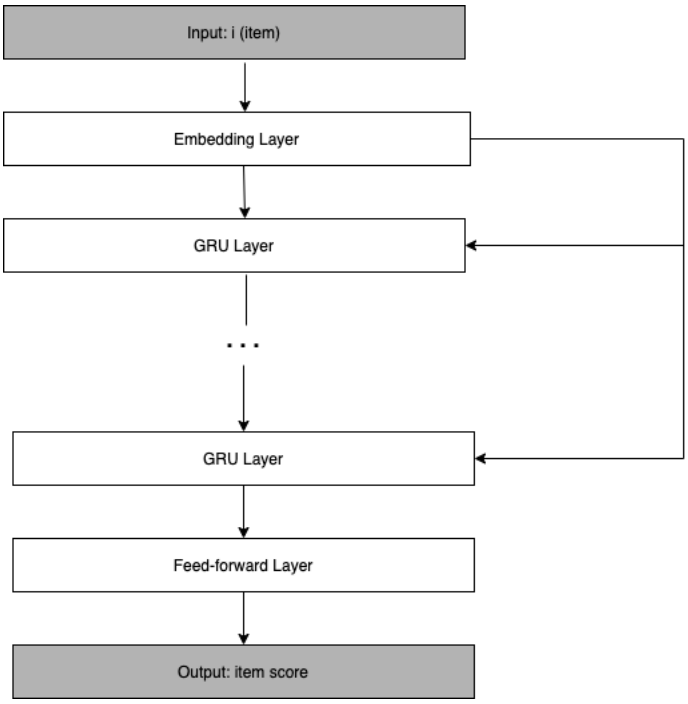


Figure 2.15 Architecture of GRU4REC

In GRU based Recurrent Neural Networks (RNN), the input given to the system depicts the current state of the system, and the output of the system depicts the subsequent event of the system. Unlike in GRU4REC, where 1-of-N encoding is used, in GRU based RNN for session-based recommender systems, a weighted sum of the representations is used. In such cases, the events which occurred in the past are given as discounted when fed as an input. Hence, to make the system stable, the input vector is normalized. When multiple feed-forward layers are added to the system, the output predicts whether the item will be included in the next session or not. The addition of multiple GRU layers uses the output of the previous layer as the input to the next layer. This deep connection of layers improves the performance of the system. In some research studies, the side information was included to make the recommendations efficient. It was observed in [19] that recommender systems based on collaborative filtering could be enhanced by including side information like natural language reviews. To carry out this task, the authors used the bag-of-words technique along with RNN. In [74], according to the authors, current recommendation techniques using RNN only took into account the user's past activities and did not consider the

essential side information. To cater to this problem, they used Contextual Recurrent Neural Networks (CRNN) to include the side information to give recommendations. Some authors considered the temporal while making the recommendations and studied its effects. To model the temporal behavior in recommendation systems, a group of researchers [30] introduced the Temporal Deep Semantic Structured Model (TDSSM) based on RNN, where they used long-term static and short-term temporal user choices to improvise the performance of the recommender system. Similarly, researchers in [71] observed that the conventional POI recommender systems did not consider the temporal aspect while generating the recommendation. Hence, they came up with the idea to capture sequential check-in data of users and used RNN based deep neural network to provide recommendations. Catering to the same issue, the authors in [92] observed that a notable problem with recommender systems is that they do not encompass the context of time. So, they came up with the idea of using RNN and including the temporal shift while performing recommendations.

In a few research papers, the RNN technique was applied for session recommendations. In [146], the authors used the Recurrent Neural Network for session-based recommendations. They incorporated two procedures for improving the model. The first one was data augmentation, and the second was to observe shifts in the input data distribution. Similarly, it was observed in [47] that in the real-life recommender systems, the session-based recommendations were based solely on the clicks of a user session. To solve it, the authors introduced several parallel RNN (p-RNN) frameworks to model sessions depending on the clicks and the features of the clicked items. The researchers in [50] proposed a model for session-based recommendations, where the recommendations were made at the starting of the sessions, which avoided the cold-start problem. In [79], the authors used Long Short-Term Memory (LSTM), a neural network-based RNN technique for quotes recommendation. Similarly, to make recommendations using the data content describing the items, the authors proposed Ask Me Any Rating (AMAR), which used Long Short-Term Memory (LSTM) networks to simultaneously learn two embeddings representing the items to be recommended and user preferences [73]. The concept of LSTM was also used in [50], where the authors proposed Recurrent Recommender Networks (RRN) to incorporate dynamic recommendations using Long Short-Term Memory (LSTM) autoregressive model. In some papers, the problem of data sparsity was addressed and rectified using RNN. The authors in [49]

introduced a ReLaVaR model that addressed data sparsity by considering the network recurrent units as stochastic latent variables with a former distribution performed over them. Similarly, it was observed that personalizing Adaptive User-Interfaces (AUIs) became difficult with several interaction modules as the item-user matrix is sparse. The authors in [91] used architecture consisting of RNN that performed sequential recommendations of content and control elements to solve this problem. Several other independent research works were performed using RNN. The authors in [55] analyzed that in documents, the semantic similarities between texts were not considered for making recommendations. They were able to effectively model the content of the abstracts of the documents using linear regression based on RNN. In another work, it was observed that the evolution and drifting of features might take place over time as users interact with different items. Hence, the authors utilized the recurrent neural network to learn a representation of impact from drift, evolution, and co-evolution of user and item attributes to make recommendations [89]. To improve the performance of multi-task learning, the authors used deep RNN to encode the text sequence into a latent vector trained on the collaborative filtering procedure [58].

Performing recommendations using sentiment analysis of short texts like sentences did not yield correct results due to less contextual information. Hence, the authors used the deep learning technique recurrent neural networks to solve this problem [105]. To find relevant citations or related work, the authors came up with Neural Citation Network (NCN) model to provide a curated list of high-quality candidates from a short passage of text [57]. For generating sequential recommendations, the authors in [22] came up with user-based RNN, which allowed generating personalized suggestions. In another work, to find relevant research articles, researchers usually rely only on the keyword-based search or by following citations. In the paper, the authors analyzed user activities to provide recommendations [73]. For song recommendations, it was observed by the authors that the combination factor of lyrics and genre was not included for the recommendation [54]. Hence, they used RNN to predict the user's next song of interest-based on the similarity factor. Some authors analyzed that the recommendation quality of rating predictions in traditional systems had scope for improvement [82]. Hence, they proposed a Neural Rating and Tips generation (NRT) framework, which could predict accurate ratings and create conceptual tips and strong linguistic quality. In another work, the authors addressed the limitation of efficient methods in the e-learning area [81]. To improve the recommendations, the authors used Gated

Recurrent Unit (GRU) neural networks, a type of RNN. In another work, the authors deployed TARMF (Topical Attention Regularized Matrix Factorization) model that co-learned user and item details from ratings and customer reviews by optimizing matrix factorization and an attention-based GRU network [28]. To fully exploit and understand the user sentiments present in the review texts, the authors in this [147] paper implemented a deep learning-based neural network model, SDRA. Gated Recurrent Unit (GRU) is a type of Long Short Term Memory (LSTM) model, which is an extension of an RNN model. GRU is a much simpler version of the LSTM model; however, the parameter setting in GRU for better accuracy is much easier as compared to LSTM [143].

2.6.8.2 Findings and Open Issues

Recurrent Neural Network is efficient for using sequential data. Also, for subsuming side information like time, logs, etc., RNNs are beneficial. An important open issue is for music recommendation systems, the recommendations can be improved by performing temporal analysis of the features of music using Recurrent Neural Network, and fascinating characteristics of music can be uncovered by interpreting automatically learned features. Another scope is to explore Recurrent Neural Networks (RNNs) with bi interaction pooling to model sequential data. To further improve the performance of LSTM models, attention-based memory networks can be explored. The possibility of using a sequence-to-sequence (seq2seq) learning framework can be explored, aiming to create a convincing recommendation description to aid customers in making a better purchase. An important limitation of RNN that can also be considered as an open issue is that although applying the neighborhood approach in RNN can lead to good results, but it is also suggested that some baselines in recent research studies are not well justified and correctly evaluated. Gated Recurrent Unit primarily addresses the problem of the vanishing gradient. GRUs often combine several memories and gates to record sequential activities. Another advantage of this model is that it can perform rating prediction and create general tips with linguistic quality. This model efficiently stores contextual information for rendering user-item latent features into a brief sentence.

2.6.9 Hybrid Networks

2.6.9.1 word2vec and convolutional neural networks

The authors in [86] worked on a blog recommendation. They stated that due to a large number of blogs surfacing each day, it was crucial to recommend the right blogs to the right users. For this, they came up with the Boosted Inductive Matrix Completion method. Hence, they used the side information of users and blogs for adequate recommendations.

2.6.9.2 RNN and CNN

The authors worked towards improving quote recommendation. For this, first, they used RNN to model the tweet sequences, and then they used CNN for mapping tweets to intermediate vectors [78]. The authors in [69] observed that for POI recommendations, modeling multi-source information was one-dimensional. Hence, they proposed a Deep Context-aware POI Recommendation (DCPR) model, which consisted of different layers. One layer performed feature mining using Convolutional Neural Network; the second layer was based upon Recurrent Neural Network, and the third layer modeled these together. In another work [33], it was observed that the main challenge with news recommendation was to recommend the latest news articles to the users. To solve this issue, the authors presented an improved session-based Recurrent Neural Network (RNN) model, which studied the users' history of reading news articles and thus made recommendations. The authors in [93] used the Dual-Regularized Matrix Factorization technique, which consisted of a multilayered neural network model by simultaneously implementing a convolutional neural network and gated recurrent neural network, to create an independently distributed rendering of contents of users and items.

2.6.9.3 CNN and Stacked Denoising AutoEncoder

To address data sparsity, the authors used the Probabilistic model of the Hybrid Deep collaborative filtering model (PHD) and combined a stacked denoising autoencoder and a CNN alongside the auxiliary side information to extract users and items' latent factors [100].

2.6.9.4 Findings and Open Issues

Several deep learning-based recommendation methods employ more than one deep learning techniques. Deep neural networks give the system an edge over every other method by combining

numerous neural computations and complementing each other to form a more efficient hybrid model. Each combination of a neural network technique is very specific and is tailored to each use case, as every problem statement requires a unique set of neural network operations.

An open issue is exploring all the possible combinations of deep learning techniques since many combinations have not been exploited yet and may provide useful insight into the increasing efficiency of recommender systems.

2.6.10 Deep Reinforcement Learning

Deep Reinforcement Learning is the application of Artificial Neural Networks on Re- inforcement Learning. The working of reinforcement learning can be regarded as the learnings of a child. A toddler does not know the difference between right and wrong. He learns the outcomes of his actions through the environment. This course of learning pertains to the penalty and reward system [148]. If a child reaches towards a hot object, he feels his skin burn and immediately retracts his hand. This action was not right, so he received the skin burn as a penalty, and the next time when there is a hot object, and the child stays away from it, he does not burn, and this is his reward. So, the child learns from the penalty he received not to repeat that task, and the reward he received furthered him to stay away from hot objects, hence leading him on the right path. Similarly, in reinforcement learning, the agent learns from the environment. If the agent moves toward the goal, he is rewarded, making him conform to his direction. However, if the agent moves away from the goal, he is charged with a penalty, hence driving him to the right path [149]. There are a few terms involved in the conception of reinforcement learning:

- Agent: an agent is considered as the user of the system. He is the one who takes action to get to the result.
- Action: action is the step that the agent takes to reach his goal. It can be any possible move that the agent can make.
- Discount Factor: it is a factor of the future reward calculated by the agent for every action he takes. We aim to dampen this discount factor to make the user look at the bigger picture than take decisions based on the current future reward and get stuck in the local minima.
- Environment: it is the world in which the agent moves.

- State: the state is the instantaneous situation the agent finds himself in. Any state change is a result of the agent's action.
- Reward: reward is the positive outcome of the agent's action. The reward is considered to be always positive.
- Penalty: the penalty is the negative outcome of the agent's action. Whenever a penalty is imposed on the agent, he is assured that he is not moving towards the goal.
- Policy: policy is the rule or strategy the agent follows to reach the next stage. The policy used is based on the current state of the user.

The equation of reinforcement learning states the definition of the value function of the reinforcement learning algorithm. As stated in Equation 2.20, given the policy π, at any given state *s*, the function computes the average rewards offered by the actions, wherein every such action has a probability of moving to the next step *s'* with an immediate and a future reward.

$$V(s) = \sum_{t} \gamma^t R(S_t) \tag{2.20}$$

Here,
V(s) represents the function of future rewards coming from other states,
R represents the reward,
S_t represents the state at time *t*, and
γ represents the discount factor, where $\gamma \in (0 \leq \gamma \leq 1)$

Broadly, this equation can be written as the following Equation 2.21:

$$V_{\pi(s)} = \sum_{a} \pi(a|s) \sum_{s' \in S} \sum_{r \in R} p(s', r|s, a)(r + \gamma V_{\pi}(s')) \tag{2.21}$$

Here, *a* represents the action,
π represents the policy,

Function *p* represents the dependency on action *a*.

Authors in [150] worked on personalized course recommendations. The system trained a profile revisor and a recommender model to authorize the user profiles to be trained. With the help of a two-level task, the hierarchical reinforcement learning agent can efficiently remove the noisy course and filter out all the actual contributing courses to get the target course. NAIS (Neural attentive item similarity) model for the recommendation [151] is an item-based collaborative filtering algorithm that differentiates the weights of several historical courses by using an attention mechanism. NASR (Neural attentive session-based recommendation) [152] is an enhanced GRU model that evaluates the attention coefficient for every historical course depending upon the hidden vector output by GRU. The authors proposed an HRL+NAIS model that used NAIS as the primary recommender tool and combined hierarchical reinforcement learning based profile reviser. Similarly, the authors proposed another HRL+NASR model wherein, unlike above, the model adopted NASR as the primary recommender tool. In another study [153], the authors decreased the complexity of deep reinforcement learning for continuous control operations. To achieve this, the authors used an extension of Q-learning, a Normalized Advantage Function (NAF) derived instead of policy gradient and actor-critic methods. The authors in [148] aimed to simplify the learning of policies for the agent in environments with sparse feedback. They developed a hierarchical deep Q-network to combine hierarchical action-value functions operating at different temporal values. The authors in [149] realized the need for introducing multiple levels in hierarchical learning. Hence, they introduced a diversity driven extensible HRL (DEHRL) framework to achieve HRL with multiple levels. In another study [154], the authors worked towards combining the extensions of the DQN algorithm. To do this, the authors studied the aforementioned extensions of the DQN algorithm to analyze their combinations. Double Q-network [155], Prioritized Replay [156], Duelling networks [157], Distributional Reinforcement Learning [158] and Noisy Nets [159].

The authors in [160] worked towards user intent prediction. The system does not take into consideration the relationship between questions, which aids in maximizing the rewards. The system follows a greedy approach to carry out this task and eventually leads to pre-determining the user's queries. The authors implemented the N-step decision process to analyze the

interdependencies amongst the various queries. The authors in [161] reduced the uncertainty in user demands for personalized content prediction. To carry out this task, the authors divided the core problem into two different sub reinforcement learning problems. Both the subproblems were aimed at working towards a unified goal. In another study [162], the authors used deep reinforcement learning to determine the hidden links in the criminal network. The authors performed a Criminal Network Analysis (CNA) to identify the hidden links to pre-empt or disrupt illicit criminal activities. To perform this task, the authors compared the performance of CNA using Deep reinforcement learning with the performance of CNA using other machine learning algorithms like Gradient Boosting Machine (GBM), Random Forest (RF), and Support Vector Machine (SVM). The authors in [163] worked towards exploring the potential of deep reinforcement learning towards detecting lung cancer due to an increase in deaths due to lung tumors. To carry out this task, the authors performed several representative deep reinforcement learning models. The authors in [164] aimed at improving the performance of deep reinforcement learning recommender systems by employing pervasive social networks. Thus, the authors created a Social Attentive Deep Q-network (SADQN) agent to produce high quality recommendations in user-item interaction by using the social impact among users.

2.7 Chapter Summary

A literature survey has been performed on the subject of deep learning in recommender systems. It was observed that deep learning provides a considerable advantage in performance, especially when data is available in abundance. It was also observed that by using deep learning, we could extract feature representations that are much more comprehensive and better performing than the features extracted using traditional feature engineering. Deep learning can also be used to incorporate side information like time using models that can incorporate temporal data like LSTM. However, deep learning only works well when data is available in abundance. In situations where data sparsity exists, deep learning is not the best fit. Also, deep learning requires extensive hyperparameter tuning to achieve the desired accuracy. Quintessentially, hyperparameter tuning is a problem that plagues all machine learning algorithms, and deep learning models have a lot of additional hyperparameters making the problem even worse. Poor tuning can lead to overfitting or underfitting of data points. Deep learning is a great tool to exploit the ever explosive data available online and has shown to improve performance in all domains of recommender systems. However,

creating and training deep learning models is a sensitive process, and proper hyperparameter tuning is required to get the best results out of it. Deep learning also lacks interpretability, and often functions as a black box. Despite these limitations, deep learning has become an integral part of most state-of-the-art recommender systems and is increasingly becoming more relevant with the advent of big data.

Chapter 3

FOOD WINE RECOMMENDER SYSTEM USING PAIRWISE RECOMMENDATIONS

Food-wine pairing is an essential study in the culinary world and requires extensive research of the underlying food and wine pairing principles. To understand the principles of pairing, we have to understand the characteristics of food, wine, the interaction between them, and certain classic food-wine pairing norms that are religiously adhered to. This knowledge, as of today, is limited to the wine sommeliers and food experts. To take this understanding to a broader class of masses, there is a need for a recommender system that would take into account all the trivial and non-trivial details of food-wine pairing and the preferences of individual users to provide the perfect pairing. This adds another dimension to the recommendation since we have to consider which wine will pair with a particular dish, the first recommendation problem, and consider the user's preference for the pairing, the second recommendation problem. Because of this added complexity and unique nature of the problem at hand, there is a lack of literature on dealing with the additional dimension in the recommendation engine. Also, to apply these pairing principles in the context of recommender systems, we require abstract features such as flavor, aroma, major ingredients, type of meat, and many others. To achieve this, our recommender system relies on text mining techniques and sentiment analysis at its core to extract the relevant information from the text in the dataset. Such extensive feature engineering has not been undertaken before in the context of generating food and wine recommendations. We have used two publicly available datasets Yummly and Winemag, and have created two additional datasets by compiling information from various sources to help us with feature extraction. Thus, by designing a new recommendation framework and employing soft computing based text mining and sentiment analysis techniques, we have created a recommender system that embodies the age-old essence of wine paring, taking into account not only the food wine characteristics but also the user's preference of pairing (congruent or contrast). Finally, we evaluated our system by calculating the precision, recall, and

F1-score values on the dataset created by collecting the recommended food-wine pairings given by the wine sommeliers and food experts.

Section 3.1. comprises of the introduction to the research work done followed by Section 3.2 which includes a detailed explanation of the benchmark and self-generated datasets used. Section 3.3 describes the mappings based on different parameters like flavors and ingredients. Section 3.4 consists of implementing the system to generate pairwise recommendations. The results and analysis of the system si presented in Section 3.5 which is followed by the chapter summary in Section 3.6.

3.1. Introduction

The culinary industry is one of the largest industries in the world. Culinary connoisseurs, sommeliers, entrepreneurs, and restaurateurs spend billions in this industry to provide delicious meals and memorable experiences to the customers. Food- wine pairing is a fundamental science of the culinary world. A good pairing of food and wine has proved to increase sales in restaurants [165]. Studies have shown that wine is strongly associated with food across three dimensions: complementarity, social meaning, and lubrication of palate [166]. Even while shopping for wine, consumers consider food pairing [167]. Thus, to increase sales, retailers need to keep wines that pair well with the local cuisine of that region. It is also known that the most important factor determining a restaurant's wine menu is its food-wine paring [168]. The art of wine pairing depends upon various parameters like characteristics of food, user's personal preferences, and many more [169]. The categorization of wines is, but not limited to, white wines, red wines, rose wines, and sparkling wines. Various aspects are taken into consideration to understand the flavors of wine entirely. As explained in [170], the taste, aroma, texture, appearance, temperature, sensation, and geographical location are some of the factors that determine the quality of a wine. In addition to these factors, there are other features like astringency and tannins, which add a distinguishing feature in wines [171], [172]. Astringency is the rough, dry, or puckered feeling one has in mouth after sipping a strong wine. The cause of this dryness is due to the presence of tannins in the wine. Tannins are the leftovers of the stems, seeds, and skin of the grapes. The storage of wines in oak barrels also adds to the astringency of the wine [172]. A typical dish of any cuisine is said to have broadly six flavors, namely, spicy, bitter, sweet, savory or umami, salty, and sour

or acidic [173]. Complementing these flavors with the aromas and textures of wine requires the science of comprehensive mapping. The features of a dish depend not only on its dominant flavors but also on the protein used in the dish. There are some basic rules established in the science of food-wine pairing. For example, high salty and bitter foods go well with wines having high effervescence. So, Moscato d'Asti goes well with high salty and bitter foods, and Chardonnay and Champagne are considered to be a bad match [174]. Figure 3.1 shows the various characteristics of wine which determine the quality of that wine [170].

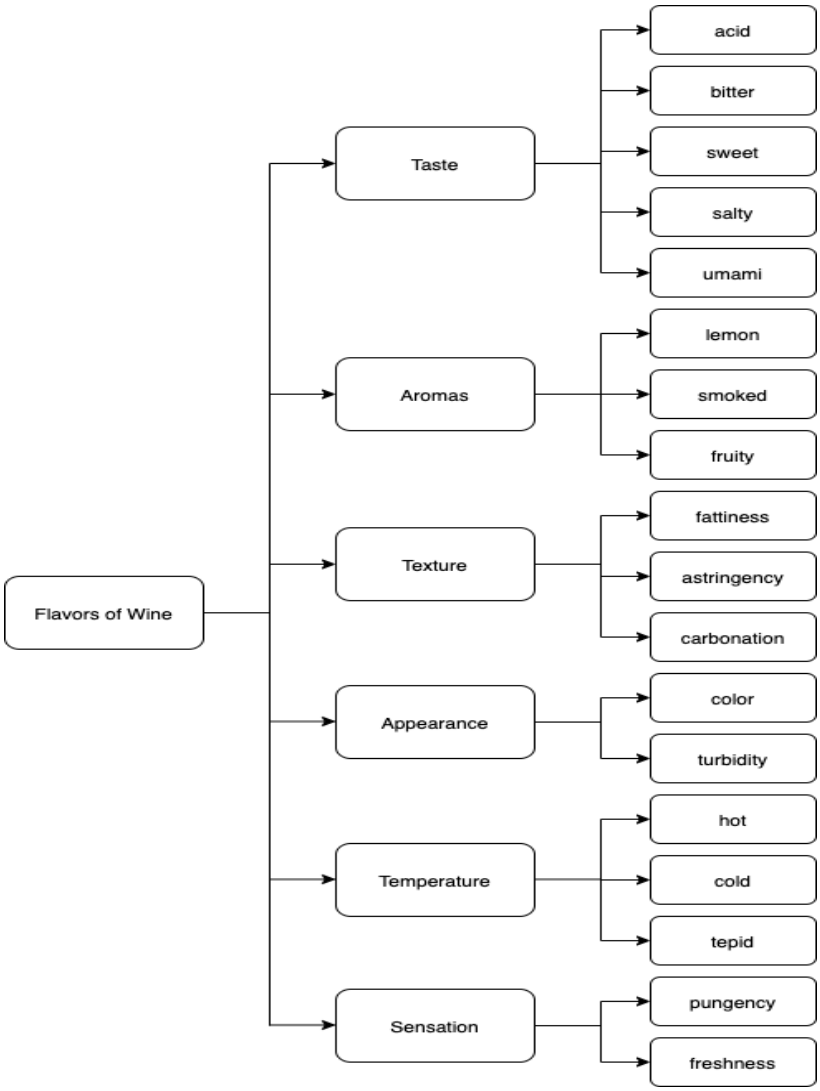


Figure 3.1 Characteristics of Wine

Though the concept of wine pairing is very old, minimal work has been done in making a computational system for the same where user preferences are also considered. Non-computational ways for pairing are not scalable given the advent of large datasets in any E-commerce setting. To illustrate, one of the wine retail websites, Winemag, lists 130K wines. The non-computational methods of charts and graphs used initially were reasonable when the choices of the wines were limited. However today, since the user has thousands of wines to choose from, there is a need for a computational system that can analyze millions of wines to find the perfect pairing. To apply the food-wine pairing principles, we require the knowledge of extensive features describing the dishes and wines. Without including such diverse yet complicated features, recommending the right wines would not be possible. In the past research studies, food and wine recommenders have not undertaken the task of feature engineering to such an extent. Our system relies on text mining and sentiment analysis to extract the required features from the given text data in Yummly and Winemag datasets. We have used the Yummly dataset in conjecture with two datasets that we compiled from various sources to extract features of dishes like flavor, main ingredients, nutritional information, and others. We trained a word2vec model on the corpus of reviews in the Winemag dataset to extract features of wines like acidity, flavor notes, tannin level, body, and others. With the transformed data and the new features as input, we have designed a novel recommendation framework that solves two recommendation problems. First, recommending wines for a given dish and generating ideal pairings. Second, personalizing the wine recommendations according to the user's personal preferences as some users may prefer to have a similar flavor of food and wine together, whereas some may prefer different flavors. Although there are a few commercial websites that recommend wines for given dishes, but the underlying literature, and the algorithm is nowhere described as per our knowledge. Hence, we aim at educating people about the importance of the problem of food-wine pairing and how we can ease the process of pairing for consumers, restaurateurs, retailers, and other stakeholders in the market.

3.2 Datasets

Two publicly available benchmark datasets are used to perform the study. All the food details, like the dishes, flavors, ingredients, etc. were taken from Yummly dataset, and all the wine details with the flavors, varietals, cost, etc. were taken from Winemag dataset. We also used two self-

compiled datasets. In the subsequent subsections, we analyzed the datasets in detail and realized their working.

3.2.1. Publicly Available Datasets

3.2.1.1 Yummly

We used the Yummly dataset, which consists of 28K unique recipes used across 19 unique cuisines. Each cuisine comprises of 200 to 200K recipes. We have used this dataset because of its vast collection of recipes, which consists of parameters like flavor, ingredients, nutritional information, etc. It is a publicly available benchmark dataset. Using this dataset, we can also deduce the similarities between different cuisines. Analyzing the similarities between the cuisines is a vital aspect since it gives us a scope to study the effect of similarity quotient of cuisines on wines pertaining to locations within proximity. In Figure 3.2, the clustering of cuisines based on ingredients is analyzed, and we can see that the similarity between the ingredients of cuisines is significantly determined by the geographical proximity of the nations [175].

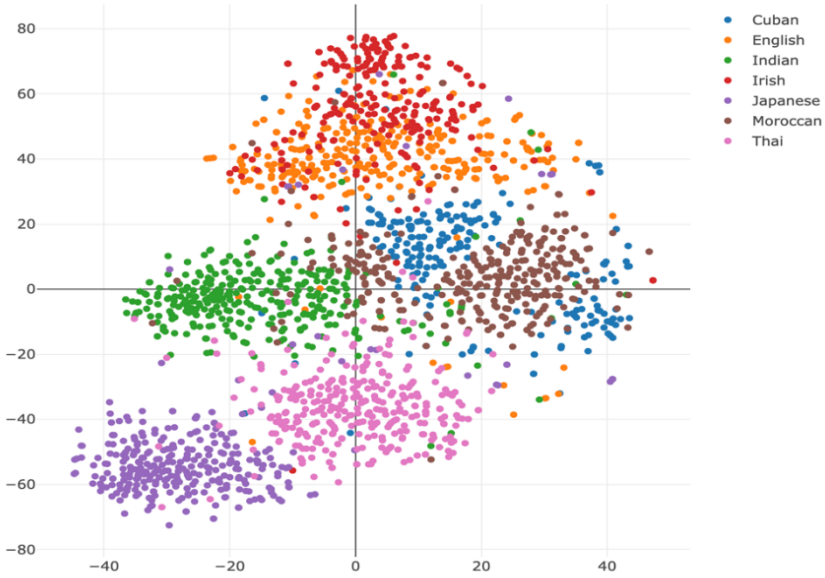


Figure 3.2 Clustering of cuisines by ingredients

Another way of finding similarities between the cuisines is by applying Logistic Regression and assessing the confusion matrix. It helped us in analyzing the confusion between the different cuisines.

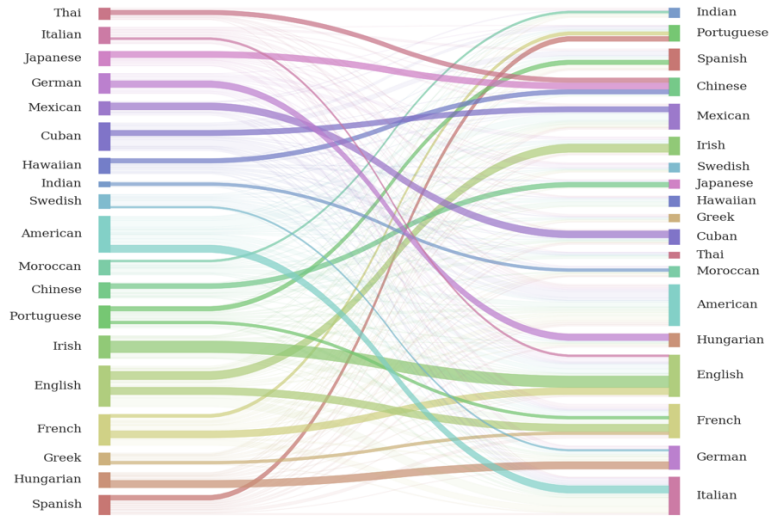


Figure 3.3 Analysing similarities between cuisines using confusion matrix

Hence, it can be seen from Figure 3.3 that Thai and Chinese cuisines are a lot similar compared to other cuisines. Many other pairs of cuisines can be pointed out. Hence, the similarity between cuisines depends largely on the geographical proximity of their native countries. Two of the most essential parameters which help in food-wine pairing are the flavour of the dish to flavour of the wine mapping and the ingredient of the dish to the flavour of the wine mapping.

3.2.1.1.1 Flavor Mapping

In the Yummly dataset, every dish was assigned one of the six different flavors, namely, salty, sweet, sour, bitter, meaty, and piquant. Any dish can be uniquely identified by its flavor. For e.g., desserts have a sweet flavor. Such details are very crucial in food-wine pairing as the flavor of the dish greatly determines which wine will go perfectly with it.

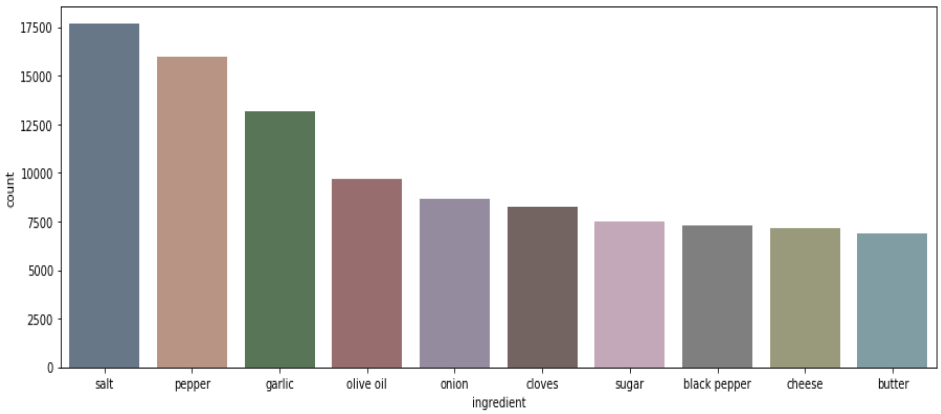


Figure 3.4 Ingredients used in Yummly dataset on the basis of flavors

Figure 3.4 describes the top ingredients which are used across all the 28,000 dishes in the Yummly dataset, and it can be seen that salt is the most commonly used ingredient. It is evident logically also since salt is used in every dish unless it is a dessert. Even in some desserts, salt is used to prepare some elements of the dish like salted caramel. Hence, pairing wine with food depends significantly on the flavor of the food.

3.2.1.1.2 Ingredient Mapping

Another critical feature to consider while pairing food with wine is the pairing of the main ingredient of that dish with the wine. Not all ingredients go with every wine, and it is essential to find out which wine goes with which ingredient [176]. Using the Yummly dataset, we analyzed the use of ingredients in every cuisine to understand how that affects their wine pairing. The ingredient we used here was chicken, and all the 19 unique cuisines were taken into account. In Figure 3.5, we can see the percentage of chicken used in each cuisine, and it can successfully be deduced that Thai and Spanish cuisines contain the most amount of chicken recipes and Chardonnay, Pinot Noir, and Zinfandel pair best with chicken [177]. Hence, these wines dominate the wine drinking culture in these cuisines.

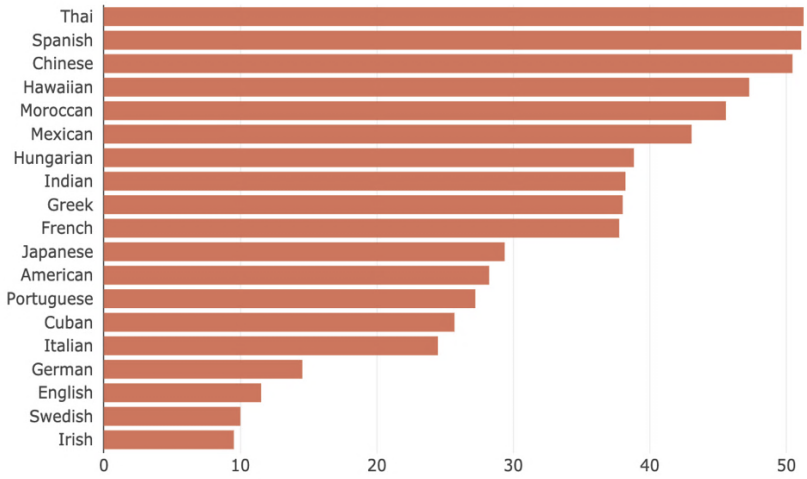


Figure 3.5 Percentage of chicken used in each cuisine

3.2.1.2 Winemag

For analyzing the parameters of wine, we used a publicly available benchmark dataset Winemag. This dataset consists of 130K reviews of wines, and it has a detailed expression of every wine. This dataset consists of 10 columns representing the detail of each wine review [178]. Following are the columns in the dataset:

1. Country: the origin country of the wine
2. Description: the description of the wine by the tester
3. Designation: the vineyard within the winery where the grapes that made the wine are from
4. Points: the score given to the wines by the wine enthusiasts on a scale of 0 to 100
5. Price: per bottle cost of the wine
6. Province: the state or province that the wine is from
7. Region_1: winemaking region in the province or state
8. Region_2: specific region of the wine growing
9. Variety: the variety of the grapes from which the wine is made
10. Winery: the winery which has manufactured the wine

We took into account the description and variety of wines and performed the mapping to understand the flavor notes of the wine and how they can be paired with the flavor of the dishes.

We performed some analysis on the wine dataset to understand it better and draw inferences. Firstly, we analyzed which countries provide the best wines by analyzing the country and points attributes. In Figure 3.6, we can see that England has the best average wine score, and hence, the English wines scored the maximum. Since it is a common fact that costly wines taste better than the cheaper wines, we analyzed this notion using our dataset. It can be seen in Figure 3.6 that better wines are costlier around the world. Hence, the general notion is justified.

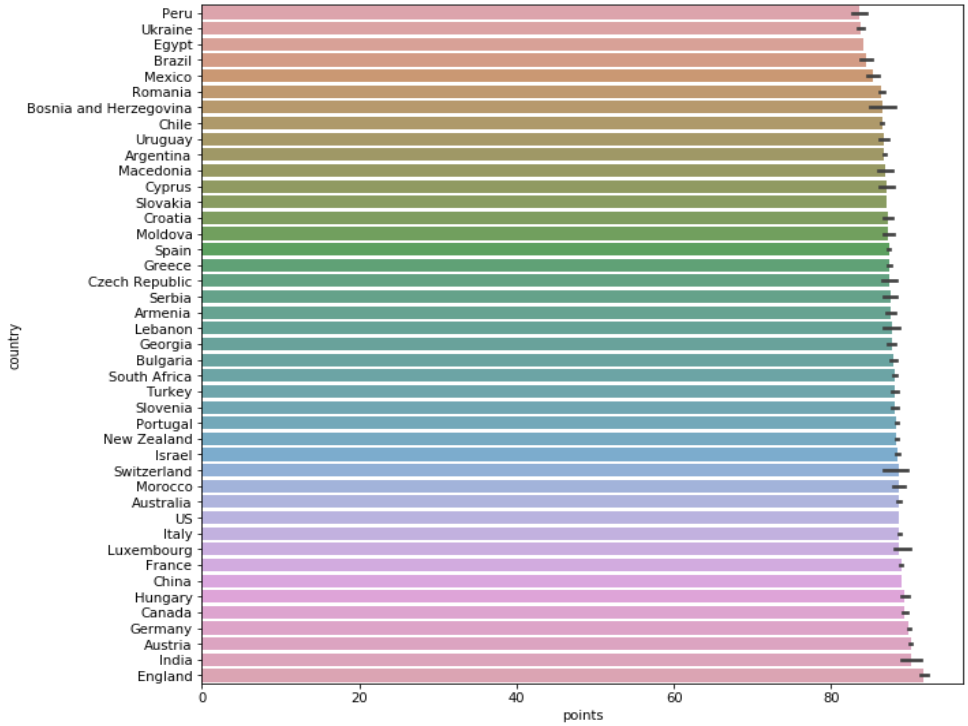


Figure 3.6 Wine points for each country

In another important analysis, we determined how the wine enthusiasts have ranked the wines based on flavors of the wine. It can be seen in Figure 3.7 that the food-related flavors used to describe the high ranking and low ranking wines are very similar, with a few exceptions. It is evident from the Figure 3.8 that the maximum of low rated wines contain the word drink, and the maximum of high rated wines contains the word fruit. It shows that the best wines are those which contain the fruity flavors and are liked the best by the wine enthusiasts.

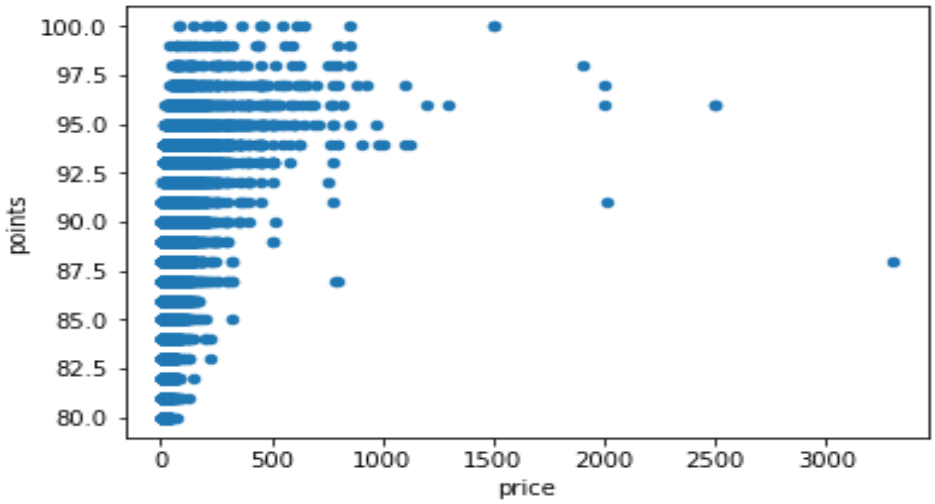


Figure 3.7 The plot against the price and points for wines

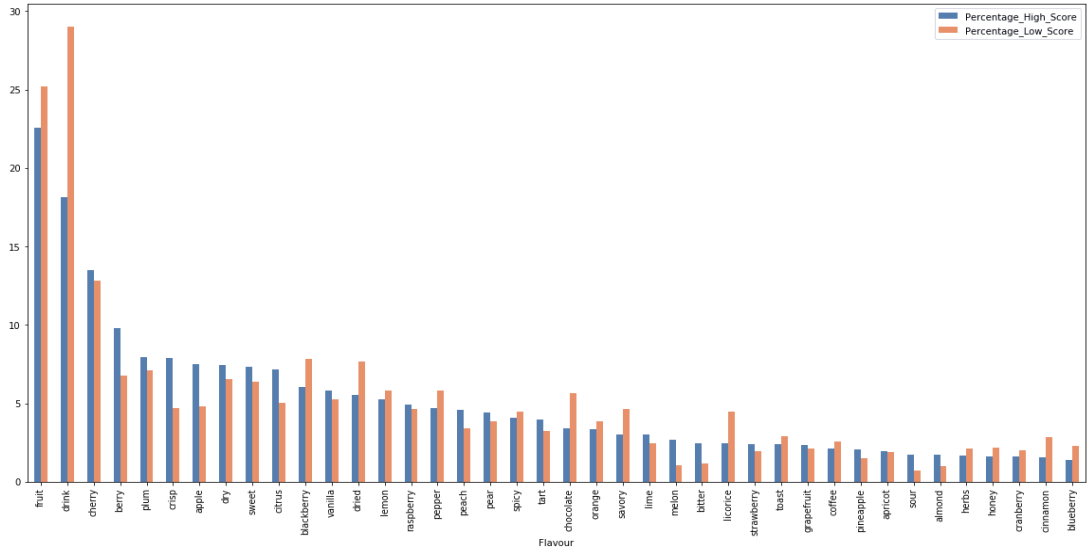


Figure 3.8 High and Low score percentage of wines on the basis of their flavors

3.2.1.2.1 Wine-Flavor Mapping

After creating the food-wine mapping based on the core ingredients of the dish, our next step is to create the food-wine mapping of the flavors used in both food and wine [171]. For food, we use the Yummly dataset, wherein the flavors for every dish are predefined. We obtained the flavor of each recipe from Yummly, and for the flavors of wine, we used the Winemag dataset, where, the description of every wine is given. We first grouped the wines by the type of grapes, region and variety. Then for each group, we considered the varietals having more than 30 reviews. Afterward,

we trained a word2vec model on the wine descriptions in the dataset. Later, we categorized the wine related terms to either one of the categories: sweet, sour, salt, spicy, bitter, or savory. For each wine, we calculated a TF-IDF weighted average embedding. After obtaining the embedding, since the flavor descriptors were unidimensional, we apply Principal Component Analysis (PCA) with one component and then normalized the resulting scalar between 0 and 1. So, at the end of the implementation, we received a flavor vector obtained by combining the six scalars for each flavor. The food-wine pairing principle adheres to certain previously established rules. In this study, we are referring to these rules as the flavor graph. This flavor graph determines which flavors are paired well together, i.e., harmonious flavors and which flavors are not paired together, i.e., discordant flavors.

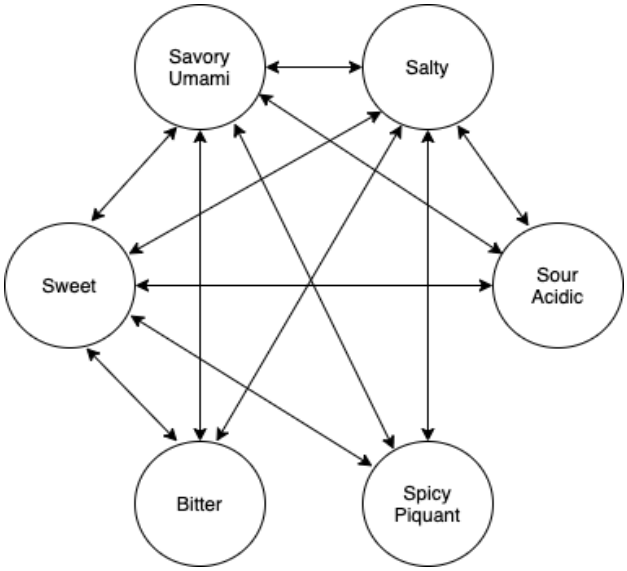


Figure 3.9 Flavor graph for food-wine pairing

It is evident from Figure 3.9 that all the flavors go with each other except sour/acidic, spicy/piquant, and bitter. These flavors do not pair well, and hence these flavors are discordant flavors. All the other flavor pairs which share the relationship in the figure are harmonious. We computed the TF-IDF weighted embedding on each wine descriptor to determine the flavors present in each wine. In this research work's context, we will discuss how crucial are the terms describing the flavor of the wine, i.e., the descriptors.

The term frequency or TF represents the total count of term *t* that appears in the description *d*. It is given by the following formula in Equation 3.1:

$$TF(t,d) = \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}} \tag{3.1}$$

The probability that a given description *d* contains the term *t*, then the relative description frequency is given by the Equation 3.2:

$$P(t|D) = \frac{|\{d \in D : t \in d\}|}{N} \tag{3.2}$$

Now, IDF can be defined as given in Equation 3.3

$$IDF = -\log P(t|D) \tag{3.3}$$

hence,

$$IDF = log \frac{N}{|\{d \in D : t \in d\}|} \tag{3.4}$$

Thus, IDF typically describes ow much information the term *t* provides across all the descriptions D. The inverse document frequency is denoted by Equation 3.3, which can further be written as Equation 3.4.

3.2.2 Self-Compiled Datsets

3.2.2.1 Ingredients Categorization Dataset

In this study, we performed an exhaustive study to identify the perfect ingredient-wine pairing because it plays a crucial role in our research. To do this, we first scraped the ingredients list from the Yummly dataset and identified the broad classification of these ingredients to make the implementation practical. Under the broad classification of the ingredients, we prepared a list of all the ingredients which come under the broad categories. It was done using various online food-wine pairing websites and research papers. The identified broad categories were Vegetables, Soft

cheeses, Hard cheeses, Pungent cheeses, Root Vegetables, Fish, Shellfish, Poultry, Red Meat, Processed Meat, Mushrooms, Herbs, Seeds and Nuts, Spices, Fruits and Berries, and Desserts. Figure 3.10 represents the number of ingredients in each category, and it can be seen that cheeses constitute the maximum amount of ingredients, and hence, cheese-wine pairing is a widely known concept.

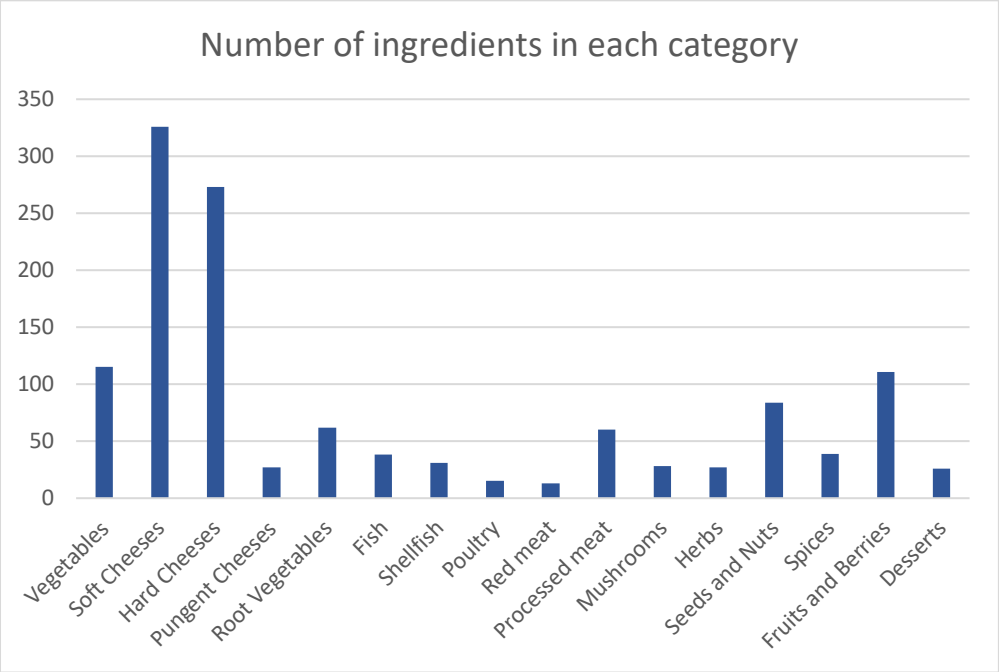


Figure 3.10 Number of ingredients in each category

While pairing food with wine, one of the essential aspects that are to be kept in mind are the core ingredients used in the dish [179]. Now the main concern is finding the core ingredient from the given ingredients list of the dish. Hence, we identified the main ingredient of the dish by analyzing the weights of each ingredient using the Yummly dataset. Any ingredient having 30% or more weight of the dish is taken into consideration and rendered as one of the core ingredients. Based on each ingredient thus obtained, we will map the perfectly paired wines to create a network. The ingredient-varietal pairing has been studied extensively using several research papers [172], [174], [176], [180]–[182] and a dataset for the same has been curated. After performing this step, we entailed the mapping of food and wine for the ingredients used in the dish.

3.2.2.2 Ingredient-Based Pairing Dataset

To carry out this research study, we curated a list of ingredients and wines based on their flavor and ingredient match. [176] suggested that the anecdotal beliefs of food and wine exist and are still preferred and supported. In Table 3.1, we have enlisted a subset of the ingredients-wine pairing so that the readers can get an idea of how this kind of pairing works. This subset has been scraped from four papers [174], [176], [180], [181]. This table consists of the basic food-wine pairing rules. For example, desserts go well with sweet wines, also called dessert wines like Ice wine [180].

Table 3.1 A subset of Ingredient-based pairing

Ingredient	Wine	Reference
Chocolate	Ruby Port	[176]
Goat's cheese	Sauvignon Blanc	[176]
Caviar	Champagne	[182]
Stilton Cheese	Ruby Port	[182]
Raspberry reduction sauce	Pinot Noir	[182]
Lobster	California Chardonnay	[182]
Beef	Sauvignon Blanc	[182]
Lamb	Sauvignon Blanc	[182]
Moderate salty food	Champagne	[174]
Highly salty food	Moscato d'Asti	[174]
Bitter food	Moscato d'Asti	[174]
Fish	Pinot Noir	[181]
Beef Bourguignon	Red Burgundy (Pinot Noir grapes)	[181]
Wood-grilled wild salmon	Oregon Pinot Noir	[181]
Dungeness crab	California Chardonnay	[181]

Tagliatelle	Barolo (red wine Nebbiolo grape)	[181]
Meaty Italian dishes	Nero d'Avola	[181]
Aussie BBQ	Australian Shiraz	[181]
Pork sausage	Beaujolais wine	[181]
Pork	Alsace Riesling	[181]
Bistecca alla Fiorentina	Chianti	[181]
Mushroom	Red Rioja	[181]
Roasted nuts or cheese	Ruby Port	[181]
Grilled anything	Zinfandel	[181]
Foie Gras	NY ice wine	[181]
Oysters	Riesling, Semillon and Sauvignon Blanc	[181]
Barbequed meat	Pinotage and Shiraz	[181]
Beef	Malbec	[181]
Feta cheese	Sparkling wine	[180]
Triple cream cheese	Oaked chardonnay, Fleur de Lis	[180]
Semi-soft cheese	Pinot Noir	[180]
Firm, Italian-style cheese	Full-bodied Meritage red (Bordeaux style blend)	[180]
Soft-ripened, aged goat's milk cheese	Ice wine	[180]
Cajun spice	Ice wine	[180]
Chevre or Goat cheese	Sauvignon Blanc	[172]

Brie	Chardonnay	[172]
Spicy Italian salami	Cabernet Sauvignon	[172]
Milk chocolate	Port Noir	[172]

3.3 Mappings Based On Flavor and Ingredients

First, we categorized all the ingredients in the Yummly dataset into broader categories. In this step, we curated a list of all the ingredients primarily used in cooking across all the cuisines. Later, we extracted the ingredients used in the dishes from the Yummly dataset. The next step encompassed manually mapping the dish ingredients with the wine. Afterward, we mapped the broad categories of ingredients with the flavors of wines and stored the results in an ingredient-pairing dataset. We obtained the flavors of the dish from the Yummly dataset. The flavor of the dish is described by a flavor vector comprising of six parameters corresponding to each dish. Next, we used our second dataset Winemag to extract the flavors for wines. The dataset consisted of 130K wine reviews given by the reviewers, and each wine was reviewed by more than one reviewer. After applying NLP on these descriptions, we successfully calculated the descriptors of each wine, which consisted of the flavour of the wine. In the next step, according to the classic harmonious and discordant flavors, we created a flavor graph that stated the food and wine flavors which paired well together (harmonious) and food and wine flavors which did not pair well (discordant). With the help of this flavor graph, we successfully removed the discordant flavour matches. We also asked the user for his preference for food and wine flavour liking, if he liked similar flavors in food and wine (congruent flavour pairs) or different flavors in food and wine (contrasting flavour pairs). Then, we performed ingredient-wine mapping. To execute this step, we utilized the broad ingredients categories and the flavors of wines. Using the knowledge of classic ingredient-wine pairing, we curated a table consisting of the mapping between the food and the wines. Afterward, we learn about his liking using the previous food and wine pairings. From this, we deduce the type of food-wine pair liking of the user. The user might like congruent pairings, contrast pairings, or both. In the final step, based on this selection, we filtered the wines pertaining to the specific dish, and afterward, we applied content-based recommender system to provide the user with similar wines which would pair perfectly with the dish according to user's like of pairing. To extract the

flavor of different wines from the Winemag dataset, we perform NLP on the description or the reviews of the wine. Here, we try to extract the keywords which determine the main flavor notes of the wine. In this dataset, several reviews are given by the wine enthusiast of different wines. Hence, every varietal of wine has at least one or more reviews. To determine the flavor notes of the wine efficiently, we require all the reviews of that wine. The flowchart of the system has been given in Figure 3.11.

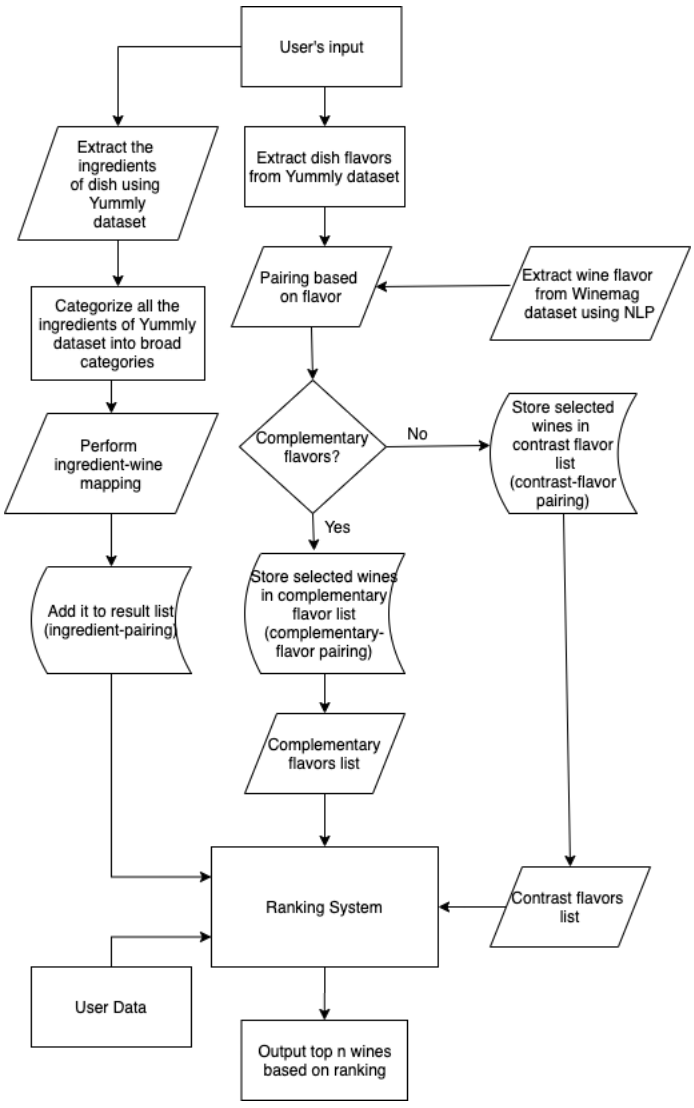


Figure 3.11 Flowchart of the food-wine recommender system

To retrieve the flavors of wine from the description of wines in the Winemag dataset, we applied the word2vec technique used in NLP. We have mapped the flavors of dishes with the flavors of the wine, and for this, we have used the cosine similarity [183]. We were given two non-zero vectors, i.e., the flavor of dish and flavor of the wine. To find the similarity between the two vectors, we found the cosine of the angle between the two vectors using the given equation. To determine the flavors of the wines, we have used CBOW (Continuous Bag-of-Words) model, which presents the flavors notes present int the wine from the description or reviews given by the experts.

3.3.1 Data Model

To describe the data model, we have introduced the mathematics involved in pre-processing the dishes, wines and the flavour profiles.

3.3.1.1 Dishes (d_i)

The mathematical representations of the dishes is given by Equation 3.5

$$d_i \in \{\ldots\ldots (set\ of\ all\ dishes) \ldots\ldots\}$$

$$d_i = \{w_i\alpha_i, \ldots\ldots, w_n\alpha_n\} \tag{3.5}$$

$$\sum w_i = 1 \text{ weights between di and } \alpha_i$$

$$\alpha_i \in \{\ldots\ldots (set\ of\ all\ ingredients) \ldots\ldots\}$$

3.3.1.2 Wine (Ψ_i)

The mathematical representations of the wines is given by Equation 3.6 and 3.7

$$\Psi_i \in \{\ldots\ldots (set\ of\ all\ wines) \ldots\ldots\}$$

$$\overrightarrow{\Psi_i} = [r_1, r_2, r_3, r_4, r_5, r_6] \tag{3.6}$$

where,

$$\sum r_i^2 = 1 \tag{3.7}$$

and,

r_1 = level of sweetness

r_2 = level of saltiness

r_3 = level of spiciness or piquantness

r_4 = level of acidity or sourness

r_5 = level of savory or umami

r_6 = level of bitterness

3.3.1.3 Flavor *(f_i)*

The mathematical representations of the wines is given by Equation 3.8 and 3.9

$$f_i \in \{\ldots\ldots (set\ of\ all\ flavors) \ldots\ldots\}$$

$$f(\overrightarrow{d_\iota}) = [\phi_i\ , \ldots\ldots, \phi_n] \tag{3.8}$$

where,

ϕ_i = weight of flavor f_i

Similarly.

$$f(\overrightarrow{\Psi_\iota}) = [\phi_i\ , \ldots\ldots, \phi_n] \forall\ \Psi_i \tag{3.9}$$

$$\exists\ a\ unique\ [\phi_i\ , \ldots\ldots, \phi_n]\ representing\ the\ flavor\ of\ wine$$

3.3.2 Cosine Similarity

The Euclidean Dot Product formula is stated by Equation 3.10:

$$A.B = \| A \| \| B \| cos\theta \tag{3.10}$$

Where, A and B are the two vectors and θ is the angle between the two vectors [184]. The similarity between these vectors is represented by the cosine of the angle. Hence, using Equation 3.11, the similarity is given by:

$$\cos(\theta) = \frac{A.B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.11}$$

After finding the similarity between the flavor vectors of dish and wine respectively, we found the similarity between the ingredients of the dish and flavor vector of the wine. Hence, applying Equation 3.12 in our work, we get,

$$\theta_d = \frac{f(d_i) \, . \, f(d_j)}{|f(d_i)| \, . \, |f(d_j)|} \tag{3.12}$$

where, θ_d is the similarity between the dishes

$$\theta_{ad} = \frac{f(\mathrm{d}_i) \, . \, f(\Psi_i)}{|f(\mathrm{d}_i)| \, . \, |f(\Psi_i)|} \tag{3.13}$$

where, θ_{ad} is the similarity between the dish and the wine as given by Equation 3.13. This similarity measure helps us in efficient food-wine pairing.

Initially, we categorized all the ingredients in the Yummly dataset into broader categories. In this step, we curated a list of all the ingredients primarily used in cooking across all the cuisines. Later, we extracted the ingredients used in the dishes from the Yummly dataset. The next step encompassed manually mapping the dish ingredients with the wine. Afterward, we mapped the broad categories of ingredients with the flavors of wines and stored the results in an ingredient-pairing dataset. We obtained the flavors of the dish from the Yummly dataset. The flavor of the dish is described by a flavor vector comprising of six parameters corresponding to each dish. Next, we used our second dataset Winemag to extract the flavors for wines. The dataset consisted of 130K wine reviews given by the reviewers, and each wine was reviewed by more than one reviewer. After applying NLP on these descriptions, we successfully calculated the descriptors of each wine, which consisted of the flavour of the wine. In the next step, according to the classic

harmonious and discordant flavors, we created a flavor graph that stated the food and wine flavors which paired well together (harmonious) and food and wine flavors which did not pair well (discordant). With the help of this flavor graph, we successfully removed the discordant flavour matches. We also asked the user for his preference for food and wine flavour liking, if he liked similar flavors in food and wine (congruent flavour pairs) or different flavors in food and wine (contrasting flavour pairs). Then, we performed ingredient-wine mapping. To execute this step, we utilized the broad ingredients categories and the flavors of wines. Using the knowledge of classic ingredient-wine pairing, we curated a table consisting of the mapping between the food and the wines. An example of such pairings has been given in Table 2. Afterward, we learn about his liking using the previous food and wine pairings. From this, we deduce the type of food-wine pair liking of the user. The user might like congruent pairings, contrast pairings, or both. In the final step, based on this selection, we filtered the wines pertaining to the specific dish, and afterward, we applied content-based recommender system to provide the user with similar wines which would pair perfectly with the dish according to user's like of pairing. To extract the flavor of different wines from the Winemag dataset, we performed Natural Language Processing (NLP) on the description or the reviews of the wine. Here, extracted the keywords which determined the main flavor notes of the wine. In this dataset, several reviews are given by the wine enthusiast of different wines. Hence, every varietal of wine has at least one or more reviews. To determine the flavor notes of the wine efficiently, we required all the reviews of that wine.

3.3.3 Training Word Embeddings

In Algorithm 3.1, we normalized the words present the wine reviews. This step is performed to remove punctuations and other trivial words from the reviews. This way, the text left to us contains all the important words representing the essential features describing the wines.

Algorithm 3.1: Training word embeddings

Input: wine_dataframe (Winemag dataset)
Output: word2vec model
Steps:
1. wine_reviews_list = list(wine_dataframe['Description'])
2. tokenized_sentences = tokenize(wine_reviews_list)
3. **for** s in tokenized_sentences:
    a. normalized _text = normalized_text(s)
    b. normalized_sentences.append(normalized_text)

4. word2vec.train(normalized_sentences)
5. **end**

3.3.4 Extracting Wine Features

Algorithm 3.2 consists of the steps used in extracting the wine features. In this implementation, we trained a word2vec model to identify relevant terms to determine the flavors of the wine. We have used the continuous bag-of-words (CBOW) model to perform this task.

Algorithm 3.2: Extracting wine features

Input: wine_data, model (word2vec)
Output: feature vector [sweet, sour, salt, spicy, bitter, savory] ∀ wg (wine groups)
Steps:

1. **for** v in wine_data[varietals]
    a. wine_data[varietals] = normalized_varietal(v)
2. **end**
3. varietal_geographies = [‘variety’, ‘Subregion’, ‘region’, ‘province’, ‘country’]
4. wine_group = []
5. **for** d in wine_data
    a. wine_group[varietal_geographies].append(d.review())
6. **end**
7. **for** wg in wine_group
    a. **if** size(wg.review) < 30
    b. wine_group.remove(wg)
    c. **end if;**
8. **end**
9. flavor_vector_list = []
10. **for** wg in wine_group
11. [sweet, sour, salt, spicy, bitter, savory] = model.computeEmbedding(wg.reviews)
12. **for** f in [sweet, sour, salt, spicy, bitter, savory]
    a. f = PCA(f,dimenation = 1)
    b. f = normalize(f,min=0,max=1)
    c. flavor_vector_list.append((wg, [sweet, sour, salt, spicy, bitter, savory]))
13. **end**
14. **return** flavor_vector_list;

The input vector, or the context is given by x^c and the output is given by y^c or y. Two matrices are created. Input matrix, v is given by $v \in R^{n \times |D|}$ where, n is the size of the embedding space. The output matrix μ is given by $\mu \in R^{|D| \times n}$. For given context of size m, the vectors are given as $\left(x^{(c-m)}, \ldots, x^{(c-1)}, x^{(c+1)}, \ldots, x^{(c+m)}\right)$. The embedded word vectors are given by Equation 3.14

$$\left(v_{(c-m)} = vx^{(c-m)}, v_{(c-m+1)} = vx^{(c-m+1)}, \ldots\ldots, v_{(c+m)} = vx^{(c+m)}\right) \quad (3.14)$$

The average of the vectors given is Equation 3.15 can be denoted as,

$$\theta = \frac{v_{c-m} + v_{c-m+1} + \cdots . + v_{c+m}}{2m} \quad (3.15)$$

The score vector, z is given by z = μθ

The probabilities of these scores is given by:

$$\hat{y} = softmax(z) \quad (3.16)$$

Using Equation 3.16, the cross entropy H$(\hat{y}, y)$ is given by the formulation of the loss function:

$$\mathrm{H}(\hat{y}, y) = -\sum_{j=1}^{D} y_j \log(\hat{y_j}) \quad (3.17)$$

The above-mentioned Equation 3.17 can also be simplified as given by Equation 3.18

$$\mathrm{H}(\hat{y}, y) = -y_i \log(\hat{y_i}) \quad (3.18)$$

After estimating the distance using the cross entropy, the objective function can be given as

$$minimize\ J = -\log P(w_c | w_{c-m}, \ldots\ldots, w_{c-1}, w_{c+1}, \ldots\ldots, w_{c+m}) \quad (3.19)$$

$$= -logP(u_c|\theta)$$

The final representation of the objective function as given is Equation 3.19 can also be written as

$$= -log \frac{\exp(u_c^T \theta)}{\sum_{j=1}^{|D|} \exp(u_j^D \hat{v})}$$

$$= -u_c^D \theta + log \sum_{j=1}^{|D|} \exp\,(u_j^D \theta) \tag{3.20}$$

Hence, the formula for the objective function can be seen in Equation 3.20.

3.4 Generating Pairwise Recommendations

After performing all the pre-processing, in Algorithm 3.3, the steps for building the recommender system are explained.

Algorithm 3.3: Recommender Engine

Input: d (dish)
Output: top n recommendations
Steps:
1. major_ ingredients = []
2. **for** ingredient in Yummly.getIngredients(D)
    a. **if** ingredient.weight > D.weight*0.30
    b. major_ ingredients.append(ingredient)
    c. **end if**
3. **end**
4. flavor_vector = Yummly.getFlavor(D)
5. wines-based-on-ingredient = wine- ingredient-map[ingredient]
6. congruent-wine = getCongruentWines(flavor_vector)
7. contrast-wine = getContrastWines(flavor_vector)
8. recom_list = ranking(contrast-wine, congruent-wine, wines-based-on-ingredient, U)
9. return recom_list.getTop(n)

The pseudo-code of the ranking algorithm used for ranking the recommendation list of the wines to the user has been provided in Algorithm 3.4.

Algorithm 3.4: Ranking

Input: contrast-wine (cn_w), congruent-wine(cg_w), wines-based-on-ingredient(wi), U (user data)
Output: recommendation_list (list of wines)
Steps:
1. w1 = wi ∩ cn_w
2. w2 = wi ∩ cg_w
3. w3 = w1∪ w2
4. major_flavor = []
5. **for** f in d.flavor :
    a. **if** f > 0.3

b. major_flavor.append(f)
c. **end if**
6. **end**
7. wu = getWinesLikedByUserForFlavor(major_flavor)
8. w4 = cn_w ∩ cg_w
9. w5 = w4
10. score_list = []
11. **for** w in w4:
a. **for** w' in wu:
b. score = max(cosine_sim(w,w'), score)
c. **end**
d. score.scale(min=0, max=75)
e. **if** w in w3
f. score_list.append(w,score + 25)
g. **else**
h. score_list.append(w,score)
i. **end if**
12. **end**
13. recommendation_list = score_list.sort()
14. return recommendation_list

3.5 Results and Analysis

In this section, we will discuss the implementation of our algorithms and the results obtained. After generating recommendations to the user for the dish, we collected seven datasets from online sources and food-wine pairing experts and sommeliers. Each dataset mentioned the classic food and wine pairings, and using these observations as data points, we analyzed our obtained results. To perform the analyses, we used precision, recall, and F1-score as these metrics were the most relevant to our implementation and results The formulae for all the three metrics have been given below:

$$Precision = \frac{\#\ of\ recommendations\ that\ are\ relevant}{\#\ items\ we\ recommend} \quad (3.21)$$

$$Recall = \frac{\#\ of\ recommendations\ that\ are\ relevant}{\#\ all\ possible\ relevant\ items} \quad (3.22)$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3.23)$$

For determining the precision of our recommender system, we calculated the number of relevant recommendations for all the recommended items, as seen in Equation 3.21. For determining the recall metrics, we calculated the number of relevant recommendations for all the possible relevant items, as seen in Equation 3.22. In the Winemag dataset, more than one wine exists for each varietal. It has made the wine data very large. In our recommendation results, we are recommending users up to a maximum number of eight wines. Since our number of recommendations is very less, the recall hence obtained, is very small according to the formula in Equation 3.23. To better optimize the recall metric, we need to either increase the number of recommendations or increase other filters or parameters like cost, region, winery, etc. to skew the data of relevant items further. Using the formula given in Equation 3.10, we have calculated the F1-score metric. In our system, we used collected seven datasets (D1, D2, D3, D4, D5, D6, D7) from online sources of food and wine experts stating the conventional food and wine pairs. Keeping these data points as a benchmark, we analyzed our results. For each dish entered by the user, a maximum number of eight wines are recommended by the system. Hence K = 8. Table 3.2 enlists the precision metrics for all the seven datasets and Figure 3.12 represents these values graphically. Similarly, Table 3.3 represents the Recall metrics and represents it graphically in Figure 3.13 and Table 3.4 lists the F1-Score values of the seven datasets and Figure 3.14 represents that graphically.

Table 3.2 Precision metrics of food-wine recommendations

K	D1	D2	D3	D4	D5	D6	D7
1	0.31	0.223	0.312	0.264	0.389	0.351	0.355
2	0.341	0.289	0.365	0.421	0.557	0.494	0.372
3	0.397	0.334	0.418	0.482	0.592	0.613	0.482
4	0.465	0.458	0.495	0.573	0.648	0.653	0.526
5	0.542	0.519	0.572	0.743	0.691	0.673	0.669
6	0.618	0.573	0.723	0.792	0.732	0.724	0.718
7	0.677	0.651	0.752	0.829	0.781	0.802	0.774
8	0.775	0.745	0.773	0.852	0.819	0.843	0.841

Table 3.3 Recall metric of food-wine recommendations

K	D1	D2	D3	D4	D5	D6	D7
1	0.0006	0.0004	0.0006	0.0005	0.0008	0.0007	0.0007
2	0.0014	0.0012	0.0015	0.0017	0.0022	0.002	0.0015
3	0.0024	0.002	0.0025	0.0029	0.0036	0.0037	0.0029
4	0.0037	0.0037	0.004	0.0046	0.0052	0.0052	0.0042

5	0.0054	0.0052	0.0057	0.0074	0.0069	0.0067	0.0067
6	0.0074	0.0069	0.0087	0.0095	0.0088	0.0087	0.0086
7	0.0095	0.0091	0.0105	0.0116	0.0109	0.0112	0.0108
8	0.0124	0.0119	0.0124	0.0136	0.0131	0.0135	0.0135

Table 3.4 F1-score metric of food-wine recommendation

K	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**
1	0.0012	0.0008	0.0012	0.001	0.0016	0.0014	0.0014
2	0.0028	0.0024	0.003	0.0034	0.0044	0.004	0.003
3	0.0048	0.004	0.005	0.0058	0.0072	0.0074	0.0058
4	0.0073	0.0073	0.0079	0.0091	0.0103	0.0103	0.0083
5	0.0107	0.0103	0.0113	0.0147	0.0137	0.0133	0.0133
6	0.0146	0.0136	0.0172	0.0188	0.0174	0.0172	0.017
7	0.0187	0.0179	0.0207	0.0229	0.0215	0.0221	0.0213
8	0.0244	0.0234	0.0244	0.0268	0.0258	0.0266	0.0266

We have further graphically analyzed our results obtained. We have represented the most important metric in our system, i.e., precision in the graphs below. It can be seen from the graphs that by the end of the eighth recommendation, the precision becomes the highest, which indicates that as the recommendations of wines are made, the results keep becoming better and precise. To address the computational challenge of data sparsity in this problem, we created self-generated datasets and performed sentiment analysis on the wine reviews to generate efficient recommendations.

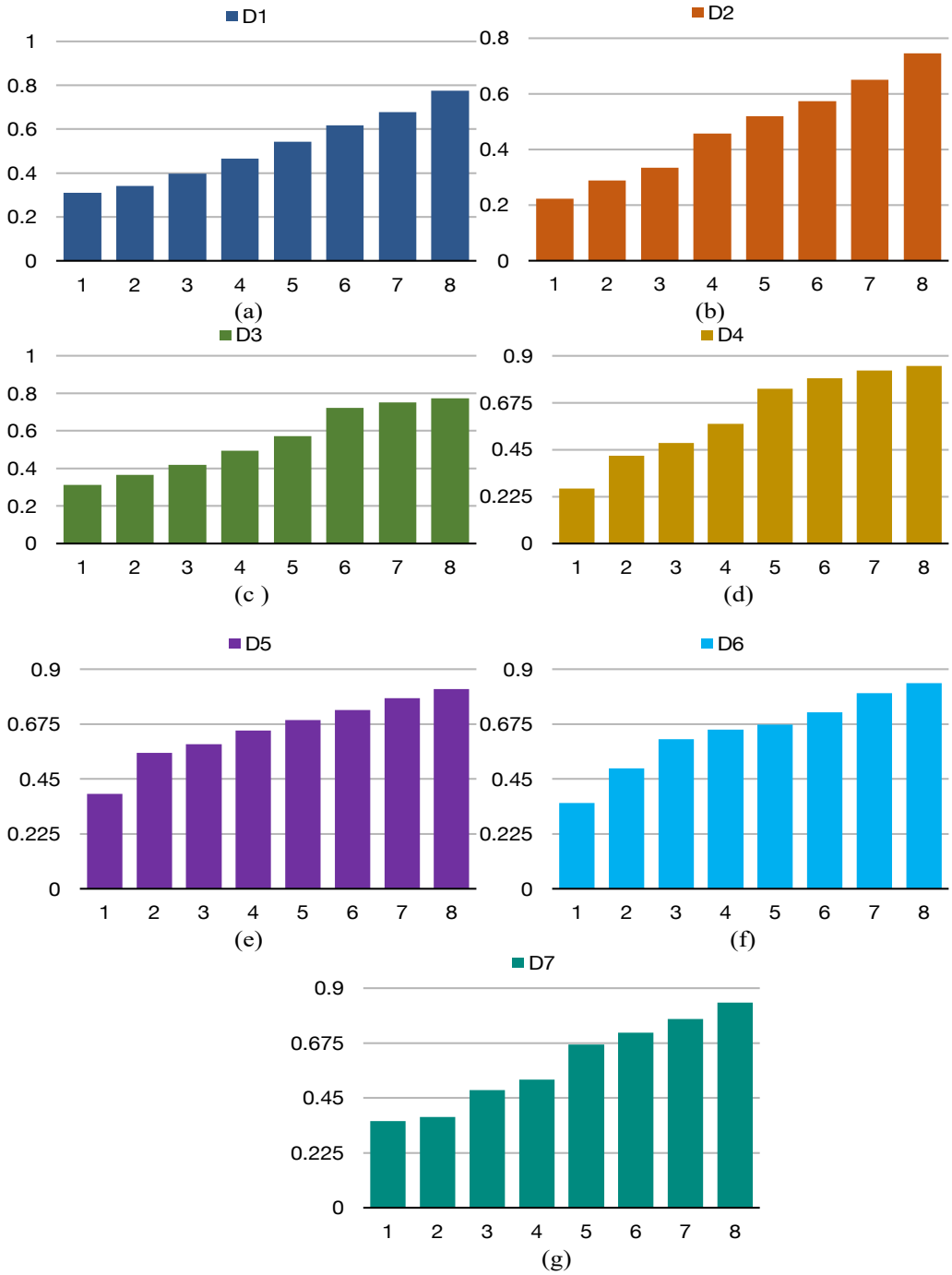


Figure 3.12 (a) Precision metrics for dataset D1 (b) Precision metrics for dataset D2 (c) Precision metrics for dataset D3 (d) Precision metrics for dataset D4 (e) Precision metrics for dataset D5 (f) Precision metrics for dataset D6 (g) Precision metrics for data D7

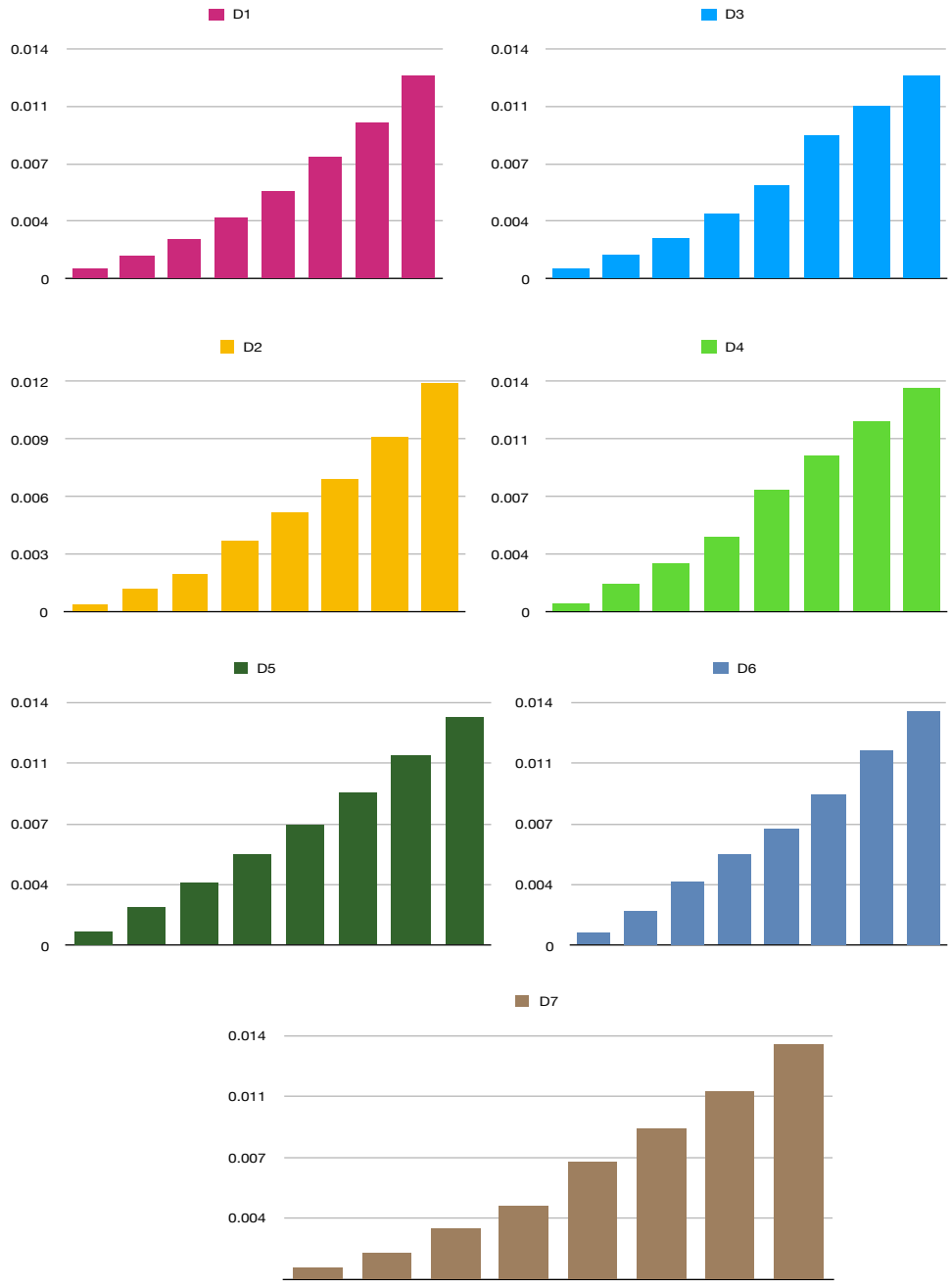


Figure 3.13 (a) Recall metrics for dataset D1 (b) Recall metrics for dataset D2 (c) Recall metrics for dataset D3 (d) Recall metrics for dataset D4 (e) Recall metrics for dataset D5 (f) Recall metrics for dataset D6 (g) Recall metrics for dataset D7

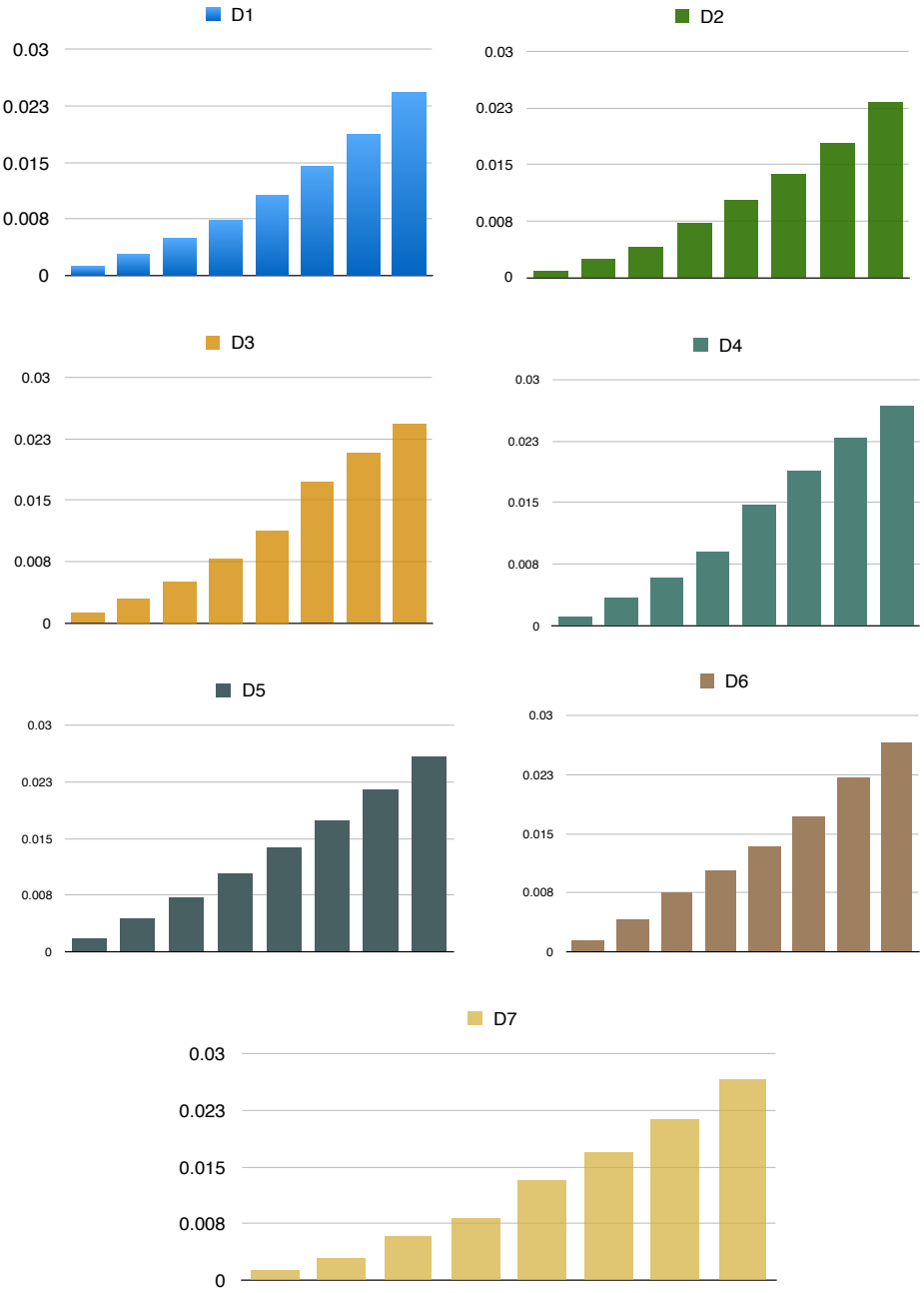


Figure 3.14 (a) F1-Score metrics for dataset D1 (b) F1-Score metrics for dataset D2 (c) F1-Score metrics for dataset D3 (d) F1-Score metrics for dataset D4 (e) F1-Score metrics for dataset D5 (f) F1-Score metrics for dataset D6 (g) F1-Score metrics for dataset D7

3.6 Chapter Summary

Food-wine pairing is an important science that requires extensive knowledge about flavor principles. Wine sommeliers and food experts are well aware of the fundamentals involving food-wine pairing, but to take this knowledge across, everyone requires a computational system that could recommend wines to the user with respect to several parameters. In this study, we created a food-wine recommender system using a novel framework designed to tackle the problem of handling two layers of recommendation. The underlying data used by our recommender system was derived by text mining and sentiment analysis operations on Yummly and Winemag datasets used in conjuncture with two datasets that we self-created. Soft computing techniques like word2vec allowed us to model abstract features required for the recommender system. We created a system that was able to apply the principles of food-wine pairing to thousands of dishes and wines to instantly generate pairings consistent with the principles while considering user preference. Although we have done extensive feature engineering, most of our features are derived from reviews and are abstract, like flavor feature and aroma feature as there is no quantitative source for this data. It is possible to improve the quality of recommendations if data for the features become available quantitatively. Our system is based on content-based recommendation, and we have not studied it in a social network setting or multiple users. Such exploration can lead to a better hybrid recommendation model incorporating collaborative filtering, graph-based recommender systems, and much more.

Chapter 4

USING ENSEMBLE LEARNING TO GENERATE EFFICIENT RECOMMENDATIONS

With the explosive increase in data on the web, recommending items to users is becoming more complex. In recent times, the best recommender systems have come from ensemble learning, which combines many models and techniques to generate recommendations that can draw the best characteristics of the constituent models. These ensemble models can improve accuracy, and they are also able to reduce the biases that come with each model. The state-of-the-art recommender systems currently rely on ensemble learning techniques to produce the best results. This can also be seen by the recommender system that won the famous Netflix competition was also an ensemble to many individual recommenders.

Section 4.1 presents the basics of ensemble learning. Section 4.2 describes how ensemble learning affects the recommender systems. Section 3 covers the optimization part of the process wherein the results achieved are further optimized using evolutionary algorithm. Section 4.4 describes the results produced and its analysis and the chapter wraps up with summary in Section 4.5.

4.1. Ensemble Learning

The field of Recommender systems is ever-growing and promises tremendous potential in the world of digital computations, whether in business, education, and much more. The implementation of recommender systems has become inevitable in today's world. Suggesting users or consumers which item to choose from an explosive range of products available to them often becomes cumbersome pertaining to several factors. It can be the context of recommendations, the size of the dataset, the type of dataset, the domain of implementation, and much more. All such factors require a vigilant approach towards identifying the most appropriate Machine Learning (ML) techniques to follow to get the most efficient results in the least possible time. With the advancement in technology, the usage of deep learning for computations in the real

world is ever booming. Implementing deep learning with numerous neural layers is a difficult task, and identifying the best-suited technique from a pool of abundant options is just challenging [185].

As it is famously said that time is money, every computational implementation aims at minimizing the time complexity of the algorithms used. Manually deciding which ML techniques to go with and setting up their input parameters and other technical operations take a toll on the time complexity of the model. Hence, we take several such ML techniques, also called ensemble classifiers which help in solving a complex computational problem. While solving a problem with an ML technique, the algorithm, however efficient, tends to make mistakes that generate false positives and false negative results. This often decreases the accuracy of the system. Hence, authors [186] came up with the unique solution of creating ensembles, wherein all the techniques that are eligible to provide a solution to the problem at hand are kept at a common ground of computation ensembles are created using those techniques. These ensembles generate results that are added or combined to reach a common consensus of results providing the best result sets. The number of ensembles formed depends upon the number of ML techniques used to solve the problems, and the more the techniques, the higher number of permutations and combinations are formed. For improving movie recommendations, [187] implemented the boosting technique of ensemble learning. In another research, [188] combined the content-based and collaborative recommender system by creating an ensemble of both techniques using a hierarchical Bayesian approach. Authors in [189] created an ensemble regressor for the prediction of missing ratings in recommender systems. Similarly, in another study [190], the authors created an ensemble of unimodal generated rankings, hence creating multimodal interactions resulting in improved recommendations. [191] used a multi-view ensemble system called MV-DEM for detecting shilling attacks using base classifiers. Authors in [192] created an ensemble of matrix decomposition methods combined with SVD++ to study its effects on collaborative filtering. To study and remediate the problem of cold start in recommender systems, authors in [193] applied Gradient Boosting Decision Tree ensemble learning on access logs. In another study concerning online product recommendations, authors [194] implemented three ensemble approaches based on user feedback's multimodal interactions. To improve music recommendations, authors in [195] applied a bio-inspired cluster ensemble of swarm intelligence and fuzzy clustering on UBCF (user-based collaborative filtering). Wu [196] created an ensemble of regularized, maximum margin,

and nonnegative matrix factorization on collaborative filtering. A few different applications of ensemble learning implemented in several other domains are [197]–[202].

4.2. Effect of Ensembles on Recommender Systems

The science involved in generating recommendations comprises complex computational problems like machine learning, deep learning, and others. As discussed before, a recommender system using just one Machine Learning technique suffers from one or more limitations. Hence, it is crucial to use an array of techniques. Generating efficient ensembles is crucial to mitigate the shortcomings of the individual recommender techniques. However, creating an ensemble is a challenging task on its own. There are endless ways to combine techniques and their outputs. For an ill-defined model, combining the base techniques in the search space for the parameters can be very large and require extensive training data and a lot of time to optimize. To solve this problem, we use a hierarchical model which, without blowing up the search space, offers us efficient ways to combine the base recommenders. The second challenge is optimization. Even with the right model, the search will still have a large number of dimensions across which we are required to search for the optimal ensemble [203].

4.2.1 System Architecture

The massive number permutations in which recommendation models are combined to create an ensemble model, adds another layer of complexity to an already complex problem. Thus, there was a need for a machine learning framework that can learn the best ensemble model for a given problem given the base models. We proposed a system EnPSO which intelligently optimized the recommendations by identifying the best ensemble architecture for the data at hand. Our proposed AutoML system can improve recommendations for MovieLens dataset by combining the results from base techniques. The employment of the system starts from uploading the MovieLens dataset and then performing data cleaning to filter out the columns which are essential to us. This can also be referred as feature extraction where the system extracts all the features or columns from the dataset which are relevant to our system. In this study, one of the most crucial columns is movies. In this column, all the movies present in the dataset are listed. After extracting this feature, the EnPSO checks for NULL rows. All such NULL rows are discarded. Afterward, the system extracts the feature rating. Similarly, it checks for NULL rows and discards such data points. Similarly, the

system extracts the feature tags and performs feature extraction on this column values. After all the data cleaning and feature extraction, EnPSO applies all the recommendation classifiers i.e., POPULAR, RANDOM, IBCF, UBCF and SVD. After this, the ensembles are created and the generation of the ensembles is performed using hierarchical ensemble technique about which we discussed earlier. Then the EnPSO optimizes the ensemble using Particle Swarm Optimization (PSO) and checks whether the ensemble is optimized or not. After generating the optimized ensemble, the content-based recommender engine is applied by the EnPSO to generate recommendations. There are five different techniques used IBCF, UBCF, POPULAR, RANDOM, and SVD. An ensemble structure ENSEMBLE is created, which takes the set of techniques and the set of ensemble methods as the parameters. In the later step, Particle Swarm Optimization (PSO) is applied to function *F* for identifying the most optimum recommendation results. Finally, we get the results and the ensemble structure, which provided the most optimum results. The architecture of the system is shown in Figure 4.1.

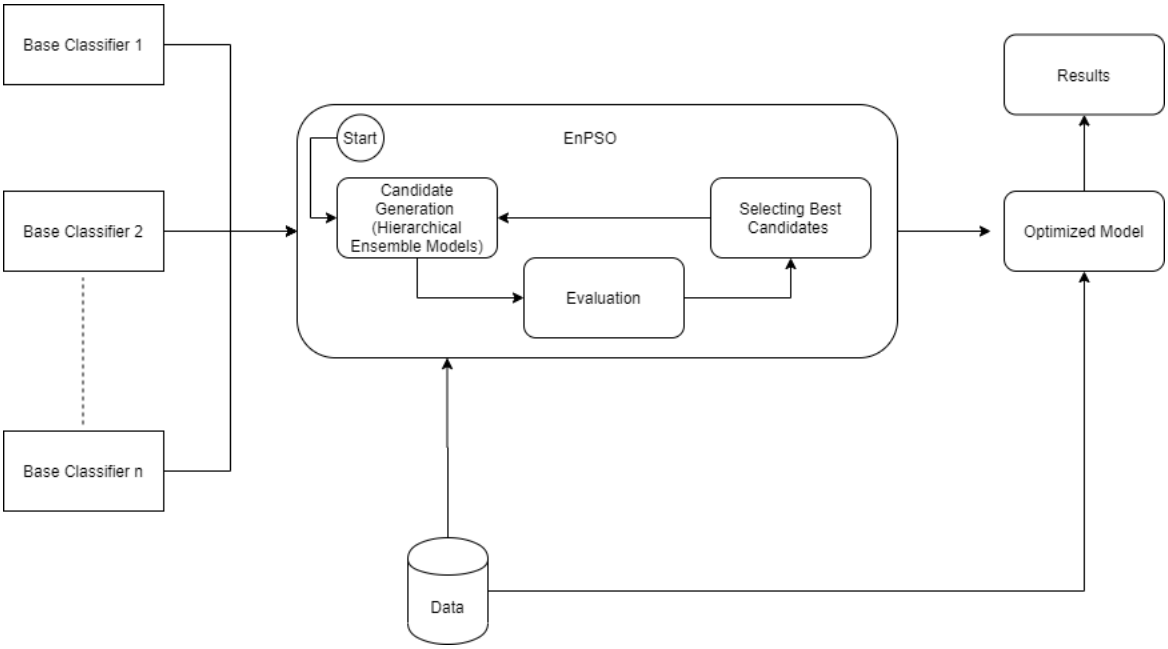


Figure 4.1 Architecture of the proposed AutoML ensemble recommender system

4.2.2 Base Recommenders

To prevail over the limited efficiency of the recommender systems, we incorporated ensemble learning in our study. Ensemble Learning is a technique wherein multiple algorithms are trained,

and their outputs are combined, forming a "committee." The formation leads to the calculation of results in such a way that the most optimal result is obtained by the end of implementing the ensemble [204]. The ensemble method primarily increases the effectiveness of the model. Weights are assigned to the outputs of multiple classifier systems, and the opinions and results of other individual decisions by the classifier are combined. Hence, the final result of the ensemble learning of these classifiers provides the most optimum result. The ensemble model designs itself in such a way that the risk of making a weak solution is avoided. For example, for undergoing surgery, the patient consults a few doctors and then takes the decision which is best for him. This decision that he makes depends upon the prescriptions provided by each doctor. Similarly, in ensemble learning, different techniques provide different results, and certain weights are assigned to the results. According to these results, the ensemble optimizes the output of the recommender by combining the results of base classifiers in an optimal way. It can also be said that in ensemble learning, different training parameters for each classifier generate different decision boundaries. The following are different benchmark techniques for making recommendations that we have used in our study to compare with our proposed algorithm EnPSO.

4.2.2.1 Popular Items

The first technique that we used is based on popular items. In this technique, a vast pool of popular items, i.e., the items with maximum positive reviews, are maintained. The users are recommended items from this pool so that they are recommended the most highly rated items, and the probability of purchase increases. This technique is best in the case of cold-start users [205]. The cold-start problem in users is one of the most common problems witnessed in recommender systems. Whenever a new user is introduced, the recommender system does not have any idea about the choice and liking of the user [206]. So, it is very cumbersome for the recommender system to recommend appropriate items. Thus, given that case, this technique of recommending the most popular items is used, and the chances of the user liking the items increase.

4.2.2.2 Random Items

In this technique, depending on the size of the dataset, random data points are selected from the entire pool of data. This dataset consists of random items. When recommendations take place from this dataset, items are recommended at randomly to the users. This recommendation technique is

primarily used whenever the problem of cold-start occurs for items. Whenever a new item is launched in the market, it gets challenging to label that data item as there are no previous reviews for that data [207]. When this technique is applied, random items are recommended to the users, and there emerges a high probability that this new item is randomly chosen to be a part of the to-be-recommended dataset. Hence, this technique effectively solves the problem of the cold-start problem for items.

4.2.2.3 User-Based Collaborative Filtering

User-based collaborative technique is a neighbourhood-based approach in the world of recommender systems [208]. There exist numerous factors which are taken into account to construct an efficient recommender system. In the previous subsections we read about the recommendations taking place based on the popularity of the items and based on the randomness of the recommender engine. But to create an algorithmic approach, the recommendations in collaborative filtering depends on the neighbour of the user. In the user-based collaborative filtering, if user's neighbour likes an item X, the recommender engine recommends Y item to the user, given that X and Y are related to each other in some way [209]. This type of generation of recommendations are specific to the users. However, the next subsection describes such recommendations generating based on the items. The similarity between two users a and b can be given by the following equation 4.1:

$$sim(a,b) = \frac{\sum_{i \in I}(r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\sum_{i \in I}(r_{a,i} - \bar{r}_a)^2}\ \sqrt{\sum_{i \in I}(r_{b,i} - \bar{r}_b)^2}} \quad (4.1)$$

here,

$r_{a,i}$ is the rating of user a for item i and similarly $r_{b,i}$ is the rating of user b for item i.

I is the list of all the items rated by the users a and b.

4.2.2.4 Item-Based Collaborative Filtering

As given in the previous subsection, item-based collaborative filtering is also a type of neighbourhood based approach as it is a collaborative filtering based recommendation technique but unlike in the user-based approach, the item-based approach provides recommendations to the user based on the similarity between the items. To give an example, if the user likes a product P

and if product P is similar to product Q in one or the other way, there is a high probability of the user liking product Q. Hence, one the recommender engine knows that the user likes product P, it generates such product recommendations such that certain feature vectors of those items are common to the product P [210]. Equation 4.2 determines which items are to be recommended to the user from a list of items.

$$score(u,i) = \sum_{j}^{I} \frac{similarity(i,j)(r_{(u,j)} - \bar{r}_j)}{similarity(i,j)} + \bar{r}_i \tag{4.2}$$

here,

u is the user for whom the recommendations are generated,

i represents the item concerned, and

j represents the list of items similar to item *i*

4.2.2.5 Singular Value Decomposition (SVD)

This technique aims at reducing the dimensionality of the dataset. The features in the datasets that are used for computation are enormous in number, and they are often not restricted to just one dimension. This technique helps in that scenario and hence extract such features which are used in comparing the users [211]. To effectively perform the recommendation, in the first step, the dataset is spread across a user-item matrix, and the entries in the cells are the individual ratings. In the second step, the empty cells are filled with average rating values. In the last step, the dimensionality of the user-item matrix is decreased by using the SVD algorithm, and the rating values can be predicted from the previous matrix. Let each item in the user-item matrix be represented by vector i and each user be represented as vector u. The expected rating can be the following Equation 4.3:

$$expected\ rating, \hat{r}_{ui} = i^T u \tag{4.3}$$

here i^T is the transpose of the item vector *i*.

to find the vectors *i* and *u*, the minimum of the user-item matrix can be found using the following Equation 4.4:

$$minimum(u,i) \sum_{(u,i)\in K} (r_{ui} - i^T.u)^2 \tag{4.4}$$

To avoid our model to over-fit the training set, we have introduced a regularization vector, λ to the following equation 4.5:

$$minimum(u,i) \sum_{(u,i)\in K} \left(r_{u,i} - i^T.u\right)^2 + \lambda(\left|\left|i\right|\right|^2 + \left|\left|u\right|\right|^2) \quad (4.5)$$

These classifiers are used as hyperparameters of the ensemble learning engine.

4.2.3 Ensemble Techniques

Recommender systems are primarily of three types, i.e., collaborative filtering based, content-based, and hybrid. Collaborative filtering technique involves the neighbors of the user, and depending upon the past purchase history of neighbor and his likes or dislikes, the user is recommended items [187]. In content-based filtering, the recommendations take place based on the content of items user previously purchased or liked [188]. However, the hybrid recommender system is an amalgamation of both content based and collaborative filtering-based recommender systems. It works on the properties of both content-based and collaborative filtering recommender systems.

Ensemble Learning is a technique where many base models are combined to form one optimized model. It is particularly useful for either an extensive dataset or a very small dataset. In the former case, the dataset is divided into smaller subsets of data, and different classifiers work on these different individual datasets. In the latter case, the concept of bootstrapping is used [212]. In this technique, one subset of data is selected at random, and then another subset of a dataset is selected with replacement from the entire pool of data points. Ensemble learning is the most efficient way of creating a hybrid recommender system. Hierarchical learning is a two-step process. In the first step, each base learner learns or processes the information either separately or after communicating. In the second step, the predictions summarized by the trained classifiers are combined, considering the functional hierarchy. The systematic structure of parametric optimization of the Hierarchical Ensemble using PSO is provided in Figure 4.2.

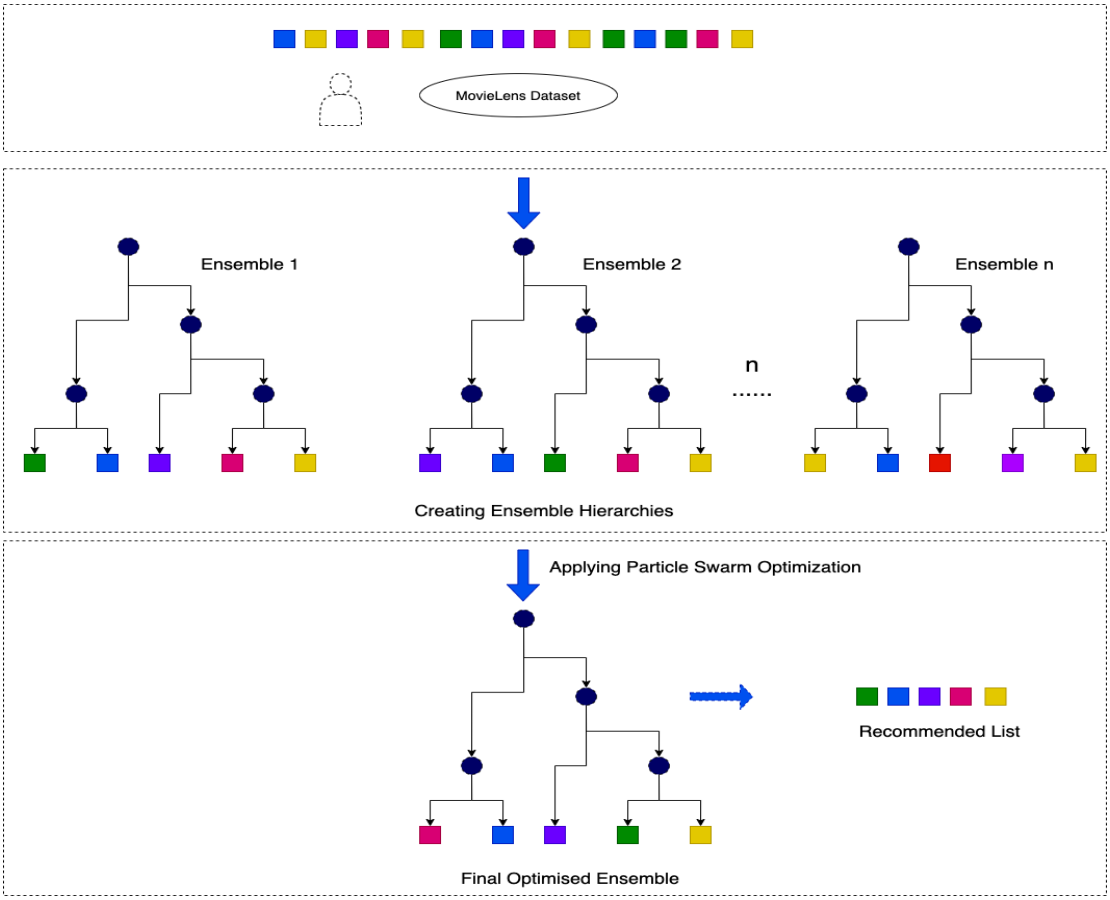


Figure 4.2 Systematic structure of the parametric optimization of Hierarchical Ensemble using PSO

There are two types of ensemble systems according to their functionality, parallel ensemble system, and sequential ensemble system. Figure 4.3 represents the architecture of the parallel ensemble systems. In parallel systems, the execution of all the recommenders involved in the system is independent of each other, and the outputs of each recommender are combined using an aggregator. Weighted and switching are two types of parallel ensemble techniques. Figure 4.4 represents the architecture of the sequential recommender systems. In these systems, the input of (n-1) recommenders depend upon the output of the previous recommender, given that there are n numbers of recommenders. Cascade and Feature Augmentation are two types of sequential ensemble techniques.

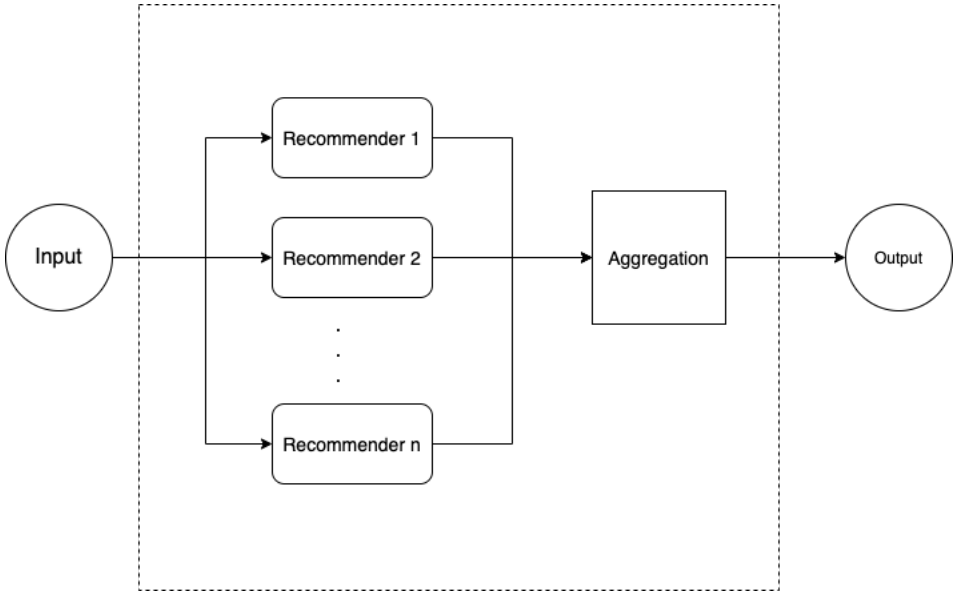


Figure 4.3 Architecture of the parallel ensemble systems

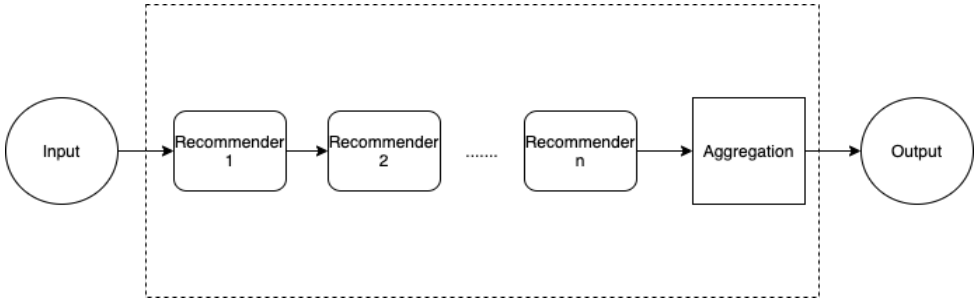


Figure 4.4 Architecture of the sequential ensemble systems

There are several types of ways to perform the ensemble learning by forming ensembles.

4.2.3.1 Bagging or Bootstrap Aggregating

It is a technique of ensemble learning wherein, bootstrapping and aggregation are combined to form one model. Given the data, several bootstrapped samples are formed, and a decision tree is formed on every bootstrapped subsample. Afterward, a decision tree is formed after aggregating all the decision trees to find the most optimum solution [213]. It is represented as following:

For every bootstrap sample (X_1^*, Y_1^*),….. (X_n^*, Y_n^*),
Compute the bootstrapped estimator using the following Equation 4.6:

$$E = h_n\big((X_1, Y_1), \ldots\ldots (X_n, Y_n)\big)(\cdot) \tag{4.6}$$

where function $h_n(\cdot)$ defines the function of the data.

4.2.3.2 Boosting

Unlike Bagging, the boosting ensemble methods are sequential, where the weights depend on the previous fitness functions. The base models considered for boosting have low variance and high bias because boosting aims at reducing bias. In addition to this, the boosting models are computationally expensive to fit [214]. The task of fitting the weak learners to form a final sequential model can be of such further types.

4.2.3.3 Adaptive Boosting or Adaboost

The Adaboost ensemble model [215] can be defined as a weighted sum of L weak learners by the following Equation 4.7:

$$s_L(.) = \sum_{l=1}^{L} c_l \times w_l(.) \tag{4.7}$$

Here, c_l represent the coefficients and w_l represents the weak learners.

Since this is an additive problem and requires both coefficients and weak learners, we employ iterative optimization technique. It can be given as following Equation 4.8:

$$s_l(.) = s_{l-1}(.) + c_l \times w_l(.) \tag{4.8}$$

This can further be denoted in Equation 4.9:

$$\begin{aligned}\big(c_l, w_l(.)\big) = \ & \arg min_{c,w(.)} E\big(s_{l-1}(.) + c \times w(.)\big) \\ & = \arg min_{c,w(.)} \sum_{n=1}^{N} e(y_n, s_{l-1}(x_n) + c \times w(x_n))\end{aligned} \tag{4.9}$$

here,

E(.) represent the fitness error of the model, and

E(.,.) represent the loss function

4.2.3.4 Gradient Boosting

In this method, the additive problem is visualized such that it is casted into gradient descent [216]. The gradient descent process can be written as the following Equation 4.10:

$$s_l(.) = s_{l-1}(.) - c_l \times \nabla_{s_{l-1}} E(s_{l-1})(.) \tag{4.10}$$

Here E(.) is the fitting error, and
cl is the coefficient of the step size.

4.2.3.5 Bayes' Optimal Classifier

It is a classification technique that makes the computations more feasible by presuming that the data is conditionally independent. Each hypothesis is given a vote equivalent to the possibility that the training data is sampled if that hypothesis were true [217]. It is given by the following Equation 4.11:

$$x = argmax \sum_{c_j \in C, h_i \in H} P(c_j | h_i) P(T|h_i) P(h_i) \tag{4.11}$$

here, x ∈ predicted class,
C ∈ set of all possible classes,
H is the hypothesis space, and
T ∈ training data

The ensembles are said to have two types of designs, i.e., parallel and sequential. For example, weighted and switching designs can be considered as parallel designs because, in these designs, all the recommenders can work on problems to either provide a weighted average or conquering tasks using different recommenders, respectively. However, cascade and feature augmentation designs are sequential since they operate sequentially, and the execution of one recommender system and generating its output is essential to carry forward the rest of the process.

4.2.4 Advantages of Ensemble Learning

There are primarily three reasons for performing ensemble learning. The first reason is Statistical. Due to the lack of appropriate data, it gets challenging to represent the data distribution efficiently. The second reason is Computational. Among many available models, it gets difficult to decide which model is to be chosen. The third reason is Representational. This reason occurs when the decided model is not able to provide adequate results because of providing improper decision boundaries. We have used hierarchical learning because it produces a high modularity system. The two-steps included in this technique increase the efficiency of the system.

4.3 Particle Swarm Optimization

In this study, we have emphasized the Model Generation segment of the process. Our technique, EnPSO, we use Particle Swarm Optimization (PSO) to search the model parameter space to find the best performing model. The model's generation is done by identifying and combining intelligent operations like intersection, concatenation, and many more. The system then generates the structure of the model. After the generation of the model, the system performs the optimization of parameters. In this stage, the system identifies the most optimum model by continually tweaking the parameters in a large search space [218].

Particle Swarm Optimization revolves around the concept of moving particles or solutions around in the search-space dimension depending upon the particle's position and velocity. Its main advantage is that it can search in vast spaces of particles. Evolutionary Algorithms are meta-heuristic algorithms that aid in solving hard problems. These algorithms mimic the working of nature in various forms. Such processes of nature are replicated to perform computations, and hence, they help in achieving the most optimum results [219]. There are several different bio-inspired evolutionary algorithms used to improve the computation results. Some of these algorithms are Genetic Algorithms [220], Ant-Colony Optimization [221], Artificial Bee Colony Algorithm [222], Bees Algorithm, Cuckoo Search [223], Firefly Algorithm [224], Particle Swarm Optimization [225], Hunting Search [226], etc.

4.3.1 Working of the Ensemble Model

There are following four steps to perform evolutionary computation, i. Initialization, this step deals with creating an initial population of solutions. The population is a collection of all the possible solutions to the problem [227]. ii. Selection, after the population is created, the members of the population i.e., all the possible solutions to the given problem, are evaluated using a Fitness Function. This function decides how efficient a solution is [228]. And hence, after using this fitness function, the most efficient members are selected. iii. Genetic Operation, for carrying out the evolutionary computation, genetic operations are required, which mimic the biological processes. This operation takes place in two steps. The first one is crossover [229]. In this part, all the selected top members are used to create the next generation. This is done so that the best features of both the parents are inherited by the children so that the results keep improving in every iteration. The second step is mutation [230]. Here, some changes or mutations are introduced in the children as produced in the previous step. This is done so that the children do not turn out to be exact replicas of their parents. If this happens, there will be no evolved generation.

PSO works on a population (swarm) of candidate solutions (particles). These particles move and interact in the search-space dimension adhering to a set of formulae. Another interesting concept in PSO is topology. A topology of the swarm can be defined as the subset of particles wherein each particle can trade information. This adds to the advantage of the algorithm as it promotes particle-particle communication [225]. Thus, the entire swarm agrees upon the same best position from a single particle. However, there is a downside to this concept that the swarm might get trapped in local minima. The working of the model is as follows. In the first step, the AutoML generates a random position for the particles within an initialization region. In the second step, the velocities of the particles are initialized, and this can further be done in three ways. i) within an initialized region. ii) zero. iii) assign random values to disallow particles from exiting the search-space in the first iteration [231]. The particles can also move to regions outside the feasible search space [232]. The activity of the swarm is characterized by two parameters, C_{soc} which determine the social attractiveness and C_{cog} which determines the cognitive attractiveness [233]. The velocity update of each particle i is given by Equation 4.12:

$$\boldsymbol{v}_i = w.\boldsymbol{v}_i + C_{soc}\,.\boldsymbol{r}_1 \circ (x_i - g) + C_{cog}\,.\boldsymbol{r}_2 \circ (x_i - b_i) \qquad (4.12)$$

where,

$\exists$ i $\in$(1,…..,n),

n is the swarm size,

r_1 and r_2 are the randomly generated vectors to stop the convergence to local minima,

b_i defines the best position established by the particle *i*,

g_i is the global best position established by the swarm for that particle *i*

After every iteration, the position of the particle gets updated depending upon its velocity and this is denoted by Equation 4.13:

$$x_i = x_i + v_i \tag{4.13}$$

Algorithm 4.1 Particle Swarm Optimization (PSO)

Steps:

1. **for** each particle p in a swarm population P:
2. **initialize** random x_p
3. **initialize** random v_p
4. **Evaluate** fitness function $f(x_p)$
5. $Local_p = x_p$
6. **initialize** $global_p$ as x_p having best fitness
7. **repeat until** stopping criteria is satisfied
8. **for** each particle p:
9. **update** v_i and x_i according to Equations 12 and 13
10. **Evaluate** fitness function $f(x_i^t)$
11. **If** $f(local_i) < f(x_i^t)$
12. $Local_i \leftarrow x_i^t$
13. **If** $f(global_i) < f(x_i^t)$
14. $global_i \leftarrow x_i^t$

Algorithm 4.1 states the steps for implementing the PSO algorithm.

Algorithm 4.2 Ensemble Particle Swarm Optimization (EnPSO)

Input: Users (u_1, u_2, u_3,…,u_n) who rated movies (m_1, m_2, m_3,…,m_4)

Output: Optimized ensemble generating recommendation results

Steps:

```
for R in rows(dataset) :
    if R.movies == NULL
        R.delete
    end
    if R.rating == NULL
        R.delete
    end
    if R.tags = NULL
        R.delete
    end
end for
ibcf = ibcf.train (dataset)
ubcf = ubcf. train (dataset)
random  = random  . train (dataset)
popular = popular. train (dataset)
svd = svd. train (dataset)
ensemble = ensemble.create(ibcf, ubcf, random, popular, svd, initial_params)
objective_function = ensemble.results.MSE
pso(ensemble, objective_function, parameter_range)
optimized _params = pso.optimize()
return ensemble(ibcf, ubcf, random, popular, svd, optimized_params)
```

In Algorithm 4.2, our system creates a hierarchical ensemble of recommendation techniques. Any two or more recommendation models can be combined using voting or mixing, which leads to many different types of hierarchical ensembles that can be formed. Each structure will have its own set of hyper-parameters, which will need to be optimized. Different structures with their hyper-parameters will form a search space where the system finds the best performing model. To

perform the search, we have used Particle Swarm Optimization. We performed the implementation of the EnPSO algorithm in R using the recommenderlab R package [234].

4.3.2 Advantages of Using PSO

In PSO, every particle has a memory that is vital to the working of the algorithm. Every particle has an independent velocity of its own and using these velocities, such particles keep updating themselves dynamically. The particles help in transmitting the information among each other and that is how the information processing takes place. Not every particle holds the power to transmit information. Only the best lot of particles are allowed to do so. The spread of information is a one way transmission and the best solution leads to the evolution. Its main advantage is that it can search in vast spaces of particles. In comparison with the other evolutionary algorithms, the key advantages of using PSO are its uncomplicated mechanism of implementation and the hyperparameters which need to be adjusted are not enormous.

4.4 Results and Analysis

4.4.1 Dataset

We have used the benchmark MovieLens dataset for the implementation. This dataset consists of 3 data files encoded in UTF-8 format <ratings, tags, movies>. Figure 4.5(a) provides an analysis of the dataset based on the count of different ratings. Figure 4.5(b) provides the average rating of the users. Figure 4.5(c) shows the average rating of the items. In Figure 4.5(d), the number of related items grouped by users is shown. Figure 4.5(e) provides the analysis of the dataset based on the number of scores items have [235]. We have done the implementation on 3 versions of MovieLens dataset:

1. 100K: It consists of 100,000 ratings from 943 users on 1682 movies.
2. 1M: consists of 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000.
3. 10M: consists of 10000054 ratings and 95580 tags applied to 10681 movies by 71567 users

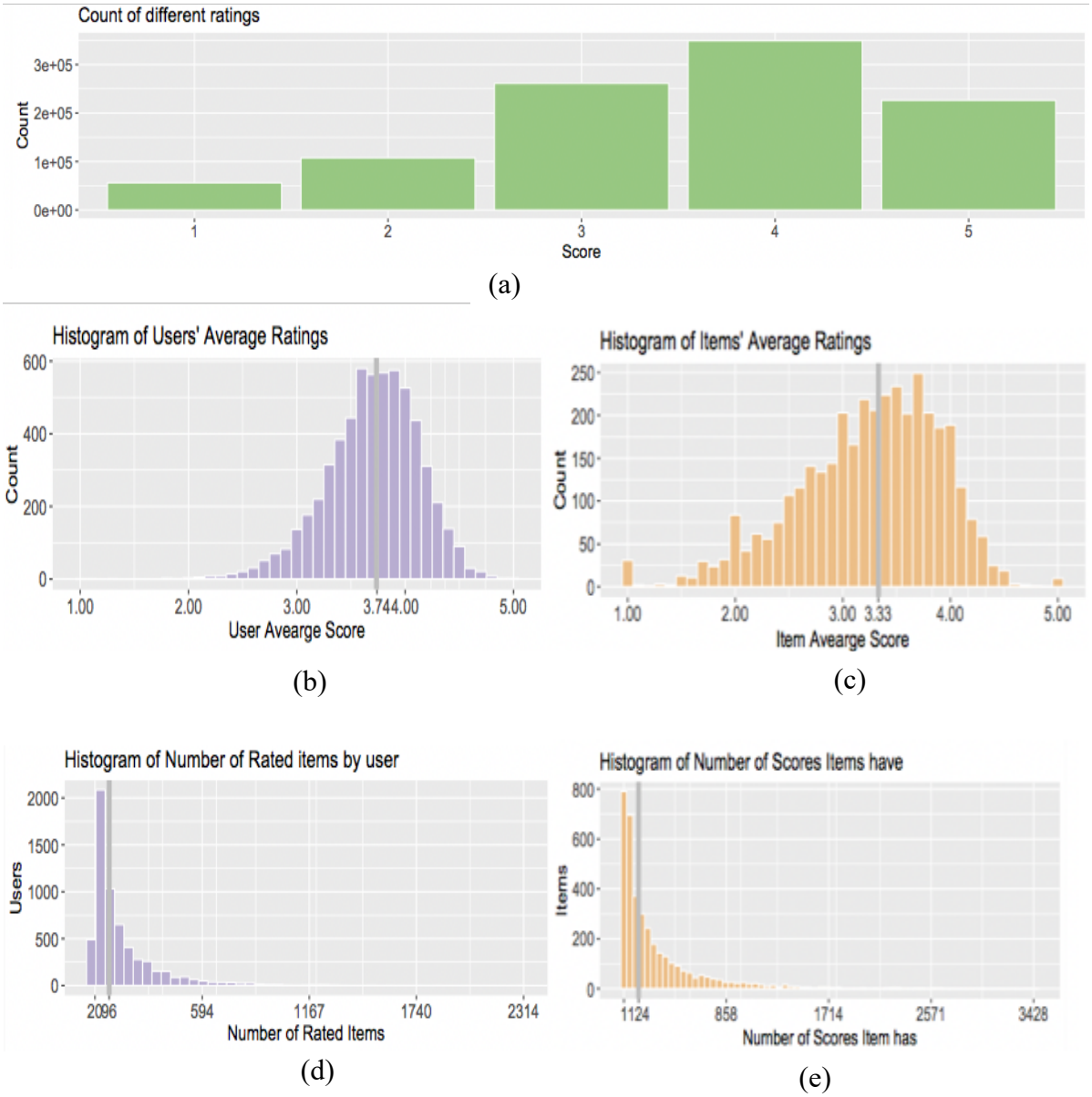


Figure 4.5 (a) Study of the dataset based on number of different ratings (b) Study of the dataset based on average ratings of users (c) Study of the dataset based on the average ratings of the item (d) Study of the dataset based on the number of related items

4.4.2 Metrics

To analyse the results obtained by the system, we have used the 3 error metrics, i.e., RMSE, MSE and MAE.

4.4.2.1 Root Mean Square Error (RMSE)

It is defined as the standard deviation of the prediction errors. It is calculated by the following Equation 4.14:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}((\hat{x_i}) - (x_i))^2}{n}} \tag{4.14}$$

For Equations 4.14, 4.15 and 4.16,

Here, $\hat{x_i}$ is the predicted value

and, x_i is the observed value for the i^{th} observation

4.4.2.2 Mean Square Error (MSE)

MSE is used to detect the quality of an estimator by calculating the mean of its squared error. It is calculated by Equation 4.15:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - x_i)^2 \tag{4.15}$$

4.4.2.3 Mean Absolute Error (MAE)

MAE is the average of all the absolute errors. It is given by Equation 4.16:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x| \tag{4.16}$$

4.4.3 Results and Analysis

The final results after execution have been shown in this section. Table 4.1 depicts the comparative analysis of error metrics between the other base recommendation techniques and EnPSO on the 100K MovieLens dataset. Table 4.2 depicts error metrics on the 1M MovieLens dataset, and similarly, Table 4.3 depicts error metrics on the 10M MovieLens dataset. The results of the 1M MovieLens dataset have been shown graphically using bar charts in Figure 4.6(a). Similarly, the results of the 10M MovieLens dataset have been shown in Figure 4.6(b), and the results of the 100K MovieLens dataset have been shown in Figure 4.6(c).

Table 4.1 Error metrics for 100K MovieLens dataset

	UBCF	IBCF	SVD	RANDOM	POPULAR	EnPSO
RMSE	1.0906673	1.266997	1.1077915	1.591971	1.0347649	**0.9183021**
MSE	1.1895551	1.555479	1.2272020	2.425979	1.0707384	**0.9837294**
MAE	0.8905862	1.130137	0.9066842	1.174222	0.8278716	**0.7529134**

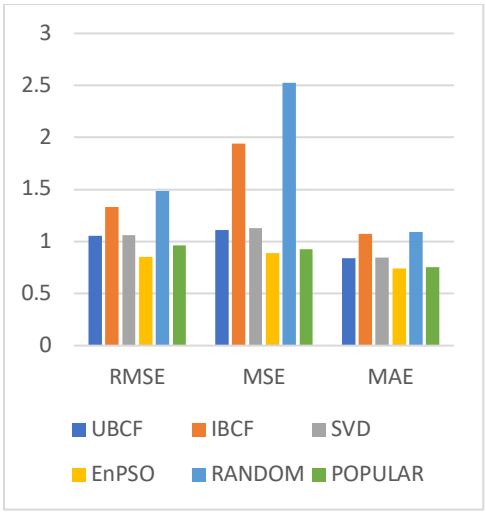
Table 4.2 Error metrics for 1M MovieLens dataset

	UBCF	IBCF	SVD	RANDOM	POPULAR	EnPSO
RMSE	1.0906673	1.366997	1.0752522	1.510546	0.9732996	**0.8837047**
MSE	1.1895551	1.98438	1.1561673	3.689639	0.9473122	**0.9183024**
MAE	0.8905862	1.196394	0.8598624	1.413785	0.7677965	**0.7192942**

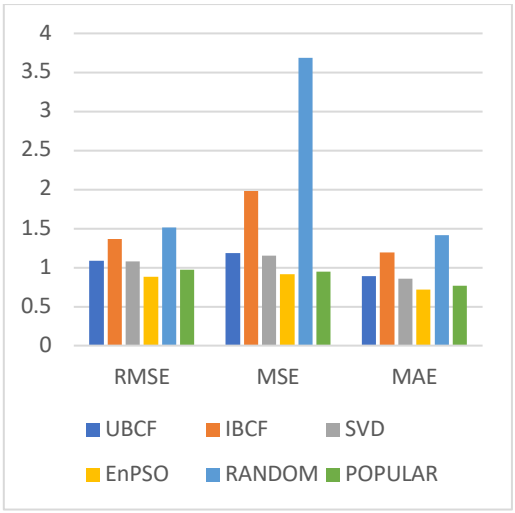
Table 4.3 Error metrics for 10M MovieLens dataset

	UBCF	IBCF	SVD	RANDOM	POPULAR	EnPSO
RMSE	1.0543233	1.329419	1.0636744	1.488290	0.9638857	**0.8528932**
MSE	1.1115976	1.943237	1.1314032	2.527348	0.9290757	**0.8920123**
MAE	0.8388887	1.075388	0.8483209	1.0922772	0.7560278	**0.7421234**

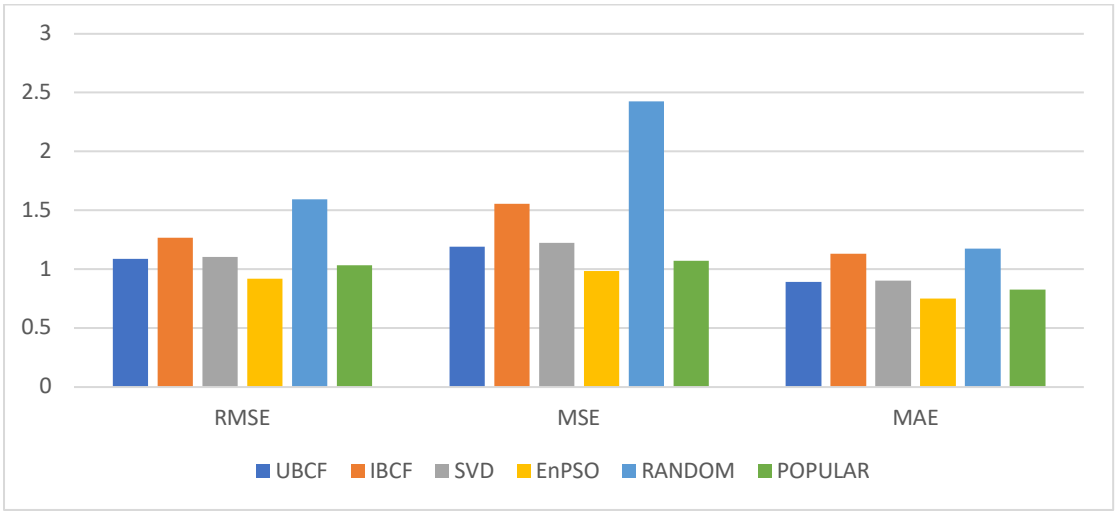
It can be seen in Table 4.1, Table 4.2, and Table 4.3 that the MSE is greater than RMSE and MAE. Furthermore, for all the recommendation techniques, our proposed algorithm EnPSO has the least error metrics. The highest error values are of Random and IBCF recommendation techniques.



(a)



(b)



(c)

Figure 4.6 (a) Error analysis of 10M dataset (b) Error analysis of 1M dataset (c) Error analysis of 100K dataset

It is evident from the results that our proposed algorithm makes the minimum error compared to the rest of the machine learning techniques. In this study, we have calculated the error metrics, i.e., RMSE, MSE, and MAE, for 100K, 1M, and 100M datasets. However, for other metrics, we have analyzed the results on 1M and 10M datasets. It can be seen from the results that our model proved to be better than all the other techniques. The error metrics of our technique were lower than the error metrics of every other technique. We used three error metrics, namely, Root Mean Square Error (RMSE), Mean Square Error (MSE), and Mean Absolute Error (MAE). For the 100K MovieLens dataset, the RMSE value of the proposed algorithm was 0.91, the value of MSE was

0.98, and the value of MAE was 0.75. These error values were the least among all the other techniques. In the 1M MovieLens dataset, the value of RMSE was 0.88, the value of MSE was 0.91, and the value of MAE was 0.71. These error values were also lesser than the error values of other techniques. For the 10M MovieLens dataset, the RMSE value of our proposed algorithm was 0.85, the error value of MSE was 0.89, and the error value of MAE was 0.74. These error values of our proposed model are also lesser than the error values of all other techniques. Since the other techniques yielded higher error while computing the results, we can safely say that our proposed model provides us with the most optimum results. In this study, we addressed the computational challenge of model selection for achieving the best results. Hence, we used AutoML to enable automated model selection.

4.5 Chapter Summary

An AutoML technique was proposed to select an optimized model of ensemble learning automatically. The system used an evolutionary technique to move towards the best performing model iteratively. The results have shown that our technique can effectively combine the base technique automatically to create an ensemble model that can outperform the base models. In this way, the model can improve performance and is also free from the biases of the base techniques. Five base recommendation techniques were used; namely, Recommendations Based on Popular Items, Recommendations Based on Random Items, Item-Based Collaborative Filtering (IBCF), User-Based Collaborative Filtering (UBCF) and Singular Value Decomposition (SVD) for the experiments. The experiments were performed on the MovieLens data, a standard benchmark dataset in the field of recommender systems which generated unparalleled results.

Chapter 5

USING AUTOML AND DEEP LEARNING FOR GENERATING RECOMMENDATIONS

The massive number of permutations in which recommendation models are combined to create an ensemble model adds another layer of complexity to an already complex problem. Thus, in line with the recent trend in Automated Machine Learning (AutoML) aimed at reducing the complexity associated with model selection, there is a need for a machine learning framework that can learn the best ensemble model for a given problem given the base models. Creating ensembles that effectively combine individual techniques is a complex problem in its rights. The number of permutations in which techniques can be combined is too large for any Machine Learning practitioner to explore all on his own. Thus, he has to rely on the knowledge of ensemble learning to make informed decisions and come up with an ensemble model that works. This becomes a big chore in itself in the development process of a recommender system. This research work aims to eliminate this process by introducing an Automated Machine Learning (AutoML) framework that provides an optimized ensemble structure to the user without any oversight or the requirement of advanced knowledge about creating ensembles.

Section 5.1 consists of the overview of the work done in this chapter. Section 5.2 comprises of generating ensembles using deep learning techniques. Section 5.3 discusses the hierarchical supervised model and Section 5.4 describes the AutoML Generation of Optimal Ensemble Model. The results and their analysis is discussed in Section 5.5, followed by chapter summary in Section 5.6.

5.1. Overview

AutoML was first introduced [186] to automate the training of a Recurrent Neural Network by Reinforcement Learning to identify the most optimum architecture. AutoML was defined as a combination of automation and Machine Learning [236]. The authors in [237] explained how, without much intervention from experts, the ML applications could be implemented by the domain experts using AutoML. As explained in [238], the implementation of AutoML encompasses all

the machine learning processes, i.e., data preparation, feature engineering, model generation, and model evaluation. Employing a single recommendation technique is already a complex problem that requires optimizing the model parameters and the hypermeters for the given data. On top of that, combining multiple such techniques using either ensemble learning or some other method adds another layer of complexity and requires the practitioner to have advanced knowledge of ensemble learning. To solve this issue, we have applied AutoML, which automates this entire process and reduces such unnecessary layers of complex computations to generate accurate recommendations. Another alternative way to reduce the complexity of the model selection is to use standard frameworks for creating ensembles. However, such a system would not be optimized and hence would not provide the best results. Attempt to optimize the ensemble will require expert knowledge and will hence introduce complexity. Our system provides an easy way to find a well-performing ensemble without the associated complexities.

AutoML was first incepted by [186] with the aim to automatize the method of training a Recurrent Neural Network (RNN) with the help of Reinforcement Learning to recognize the best performing architectonics. Authors in [236] described AutoML as an amalgamation of automation and Machine Learning (ML). The study [237] described how efficiently the ML techniques could be implemented without the requirement of mediation by domain experts. Authors [238] explained that the implementation of AutoML includes every machine learning procedure, i.e., data preparation, feature engineering, model generation, and model evaluation. This work proposes an AutoML (Automated Machine Learning) framework wherein the algorithm will optimize and pick out a well-performing ensemble from the search space by giving the base recommender. The framework En-DLR (Ensemble Deep Learning Recommender) can be extended to any domain, any number of base algorithms, and customized with various ensemble techniques in the framework. The framework optimizes a hierarchical ensemble and returns a well-performing ensemble from the search space. Genetic algorithm is used for our optimization flow. With this technique, not only a well-performing recommender system can be created but also adding a new recommendation technique to the framework is algorithmically trivial. Recommender system is a growing field, and much new development is taking place. Industries that are heavily dependent on recommender systems can easily plug in a new recommender into the framework and get a better-performing system with no effort. The work aims to reduce the complexity of creating

ensembles for recommender systems. Classifier ensembles, in the past, have given promising results in the field of pattern recognition structures. They have been applied in several studies [239], [240]. Although a plethora of studies have been done on this topic, determining a set of parameters that results in a maximized classification accuracy of the ensemble is unknown. An ensemble consists of several parameters like classifier type, parameters of the classifier, size of classifier, combination method of the classifiers, and feature selection. Since the search space of such expansive parameters is large, finding the most optimal values for the parameters is challenging.

Several studies have been performed on finding the most optimal machine learning techniques for classification problems. [241]–[246]. There has been considerable work done in meta-learning problems [247], Auto-ML and optimization problems [248]. To generate efficient recommendations, tremendous amount of research work has been performed in the field of meta-learning. Although meta-learning was first introduced as a proof of concept in the paper [249], authors in [250] carried out a study to determine the rules of meta-features to gauge the viability of the algorithm for the problem. Authors in [251] further extended this work implementing several other features and learnings based on Decision trees. The authors [20,p4] analyzed the connection between the problem complexity parameters to distinguish the dataset and the results of classification algorithms. Authors in [252], [253] proposed techniques to recommend algorithms and parameter settings leading to performance variation of said algorithms on different datasets. Authors [253] implemented these works to develop a novel rule-based classifier selection method. In [254], the authors created a novel meta-learning algorithm to deduce the most optimal feature selection algorithms for the given dataset. In another study [255], the authors implemented five unique classifications of meta-features, namely, simple, statistical, information-theoretic, model-based, and landmarking. They categorized the datasets and developed a novel regression model to join different datasets to every candidate algorithm. Authors [256] developed a novel method to engineer meta-features. In another study, the authors [246] created a technique pertaining to a ranking list that helped deduce which aggregation algorithm is best suited for that ranking list. In [254], the authors created a novel meta-learning algorithm to deduce the most optimal feature selection algorithms for the given dataset. In another study [255], the authors implemented five unique classifications of meta-features, namely, simple, statistical, information-theoretic, model-

based, and landmarking. They categorized the datasets and developed a novel regression model to join different datasets to every candidate algorithm. Authors [256] developed a novel method to engineer meta-features. In another study, the authors [257] created a technique pertaining to a ranking list that helped deduce which aggregation algorithm is best suited for that ranking list. There is ample literature on recommendation techniques for any domain. Different techniques optimize for different parameters such as privacy and trust. There is considerable literature on creating ensembles from these techniques to get the best performance by combining the model in unique ways. However, the literature lacks its algorithm, making creating ensembles effortless for a practitioner who is not an expert on the subject.

5.2. Generating Ensembles Using Deep Learning Techniques

Deep learning is a discipline of machine learning algorithms that are layered. Here each layer does nonlinear processing producing different abstractions of the data. Each layer takes the output of previous layers as input, hence producing hierarchical abstractions. For example, in computer vision, the original image matrix is the data that is processed. The first layer takes this data, and it performs feature extraction and transformation in such a way that it identifies the edges. It then gives this extracted feature as an input to the next layer, which may create a new layer of abstraction by identifying the orientation of the edges detected by the previous layer. This is repeated numerous times with multiple layers. As depicted in Figure 5.1, research work done using several deep learning techniques has been studied.

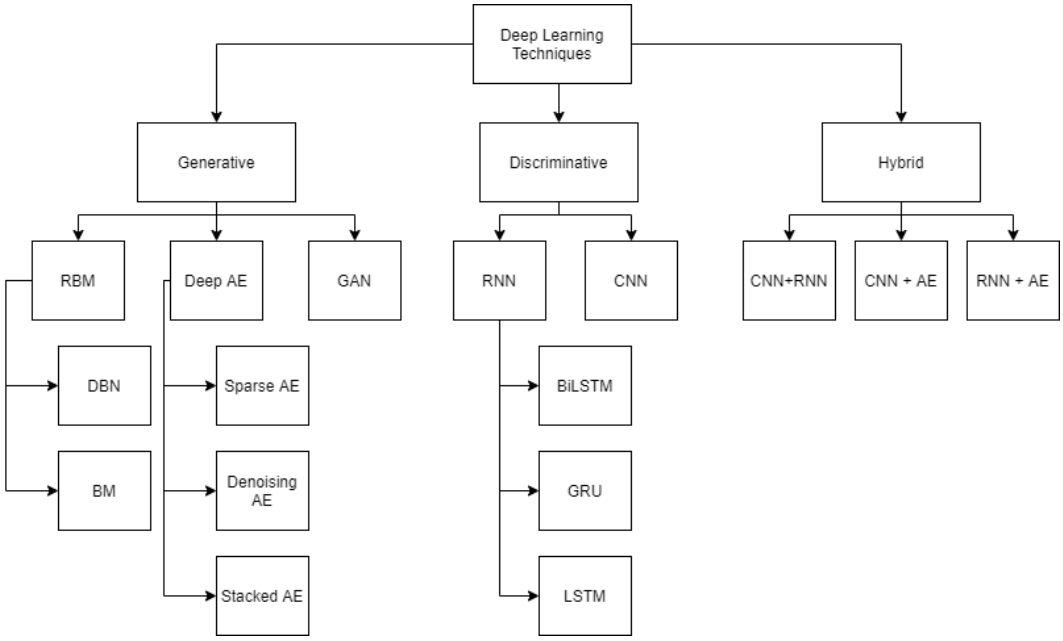


Figure 5.1 Deep Learning Techniques

The web is a vast pool of data; the dimensionality and modality of data present online are very high. To work with such multi-modal and complex data with extensive features requires extensive Machine Learning support. If we use traditional design and technology, the recommendations generated will not be of any use because the system will not be able to exploit the information at hand, and the complex pattern analysis of data would not be done. To solve this problem, we have used deep learning, which can work on highly complex data with hidden features and a high number of training instances. Deep learning's capability provides breakthrough results by extracting the data on a granular level and analyzing the patterns. Hence, the representation of all the details of data in a joint unified framework is made possible by using deep learning [258]. Conventional machine learning models like Matrix Factorization, etc., are linear models that can work efficiently on linear data, but when we have to deal with non-linear data, the computations require complex functions like sigmoid, tanh, etc. To catch such intricate user-item interaction patterns, there arises a need for deep learning models. The web is full of illustrative data; even for making recommendations, there is a lot of description and content attached to data. To understand this information and use it to our advantage, we have employed recommender systems. Another significant advantage of using deep learning is simplifying the process of feature engineering.

Every set of data has sophisticated features attached to it, and the task of deep learning is to understand the raw features and process them using supervised or unsupervised machine learning algorithms. Similarly, even for the content to every set of data, deep learning makes it possible for the system to use all the available content of data to generate expert recommendations. Another important reason for using deep learning is sequential pattern mining. The task of sequence modeling that includes Naturel Language Processing, speech recognition, etc. are usually performed by either Recurrent Neural Network (RNN), which has internal memory to learn the next in sequence or Convolutional Neural Network (CNN) that performs the sequence modeling using temporal computations.

The execution of the algorithm initiates after executing data cleaning on the dataset to extract the columns that are crucial for the working of the system. This is called feature extraction, wherein the model filters out the entire attribute set from the dataset that are appropriate to the model. In our study, the most crucial feature is movies. This attribute entails the movies enlisted in the dataset. After filtering out the movies, the system identifies NULL rows. All such the rows with NULL values are rejected. To conclude, the extracts the feature rating. Correspondingly, the system verifies the rows with NULL data points and discards them. Afterwards, the model filters out the tags attribute and carries out the process of feature extraction on such column values. Once the data is cleaned and features are extracted, our system Ensemble Deep Learning Recommender (En-DLR) applies all the recommendation classifiers. Subsequently, the ensembles are generated, and this process is entrusted with the hierarchical ensemble technique. Afterward, the system optimizes the ensemble with genetic programming. The authors [259] used genetic programming to develop the structure of the hierarchical ensemble and parameter optimization to tweak the parameters of algorithms pertaining to specific data sets. Their system allocated exponentially more time for the evolution of above-average templates, mimicking Hyperband approach [260]. In the final phase of the implementation, structures that are carried from the previous generation have maximum time to showcase their capabilities.

5.2.1. Ensemble Recommendation Method (ERM)

This study elucidates two techniques for Ensemble Recommendation Method (ERM) [246]. The two approaches for ERM are: 1) ERM-ML: Ensemble Recommendation Method - Using Meta-

learning, and 2) ERM-3ML: Ensemble Recommendation Method - Using three steps Meta-learning. It is inevitable that distinct ensemble configurations have different results metrics, and the implementation of meta-learning provides an unparalleled way to aid in shortlisting the best performing ensemble attributes for any given problem. Owing to the importance of defining the ensemble structure, the implementation of this study is primarily dependent on the generation of two models to recommend this structure using meta-learning techniques. The two phases are:

5.2.1.1 Training

This phase primarily creates the Meta-base. As the evaluation of datasets increases, the expected metrics related to performance of the recommender system also increases. The Meta-base hence generated possesses the meta-features of every dataset used in the system and predicting the most optimal structure for each data instance or the related meta-features.

5.2.1.2 Generalization

This phase typically carries out three crucial tasks, i.e., creating, training, and applying the meta-learners in real-world scenarios. This leads to the generation of the best performing ensemble created by the system. In this phase, the meta-features are filtered in from the newly created data instances and implements a classification model to generate recommendations for the best performing ensemble. Once the metabase is created, the original data instances are not needed because the entire computation is performed on the Meta-base. This leads in a massive reduction of computational effort of the system. In the phase, the meta-learner will recommends the best performing structure (number of classifiers, type of classifier, and model aggregation) for the proposed ensemble.

5.2.1.3 ERM-ML (Ensemble Recommendation Method - Using Meta-learning)

This was initially proposed in [261]. To implement this method, several datasets or instances of datasets are trained using distinct sizes of classifiers and aggregation techniques.

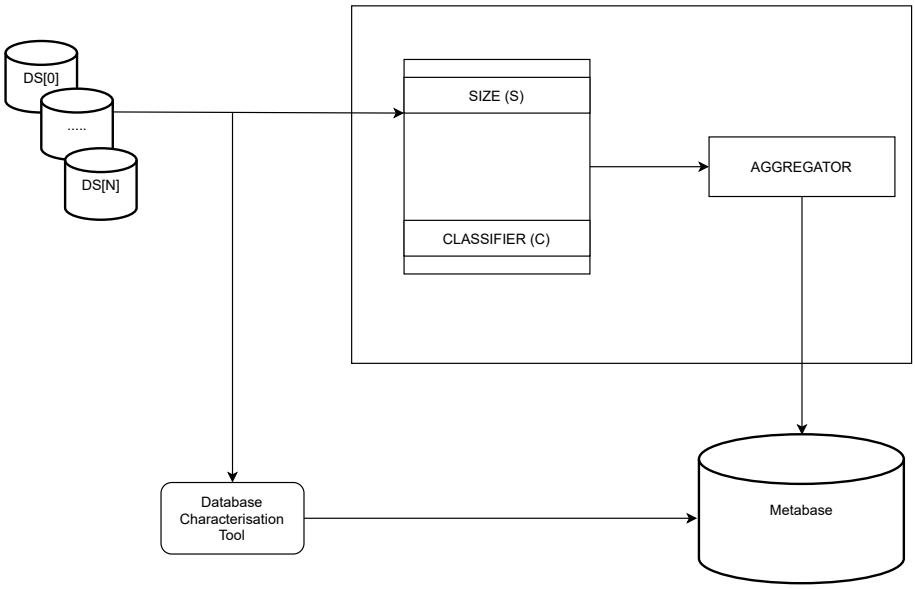


Figure 5.2 Architecture of ERM-ML

For any given dataset, the best ensemble is selected based on the result of implementing this method. The purpose of the Dataset Characterization Tool (DTC), as shown in Figure 5.2, is to extract the desired characteristics of the datasets, which are crucial for building the metabase. Hence, at the end of this step, the best ensemble size *S[best]*, best classifier *C[best]*, and best aggregation function *A[best]* are stored in the metabase file.

5.2.1.4 ERM-3ML (Ensemble Recommendation Method - Using 3 steps Meta-learning)

This is another technique for creating ensembles. It first calculates the best ensemble size S[best], and this is given as input into the next step, and the best classifier is selected *C[best]*. This classifier is further given as input into the next step where the best aggregator is selected, *A[best]*. Hence, we can conclude that ERM-ML is a parallel implementation technique, whereas ERM-3ML is a sequential implementation technique. It can also be concluded that the computation time of ERM-3ML is more than the computation time of ERM-ML. It can also be noted that ERM-ML generates only one metabase with the recommendation of all three attributes. In contrast, ERM-3ML generates three metabases in an incremental way, one for each parameter [262]. The architecture of this method is shown in Figure 5.3.

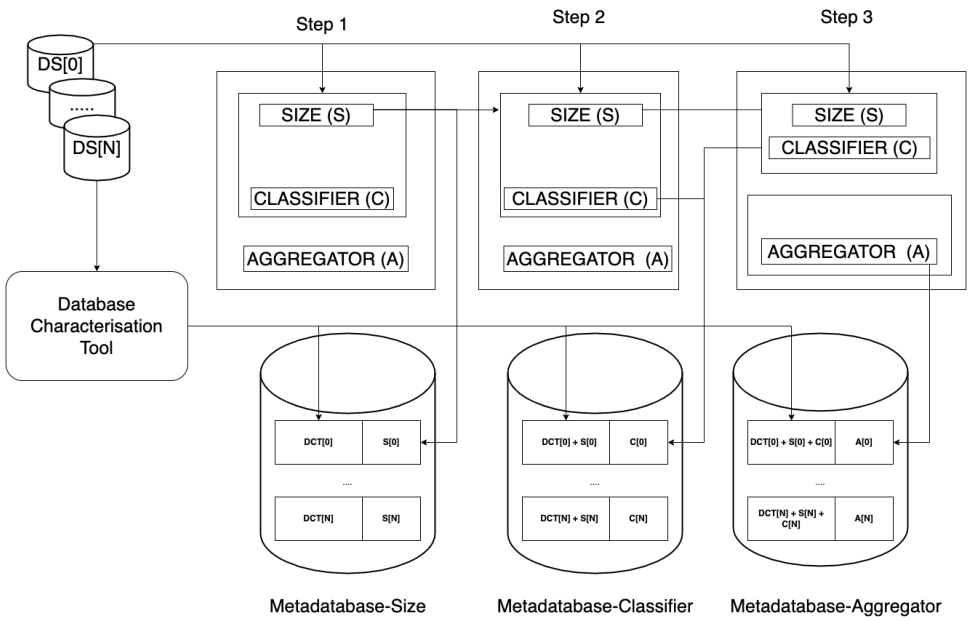


Figure 5.3 Architecture of ERM-3ML

We used ERM-ML in this study as this is a parallel approach to performing ensemble recommendations. In comparison, ERM-3ML is a sequential approach resulting in more space and time complexity. Hence, owing to this fact, the ERM-ML approach is the most optimal approach of the two implemented in our study.

5.2.2 Base Recommenders

Five base recommenders or ensemble classifiers are used in this study. These recommenders are chosen to amalgamate standard ML techniques and a deep learning technique. Using this combination, we can provide these base models to the system, and they hence, work in tandem to provide relevant results. Due to the vast efficiency of the techniques based on several parameters, the results are sparsely varied. Hence, we use these ensemble classifiers in an ensemble learning network or meta-learning network, wherein we combine the results of the classifiers by using an ensemble technique and present the final result, which is the most optimum compared to the results obtained by the classifiers individually. The ensemble classifiers used in the study are as follows:

5.2.2.1 Convolutional Matrix Factorization

The Convolutional Neural Networks (CNNs), which exist today, do not provide the expected results on recommender systems due to different objectives [57]. CNNs typically solve the

classification problems to estimate text labels like words, phrases, or sentences. Whereas, while implementing recommender systems, the primary task is to solve regression problems so that the ratings of users and items are accurately produced. Hence, the CNNs are not used for implementing recommender systems. We used a context-aware recommender model to rectify the aforesaid problem, i.e., Convolutional Matrix Factorization or ConvMF. This model stores the contextual information of textual documents citing description of items by using Convolutional Neural Network (CNN) and hence helps in improving the rating prediction accuracy. ConvMF effortlessly combines CNN into Probabilistic Matrix Factorization (PMF), which has widespread usage in recommender systems. Hence, the integration of PMF and CNN leads to practical usage of the combination of contextual and collaborative information. In this way, ConvMF accurately predicts ratings given a sparse dataset. Figure 5.4 represents the architecture of ConvMF.

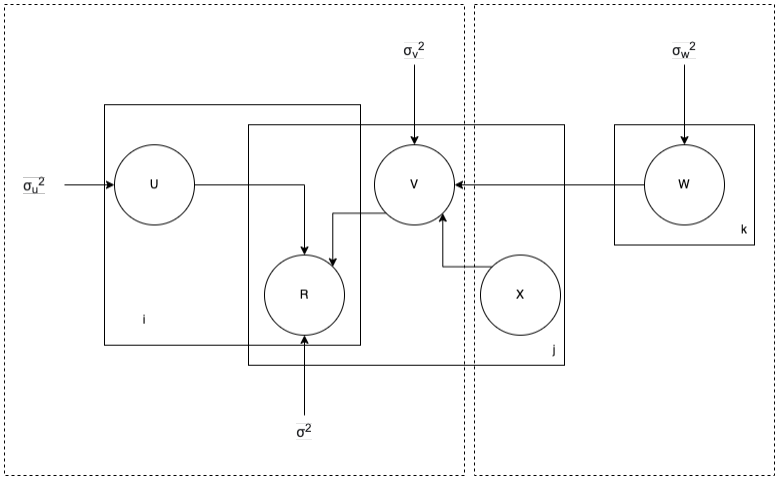


Figure 5.4 Architecture of ConvMF; left box represents PMF and right box represents CNN

The ideation behind this algorithm is to create document latent vectors from textual data of items which further use epsilon variables to generate item latent models. The CNN structure for ConvMF, as shown in Figure 5.5, is composed of four significant layers, i.e., embedding layer, convolution layer, pooling layer, and output layer.

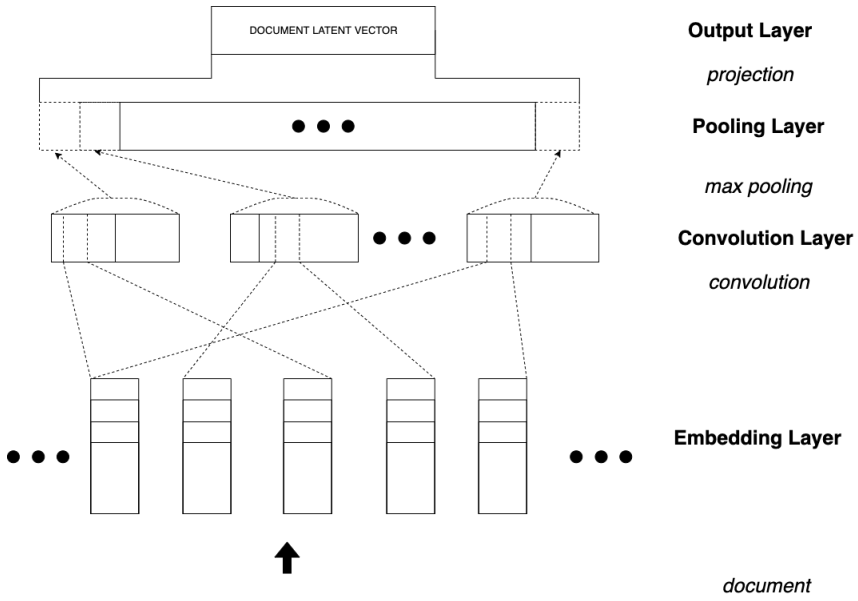


Figure 5.5 CNN architecture for ConvMF

5.2.2.2 Deep Collaborative Filtering

Deep learning techniques are highly efficient in determining the high-level representations from the input in various learning mechanisms. Such learned parameters constitute high-level knowledge. While using collaborative filtering, the issue of determining high-level features, user-item ratings, and related features from raw data is challenging [263]. Although the merged implementations of Matrix Factorization and Collaborative Filtering techniques successfully determine the dependencies between users and items, they cannot eliminate the cold start and data sparsity issues. Hence, to improve the collaborative filtering method, deep learning models need to be introduced. Authors in [264] proposed a novel deep learning method called Deep Collaborative Filtering (DCF) to amalgamate matrix factorization and deep feature learning. This technique implements the mapping between latent layers of the deep model and latent factors used in collaborative filtering. It can be seen the architecture of the model in Figure 5.6. DCF is a hybrid model that integrates user-item rating matrix and side information so that the matrix factorization and feature learning are connected.

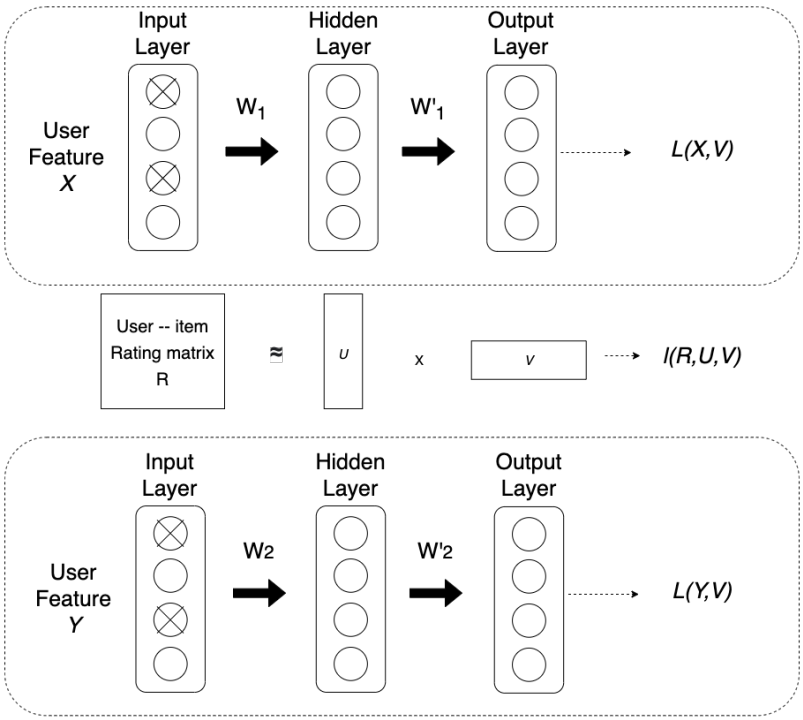


Figure 5.6 Architecture of Deep Collaborative Filtering (DCF) model

Mathematically, the model can be represented by the following Equation 5.1:

$$\arg\frac{min}{U,V}\ l(R,U,V) + \beta(\|U\|_F^2 + \|V\|_F^2) + \gamma\mathcal{L}(X,U) + \delta\mathcal{L}(Y,V) \qquad (5.1)$$

Here,

R represents the user-item matrix,

X represents the side information of the user,

Y represents the side information of the item,

β, γ, and δ represent the trade-off attributes,

l(R,U,V) function decomposes the rating matrix R into two latent matrices,

L(X,U) function joins the user contextual attributes to the latent factors,

L(Y,V) function joins the item contextual attributes to the latent factors

The function *l(R,U,V)* is derived from the matrix factorization and extracts latent knowledge from the rating matrix. The functions *L(X,U)* and *L(Y,V)* are generated using deep learning to join the side information with the latent factors.

5.2.2.3 Deep Matrix Factorization

Deep Matrix Factorization is a deep learning technique with neural network architecture. In this method, a user-item matrix is constructed with the user-item ratings and the implicit feedback that is non-preferential. With this constructed matrix as the input to the system, a deep learning structure is created, which enables to learn a low dimensional user-item representation [265]. A loss function is introduced to optimize the system to its full capacity, which includes both implicit feedback and explicit user-item ratings as the input. The architecture of Deep matrix Factorization is shown in Figure 5.7.

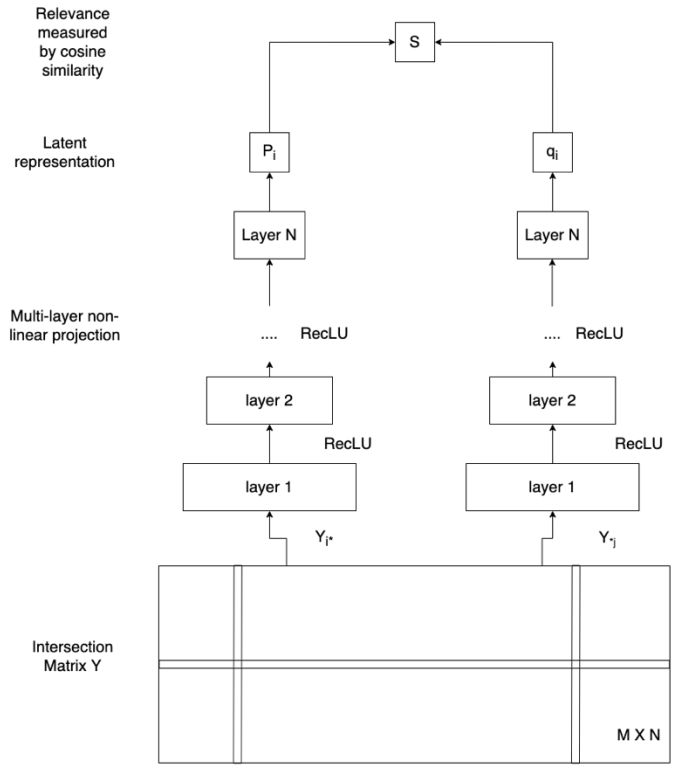


Figure 5.7 Architecture of Deep Matrix Factorization

The results obtained in the study [103] showcased the efficacy of the proposed algorithm and the introduced loss function, which was compared to many benchmark datasets and algorithms. The model can be mathematically expressed as follows:

$$Y_{ij} = \begin{cases} 0, & if\ R_{ij} = unk \\ 1, & otherwise \end{cases} \tag{5.2}$$

$$Y_{ij} = \begin{cases} 0, & if\ R_{ij} = unk \\ R_{ij} & otherwise \end{cases} \tag{5.3}$$

In Equations 5.2 and 5.3, matrix Y is taken as the input and given into the deep learning model of the neural network to present users and items in the latent feature space.

5.2.2.4 Graph Convolutional Matrix Completion

Graph Convolutional Matrix Completion is a deep learning method that implements the graph auto-encoder architecture for matrix completion. This method uses the information obtained from the interaction between the users and items. In addition to this, this technique also includes the side information of both the users and items. The link predictions generated in a graph act as the basis of the matrix completion in a recommender system. A bipartite graph of users and items is used, which depicts the interaction data such as movie ratings for a movie dataset. The edges in the graph denote the user-item ratings. In this technique, introduced by authors in [266], a graph encoder architecture is created depending upon the differentiable message passing mechanism using the bipartite interaction graph. This model works particularly perfectly well on graphical representations of social networks.

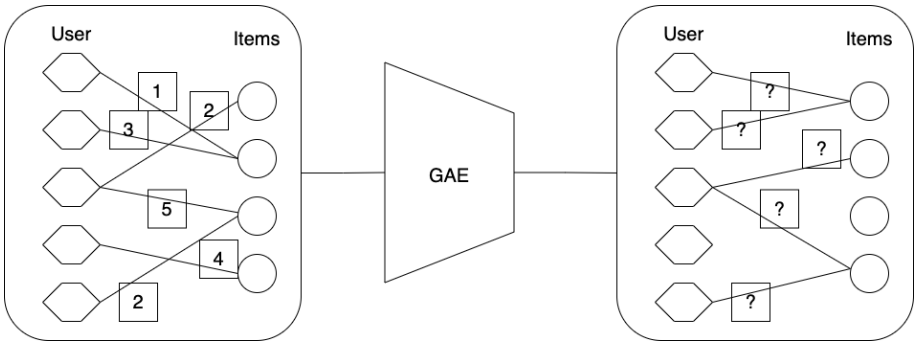


Figure 5.8 Representation of rating matrix M with user-item interactions. The left box represents the bipartite graph, GAE represents the Graph AutoEncoder and the right box represent the Link Prediction of users and items

As shown in Figure 5.8, the rating matrix M of size N_u *x* N_v, where Nu represents the number of users and N_v represents the number of items. The entries M_{ij} in matrix *M* encode either of the two things. i) observed rating (rating of user *u* for item *i*) from a set of possible ratings, ii) unobserved rating, which is masked by 0. Matrix Completion algorithm predicts the values of unobserved entries in matrix *M*, and hence, the recommendations are generated. The user-item interaction data is typically represented by an undirected graph, as shown in Equation 5.4:

$$G = \langle W, E, R \rangle \tag{5.4}$$

Where the user nodes are represented as in Equation 5.5:

$$u_i \in U, \qquad where \;\; i \in \{1, \ldots., N_u\} \tag{5.5}$$

And the item nodes are represented as in Equation 5.6:

$$v_j \in V, \quad where \; j \in \{1, \ldots., N_v\} \tag{5.6}$$

Such that,

$$U \cup U = W \tag{5.7}$$

With Equation 5.7, the edges of the graph represent the rating levels as shown in Equation 5.8:

$$\left(u_i, r, v_j\right) \in E \tag{5.8}$$

Where, rating *r* is represented by Equation 5.9:

$$r \in \{1, \ldots., R\} = R \tag{5.9}$$

5.3 Hierarchical Supervised Model

The meta-learning template suggests building hierarchical supervised learning models [267]. They can amalgamate ensemble-based algorithms and base machine learning or deep learning techniques brought together hierarchically. In the structure thus formed, the leaf nodes of the

structure are the base algorithms that are connected by ensembling nodes. The regression and classifier models will be further optimized as per certain data instances and their attributes. To carry out his task, the problem is broken down into several sub-problems and the final sub-solutions hence obtained are combined to form a unified solution to the main prediction problem. The method of breaking down the bigger problem into smaller ones depends upon the ensembling technique used. In a typical method, the dataset is distributed among all the participating models, and after the availability of outputs from every such model, they are combined to form the final output.

Upon implementation, the training dataset is forwarded to the top-level Bagging method, which generates *N* bootstrapped sub-datasets, also known as metabase, where *N* is the number of machine learning or deep learning models used. The subsequent bootstrapped training datasets implement a classifier by the Boosting method. The classifier shows a significant error rate and is probable to be implemented in training set for the second member method of the ensemble technique, i.e., Boosting in this case which stacks the two classifiers. The classifiers now placed at the bottom are assessed on the training data, and pertaining to the output, another meta-model is trained. The stacking method of ensembling is implemented, and the output thus produced is assigned a weight in Boosting. The final output of Boosting is averaged along with all the other top-level base models, and that is how the final classifier is formed. The meta-learning architectures are hierarchical, whereas the data mining architectures are directed acyclic graphs in structure. As previously mentioned, the inner nodes of the graph in our model are ensembling algorithms, and the leaf nodes in the graph are the base recommender algorithms.

5.4 AutoML Generation of Optimal Ensemble Model

Implementing genetic programming revolves around solving four problems i. characterization of the individual, ii. Blueprint of genetic operators and genetic evolution, iii. Founding of fitness function and iv. Building of the preliminary population. In the genetic programming methodology, the individuals are denoted as nodes in trees, and encoding is easy. Every given template has inner nodes representing the ensembles and the leaf nodes representing the base recommender techniques. Generic templates have wildcards embedded in their chromosomes. These wildcards are characterized as a list of genomes. Out of all such genes, one is selected at random whenever

there is a need for offspring from a given template. If, for instance, ten base recommenders are to be created, the system randomly selects a gene ten times to get ten unique genes from a list of all the available algorithms. It is essential to deduce whether the generated metabase is good enough to carry forward the further process of evolution and the time stamp of the running of the algorithm. The algorithm also has many intrinsic attributes, most of which are adaptive. The time limit decides a large number of internal attributes because a faster template on small data instances can be implemented for smaller time durations. So, when a larger chunk of time is available, finding the best resulting meta-learning structure is improved, and hence a larger part of the search space can be explored. The dataset is presented to the algorithm as the input. A Database Characterization Tool (DCT) is created from selecting data instances from the dataset. Authors create an initial population of the initial evolution in [268], [269]. Whenever the metabase is not used, the base recommender models form the population. This is done so that every base recommender is considered before the ensembles are formed. With the help of the metabase, the initial population is populated with the best performers from most alike metadata. For successive evolutions, the population from the previous evolution is used. The initial evolution works on a minimized dataset, and a maximum number of designated evolutions are run. Afterward, the data is doubled, and the next evolution happens. After every successive evolution, the template becomes more distinct, and the number of wildcards is also minimized, which further increases the range of explored attributes due to the formation of more precise and distinct templates.

5.4.1.1 Fitness Evaluation

The fitness evaluation is taken place using multiple cross-validations [270], [271]. A template's fitness is equivalent to the average performance value of models created on the training dataset and implemented on the testing dataset. To assign more resources to better-performing candidates, the authors in [272], [273] allot additional resources. Once fitness evaluation takes place, the process of selection is carried out. Authors in the paper [243] used only the mutations and not the crossovers for selection criteria. The mutations hence done affect the topology and attributes of the structures. The structural mutations are implemented using context-free grammar [274], which determines the growth of structures from simple base classifiers to huge hierarchical ensembles, which consist of regression ensemble sub-trees. After enabling a metabase, the population of other templates is stemmed out from this metabase. Upon implementing the algorithm, the templates

containing one base algorithm are assessed on the dataset and saved in the metabase. The results obtained are used as landmarking parameters [275] and form meta-features. Afterward, the meta-feature vector is compared with other vectors which are present in the metabase, and the results matching the most are given as the output. The records consist of the best-performing templates added to the initial population. The fitness of every structure is then updated while evolution takes place, and once the optimization process is over, the final structures are stored as new records in the metabase.

5.5 Results and Analysis

The results obtained after implementation have been discussed in this section. Table 5.1 presents the comparative analysis of the Root Mean Square Error (RMSE) error metric for all three instances of the MovieLens dataset, i.e., 100K, 1M, and 10M in Figure 5.9(a), 5.9(b), and 5.9(c) respectively. Our proposed method En-DLR (Ensemble Deep Learning Recommender), is compared with other benchmark deep learning techniques for recommendations. The compared algorithms are Graph Convolutional Matrix Completion (GC-MC), Deep Matrix Factorization (DMF), mSDA-CF (Deep Collaborative Filtering), and EnPSO (Ensemble Particle Swarm Optimization). In this study, we addressed the computational challenge of model selection for achieving the best results. Hence, we used AutoML to enable automated model selection.

Table 5.1 (a) RMSE error metric for 100K MovieLens dataset (b) RMSE error metric for 1M MovieLens dataset (c) RMSE error metric for 10M MovieLens dataset

METHODS	RMSE
GC-MC	0.899
DMF	0.905
mSDA-CF	0.903
ConvMF	0.904
En-PSO	0.918
En-DLR	0.885

(a)

METHODS	RMSE
GC-MC	0.854
DMF	0.832
mSDA-CF	0.841
ConvMF	0.854
En-PSO	0.883
En-DLR	0.813

(b)

METHODS	RMSE
GC-MC	0.799
DMF	0.777
mSDA-CF	0.787
ConvMF	0.793
En-PSO	0.852
En-DLR	0.752

(c)

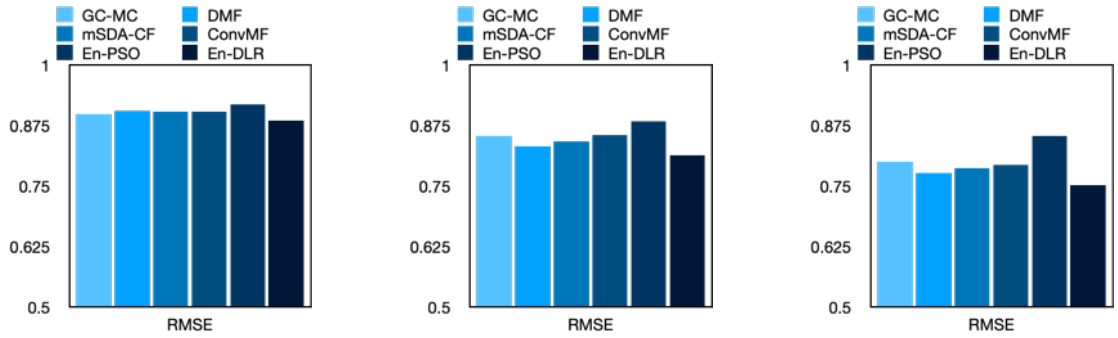


Figure 5.9 (a) RMSE error analysis for 100K MovieLens dataset (b) RMSE error analysis for 1M MovieLens dataset (c) RMSE error analysis for 10M MovieLens dataset

It can be seen from the results that the algorithm proposed in this study (En-DLR) has the minimum error, i.e., minimum RMSE values. Since the error values of our experiments have the lowest error values compared with other benchmark deep learning recommendation techniques, we can conclude that our study has provided the best possible results on the given instances of MovieLens datasets.

5.6 Chapter Summary

The results show that using the proposed framework, a practitioner can combine state-of-the-art recommendation techniques to get even better results. The ensemble generated not only has the biases of the base recommenders but is, in general, a more robust system. The research work is a proof of concept that creating ensembles can be automated and generate optimal results. This system works on the state-of-the-art, deep learning recommenders for the movielens dataset. A diverse set of base recommenders were selected based on matrix factorization, collaborative filtering, and graph-based approaches to create a well-rounded ensemble that can benefit from such diversity. The generated ensemble is a hierarchical model which combines results using various ensemble techniques in a bottom-up tree structure. Genetic programming was used to optimize the best ensemble. One limitation of all the implemnetations are the problem of cold start and data sparsity which are remediated in the forthcoming chapters.

Chapter 6

ALLEVIATING DATA SPARSITY FROM DYNAMIC RECOMMENDER SYSTEM

Recommender systems have become a core part of the retail experience. Retailers often rely on recommender systems to help them drive more conversions through targeted communication and advertisements. However, recommender systems are not one size fits all. Specialized retailers require specialized recommender systems to consider various features, attributes, and dynamics about the product category. A novel fruit recommender system is proposed that generates dynamic recommendations while remediating the problem of data sparsity. A novel fruit recommender system is developed that considers the temporal dynamics in the fruit market, like price fluctuations, fruit seasonality, and quality variations that occur throughout the year. To perform this task, Recurrent Recommender Network (RRN) is used which uses the deep learning method Long Short-Term Memory (LSTM) to implement the system model. To ensure that the work and results obtained are practical, the system has worked in a real-world setting, by tying up with a specialty fruit retailer based in New Delhi to get the real-world Point-of-Sale (POS) data of consumers. The result of the study suggests the proposed algorithm performs better than other benchmark algorithms along NDCG and RMSE metrics.

Section 6.1 discusses an in-depth understanding of the dynamic fruit recommender systems, followed by Section 6.2 which brings to light the problem of data sparsity in detail. Section 6.3 describes the Recurrent Recommender Network (RRN) and the implementation of the system. The results and their analysis is provided in Section 6.4 followed by summary in Section 6.5.

6.1. Dynamic Fruit Recommender System

The retail industry has found recommender systems very helpful. Their use has been growing year over year, so much so that it has become a core part of their business model for many retailers, especially those that are more data-driven. Whether online or offline, recommender systems have found a use case in the retail industry. Our work is based on solving problems for one such retailer. Recommender systems are not one size fits all. To get the best out of them, they should be tailored

to the products and the market. Our research is based on creating a tailored solution for fruit retail. Application of the recommender system in fruit retail is challenging as it has to consider various temporal changes in the fruit market over the year. These changes include the shift in demand based on seasonality, price, and quality variations of fruit throughout the year. Fruit recommendation [276] has barely been studied in academia, primarily because of a lack of data to solve this problem. We have partnered with Native Picks, an exotic fruit retailer based in Delhi, to obtain the data under confidentiality clauses. Using this data, our research aims to understand the dynamic in the fruit market and design a novel recommendation framework best suited for such a retail environment. Dynamic recommender systems are being widely used in recent times to generate efficient on-the-go recommendations. There exist several studies in which dynamic recommender systems have been used. To predict the subsequent preference of the user from cross-session data, the authors [277] created an item-based hierarchical deep learning architecture based on a dynamic recommender system. For this, the authors used Hierarchical Temporal Convolutional Network taking into account the short-term interactions and long-term predilection dynamically to generate recommendations using XING and Pinterest datasets. Treading on a similar concept, the authors [278] realized that convolutional networks model sequential interactions exceptionally well. To mirror similar behavior based on the current input, they implemented DynamicRec algorithm for dynamically predicting the next item. To efficiently learn the network embeddings from dynamic networks, the authors [279] introduced a neural recommendation technique to model the dynamic network interactions. To learn from the interaction between the social contacts, which are static in nature, the authors [280] introduced a dynamic recommender model implementing the random lazy walk that used FourSqaure to predict the top-n locations to users. Authors [281] predicted a 5-day recommendation for consuming fruits and vegetables in middle-aged French people concerning several parameters like socio-economic growth, demography, and behavioural factors. The study was carried on for two years spanning over more than four thousand participants. It was observed that the recommendation was adhered to primarily by women, physically active, non-smokers, and highly educated people. On similar grounds, another study [282] was carried out for analyzing the 5-day recommendation based on analyzing oxidative damages and antioxidant defense. It was observed that people with low oxidative stress and enhanced antioxidative status were the ones who followed the recommendations. Another study [283] analyzed the role of consuming fruit juice to achieve the

recommended target of consuming fruits and vegetables. Similarly, the study [284] researched the availability trend of fruits and vegetables across ten European nations to follow the 5-day fruits and vegetables recommended diet.

6.2. Data Sparsity

Whenever a new item is introduced into the system, it does not have any pre-associated tags with it describing which users will like that item. Due to unavailability of information about that item, it suffers from a problem called cold start [285] . This problem exists not only for new items, but for new users as well. The problem of cold start hampers the process of generating recommendations resulting in sparse data due to unavailability of <user-item> tags for items and users suffering from cold start. To remediate the problem of data sparsity, we have used Recurrent Recommender Network (RRN) [286] in our model. Other methods learn the parameters to be used in the system for model training purposes. But in the proposed model, the system learns the functions which identify the parameters. By doing this, the statistical power is shared across all data points and hence, no item suffers from cold start problem, rendering the system free of the problem of data sparsity, making the system a dense network. To use the Point-of-Sale (POS) data to help with the sales in any offline store, the authors [287] worked towards generating shopping list predictions for every customer. To carry out this task, they trained and learned from dedicated classifiers for every customer pertaining to the previous transaction data. In a similar study [288], the authors studied the application of recommender systems in detail for offline POS systems for retail businesses.

6.2.1. Dataset

The dataset contains all the sales and inventory information captured from the point of sale system at Native Picks, New Delhi. The sales data show the dynamic nature of the fruit market. As we can see from the data, the prices of various fruits are volatile and significant change is observed month to month. This volatility is due to the dynamics of the supply chain. For example, there is only a specific window when new Zealand apples are harvested in India. There is a peak in demand for apples during that time, and we see higher prices being fetched. For the other part of the year, apples from New Zealand are not available, and apples from the USA or Chiles are dominant in the Indian market. Demand and supply for these apples follow a different curve, which is reflected

in the data. A recommender system that can take into account these dynamics can recognize these trends and recommend people with more relevant fruits and drive higher conversions. Table 6.1 depicts the description of the dataset.

Table 6.1 Dataset Description

Dataset Columns	**Description**
Order id	Unique order attribute
Customer id	Unique customer attribute
List <Order Line Item>	List of order line items consisting SKUs, their quantities and relevant tax and price information
Total Invoice Value	Total invoice value
Payment id	Unique identifier linking to the payment details for the corresponding order
List<Coupon id>	List of coupon code used
Timestamp	Timestamp

The fruits dataset of the exotic fruits retail store Native Picks consists of point-of-sale fruit purchase data for over 10,000 customers spanning more than two years. During this time, 2,000 customers shopped from 10 and 200 times. This number of customers underwent sampling to generate a dataset with 14,600 respective transactions. For finding out the total transactions for every customer, uniform random sampling to select customers with 90 percentile delivers a skewed sample set as compared with customers having much lesser percentiles. As an illustrative sample, we distribute the data points across three parameters: gross billing amount, the total number of transactions, and ticket size of every transaction. For every amount spent on a dedicated chunk of the dataset, one-tenth of the data points were shortlisted with an invariable probability from each chunk. The result acquired by each chunk was similar to one another, and the final sample used was taken from the gross billing amount. The transactional information in the data includes the lists of products purchased in every transaction and the associated parameters. Table 6.2 describes the record information of the dataset used in our study.

Table 6.2 Records Information of the Dataset

Records Type	Number of Records
Customers	10,427
Transactions	18,421
SKUs	2,314

6.3. Recurrent Recommender Network (RRN)

A recommendation framework is developed that can capture the temporal dynamics of both user and item.This recommendation framework uses Recurrent Neural Network (RNN) [22] at its core (as a part of RRN). Recurrent Neural Network differs from other proposed temporal models [289] as it is a model without attributes that can conclude future behavior by learning the fruit consumption pattern of the customers. To enable the model with memory, the Long Short-Term Memory (LSTM) [290] is used, which can be recalled as an RNN with memory cells. As conclusive in this study, Recurrent Recommender Network (RRN) [286] does not suffer from data sparsity. The proposed model learns the functions that discover attributes rather than learning the attributes directly. In this way, the numerical strengths are divided among every data point. In a dense network, Recurrent Recommender Network results in a smaller size and exemplifies an equivalent or improved accuracy. The system is designed to work with POS data, readily available at all retail stores, whether online or offline. Designing the algorithm based on this data gives the advantage of the model being easy to plugin within any retail environment. To create recommendations from POS data, it is processed into ratings based on user interaction with the item. These ratings can now be used in a recommender system algorithm for predictions. Since it is interesting to draw recommendations based on temporal dynamics of both user and item, the system has chosen to use Recurrent Recommender Network (RRN), which can make recommendations based on temporal dynamics. Figure 6.1 represents the recommendation framework used in this study.

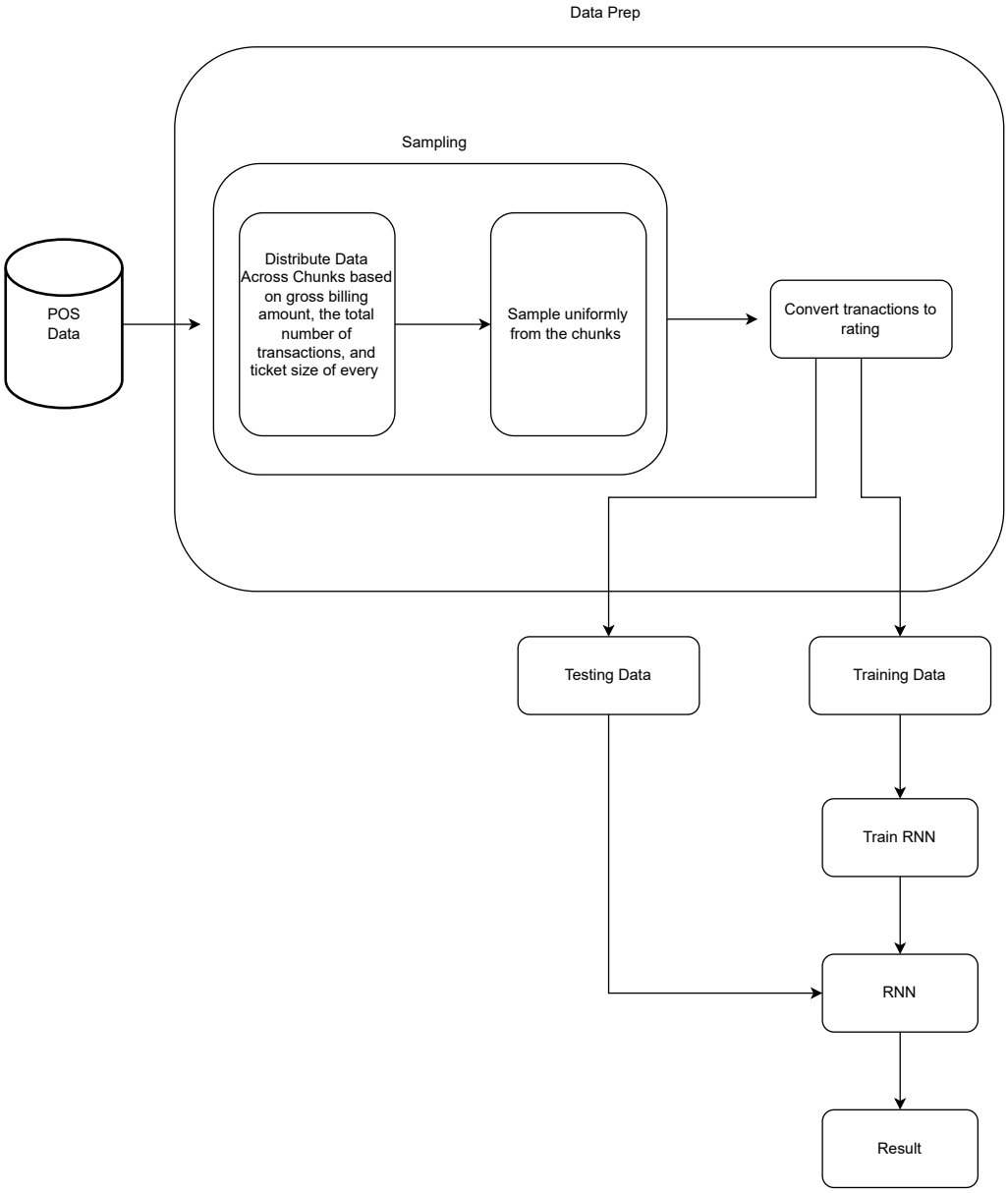


Figure 6.1 Recommendation Framework of the proposed model

6.3.1. Frequency to Rating

It was realized that deducing the ratings required estimating regular fruit consumption by every customer. This is the sole information consisting of implicit feedback of customers [291]. The purchase frequency for a given fruit *f* for customer *j* is described in the following Equation 6.1.

$$Freq_{f,j} = \frac{p_{f,j}}{\sum_{f'} p_{f',j}} \tag{6.1}$$

Here,

Where $p_{f,j}$ represents the number of times customer j has consumed fruit f

$$r_{i,j} = 4\left(1 - \sum_{k'=1}^{k-1} Freq_{k'}(j)\right) \tag{6.2}$$

In Equation 6.2, $Freq_k(j)$ represents the kth most consumed fruit for customer j. Afterward, a rating for a fruit having rank k is calculated as a linear function of the frequency percentile.

After computing the ratings, collaborative filtering can be implemented for the dataset consisting of explicit user preferences [291].

6.3.2. Dynamics of the System Model

For the graphical model inscribed in the Figure, the most challenging aspect is its requirement to deduce the future states pertaining to the current observations. This challenge increases the cost and difficulty of matching emission models of ratings with the latent state. One of the methods to input ratings back into the latent space can be described as the probability $p(r_{ij}|u_{it}, m_{jt})$ [286]. A few ways to carry out this task are Message Passing and Particle Filtering. These methods are imprecise but crucial to achieving accuracy at scale. However, the mapping can also be learned as part of a non-attribute state update. For example, by using a Recurrent Neural Network (RNN). This is essentially done to use a latent variable autoregressive model as seen in the following Equations 6.3 and 6.4:

$$\hat{z}_{t+1} = f(h_t, z_t) \tag{6.3}$$

$$h_{t+1} = g(h_t, z_{t+1}) \tag{6.4}$$

Here, z_t represents the value at time t,

$\hat{z}_t$ represents the corresponding estimate, and

h_t represents the latent state

The introduction of non-linearities and techniques to ensure stability and disappearing gradients enable the equations to be efficient. However, the generic functional form has been cited in [292]. One of the most popular ways to carry out this task is Long Short-Term Memory (LSTM) [293].

It encapsulates the temporal dynamics and uses it as a building block for a collaborative filtering (CF) system. The state updates implement the operations calculated by Equations 6.5, 6.6, 6.7, and 6.8 to capture the temporal dynamics.

$$[f_t\,, i_t\,, o_t] = \sigma[W[h_{t-1}\,, z_t] + b] \tag{6.5}$$

$$l_t = tanh\,[V[h_{t-1}\,, z_t] + d] \tag{6.6}$$

$$c_t = f_t \cdot\, c_{t-1} +\, i_t \,\cdot\, l_t \tag{6.7}$$

$$h_t = o_t \,\cdot tanh(c_t) \tag{6.8}$$

here f_t represents the forget gate,
i_t represents the input gate, and
o_t represents the output gate

They administer the information flow throughout the sequence. For ease of understanding, the following Equation 6.9 is used to depict the aforesaid operations.

$$h_t = LSTM(h_{t-1}\,, z_t) \tag{6.9}$$

This study uses LSTM Recurrent Neural Networks to capture time-based dependencies for both the consumers and fruits. Hence, we captured the past observations and deduced the future trajectories in a unified way by doing this. We use the following Equations 6.10, 6.11, and 6.12 to calculate the following update functions functions at time *t+1*. Given the values of functions at time t, $u_{i\,,t+1}$ and $f_{j\,,t+1}$ calculate the updated values To implement temporal dynamics, we use a time index, i.e., u_{it} and f_{jt}.

$$\hat{r}_{ij|t} = f(u_{it}\,, f_{jt}) \tag{6.10}$$

$$u_{i\,,t+1} = g\big(u_{it}\,, \{r_{ij|t}\}\big) \tag{6.11}$$

$$f_{j\,,t+1} = h\big(f_{jt}\,, \{r_{ij|t}\}\big) \tag{6.12}$$

Here, u_{it} represents the latent attributes for customer *i* at time *t*
f_{jt} represents the latent attributes for fruit *j* at time *t*

$\hat{r}_{ij|t}$ represents the predicted rating for customer *i* for fruit *j*, and

$r_{ij|t}$ represents the actual rating, at time step *t*.

The functions *f(), g(),* and *h()* are implemented to deduce the new user's likes without applying any optimization processes. Instead of resolving the optimization criteria to make out the user parameters, in this study, we have solved that problem to identify the functions that discover the user variables. For example, the deep autoencoders in [294] implement an encoding function for previous ratings. The distinguishing factor in the study is that we aim to implement a function that consecutively refurbishes scores and allows the process of forwarding predictions for a dedicated set of ratings at any given time.

6.3.3. User State and Item State

To explain the model in-depth, the analogy of a user-state Recommender Neural Network is assumed as the fruit-state RNN. To explain the concept, the notion of having a user-state Recommender Neural Network is exemplified, as the fruit-state RNN is defined similarly. For the dataset of F fruits, the rating vector is represented as following for a given customer at time t.

$$x_t \in R^F$$

Here, $x_{tj} = k$ iff, the customer rated fruit *j* with score *k* at a given timestamp *t*, else

$$x_{tj} = 0$$

$$y_t := W_{embed}\,[x_t\,, 1_{new}\,, \tau_t\,, \tau_{t-1}] \tag{6.13}$$

Here in Equation 6.13,

W_{embed} represents the evolution of source information into embedding space,

The wall clock is denoted as τ_t at any given timestamp *t*,

This study uses $1_{new} = 1$ to realize the novelty of the user

This yields y_t that is given as the input to a Long Short-Term Memory (LSTM) at a given timestamp *t* leading us to the model realized in Equation 6.14.

$$u_t := LSTM\,(u_{t-1}\,, y_t) \tag{6.14}$$

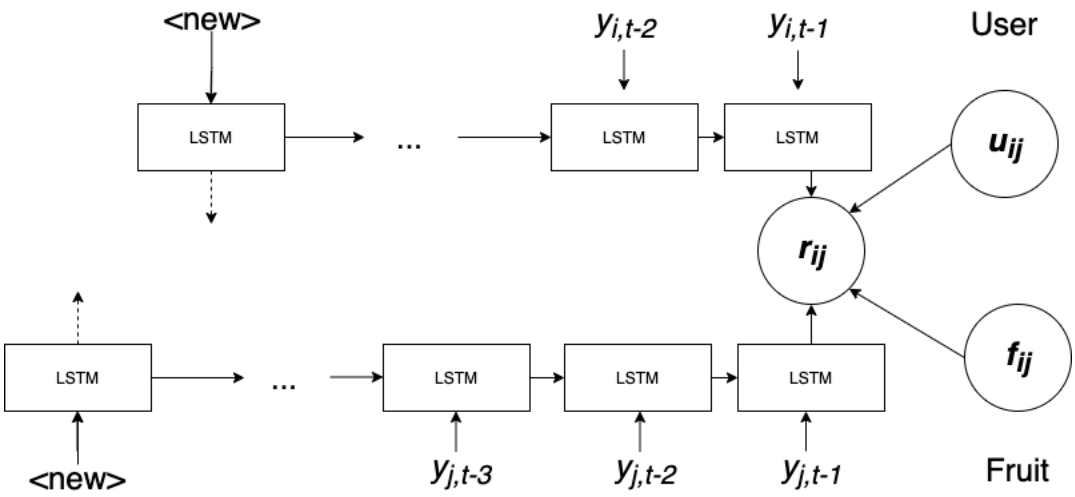


Figure 6.2 The proposed system model

Figure 6.2 represents the system model used in the study. The points where there has been no rating by the consumers are not included in the RNNs to reduce the number of computations. However, upon adding the wall clock, the model receives the essential to suffice for the no-rating steps and identify parameters like rating scale change and fruit consumption. To alter between customer's purchases, a unique index i is used in this study for customer u at a given time t in u_{it}. Similarly, for fruits, we use f_{jt} for fruit j at time t.

6.3.4. Rating Emissions

Since the states of consumers and fruits are temporal, we assume that specific fixed parameters exist to encode certain properties like the profile, time-dependent preference of a customer, and much more. To carry out this task, we add the stationary profile vectors u_i and f_j with the time-varying profile vectors, i.e., u_{it} and f_{jt} with respectively. In other words, we conceive that the rating is a function of both dynamic and static states, as shown in Equation 6.15.

$$\hat{r}_{ij|t} = g\left(u_{it}, f_{jt}, u_i, f_j\right) := \langle \tilde{u}_{it}, \tilde{f}_{jt} \rangle + \langle u_i, f_j \rangle \tag{6.15}$$

Where, $\tilde{u}_{it}$ and $\tilde{m}_{jt}$ are affine functions of u_{it} and f_{jt} respectively as depicted in Equations 6.16 and 6.17.

$$\tilde{u}_{it} = W_{user}\, u_{it} + b_{user} \tag{6.16}$$

$$\tilde{f}_{jt} = W_{fruit}\, f_{jt} + b_{fruit} \tag{6.17}$$

In other words, the classic factorization incorporates stationary effects, and this study implements LSTMs for higher range dynamic updates. This renders our model a superset of the other widely used Matrix Factorization (MF) recommender systems.

6.3.5. Rating Prediction

Unlike the conventional recommender systems for rating prediction, this study uses the extrapolated states in prediction time rather than the estimated states. The model considers the latest observations as input, updates the states, and generates predictions pertaining to the latest updated states. Hence, the factor of causal effects is considered, which is brought in by the past ratings. This can be explained by the concept of hedonic adaptation [289]. As an example for a fruit recommender system, hedonic adaptation can be described as reducing the level of satisfaction after consuming a fruit once someone consumes an even better tasting and fulfilling fruit.

6.3.6. Inference

To have optimized results, the model requires to deduce attributes that generate predictions that are similar to the actual ratings, as shown in Equation 6.18.

$$\min_{\theta} \sum_{(i,j,t)\in \chi_{train}} \left(r_{ij|t} - \hat{r}_{ij|t}(\theta)\right)^2 + R(\theta) \tag{6.18}$$

Here,

θ represents all attributes to be learned,

χ_{train} represents the set of observed tuples in the training set, i.e., customer, fruit, timestamp, and,

R represents the regularization function

In this study, we have used a traditional objective function that implements backpropagation challenges. The key point is that every set of ratings rely on user-state Recommender Neural Network (RNN) and fruit-state RNN. However, executing backpropagation in simultaneous successions for each rating is computationally exhaustive. This problem can be remediated slightly by back-propagating gradients pertaining to the ratings of a user. However, each rating would depend upon the purchase and the fruit's entire sequence even after that. Furthermore, we implemented an alternating subspace descent methodology that does not carry this limitation. In this method, the system back-propagates the gradients of the entire rating set of the user at once to refurbish user-sequence attributes and presume that the purchase state of fruits is constant. Hence, it becomes irrelevant to propagate gradients into those fruit sequences. Afterward, the system alternates amongst updating user sequences and fruit sequences. In this manner, the system implements the feed-forward and back-propagation for every user once at a time.

6.4. Results and Analysis

In this study, the Fruit Recommender Framework's capacity to model various time-based effects is exhibited and an accurate recommendation to the consumers by itself is generated. Especially, it is presented that the recommendation framework is able to take into account the temporal patterns and changes in both the item and user states and preferences. To understand the efficiency of modelling time-specific dynamics, the study implements the model on a real-world dataset that was, as mentioned before, obtained from a prominent fruit retailer in the New Delhi-NCR area. The data consisted of sales information along with item and user descriptors. Every data point is a (user id, item ids, quantity, timestamp) tuple. To explain the several aspects of the proposed model, the system performs the implementation on other temporal statistics. The model splits the data whenever it is required to predict future ratings instead of interpolating previous ratings. Such a set of temporal-based testing period ratings are uniformly distributed among validation and testing sets.

6.4.1. Setup

The proposed algorithm attains a pretty good accuracy with a limited number of parameters. The setup of the experiment has been carried out with a layer of 40 latent neurons, 40-dimensional input embeddings, and 20- dimensional dynamic states. We have implemented the algorithm on

MXNet [295]. This is an open-source platform typically used to implement deep learning models. To examine the efficiency of our system, we implemented the dataset on several other models.

6.4.1.3. Probabilistic Matrix Factorization (PMF)

It is widely used for its close-to-perfect rating prediction results. The system achieves mirrored results of factorization as achieved by PMF. In addition to that, the system also cites the advantages of modelling temporal dynamics [296].

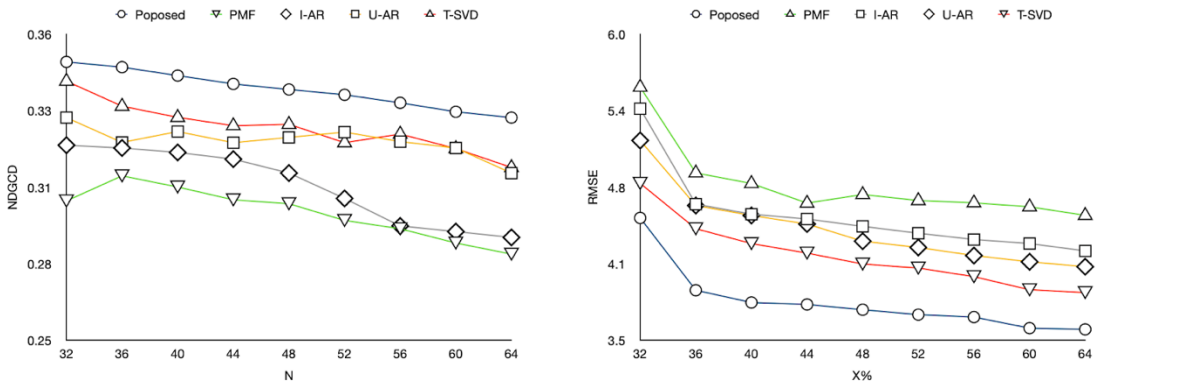
6.4.1.2. TimeSVD++

This algorithm is primarily used to model the temporal dynamics of the system and achieves outstanding results [289].

6.4.1.3. AutoRec

This algorithm encodes every data point into lower-dimensional space and decodes it to generate predictions [294].

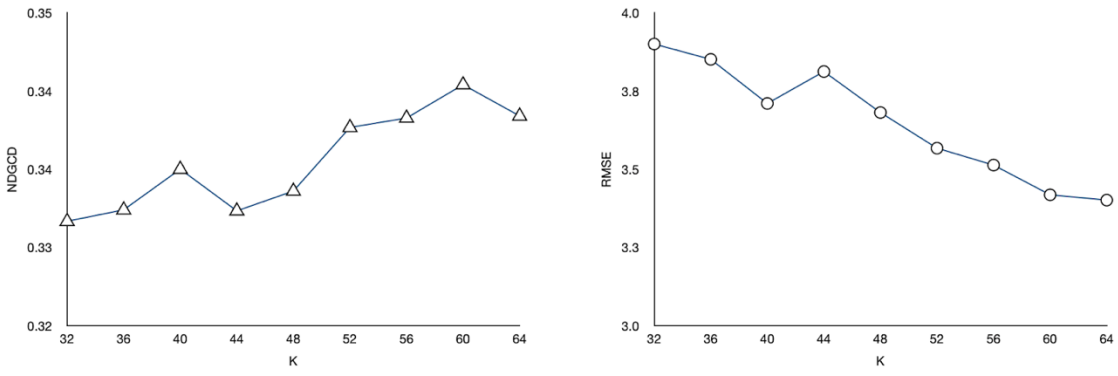
Recommendations generated are stored in a recommendation list, and by default, the list is sorted in decreasing order, the topmost entry being the most relevant recommendation and the bottommost being the least relevant recommendation in the list. Since people are most inclined towards the most relevant recommendation, thus we try to improvise the quality of those *x%* entries of the list. To carry out this task, the recommendation list *N* is altered to understand the result of Normalized Discounted Cumulative Gain (NDCG) [297] and the study the Root Mean Square Error (RMSE) [298] of the top *x%* recommendations generated in the recommendation list *N*.



(a) (b)

Figure 6.3 (a) Progression of NDCG with size of recommendation list (b) Progression of RMSE error with top x% of recommendation list

Upon comparing our proposed model with other models, as shown in Figure 6.3, the basic parameters, NDCG and RMSE, indicate our approach providing the best results as we can see in the result graphs as we vary *N* and *X%*. The model uses a bias parameter to capture and model the user-item change over time. It is usually seen that the affinity of a consumer for an item changes over a certain period of time, resulting in a variation in time and monetary interests. In our model, such variations are taken care of. Second to the efficiency of our model in TimeSVD++. It generates efficient recommendations with low computational cost.



(a) (b)

Figure 6.4 (a) Progression of NDCG with model order, K (b) Progression of RMSE error with model order, K

In our system, the item vector is a combination of tag-related attributes and other tag-unrelated attributes. In this study, the item vector is constituted of a concatenation of tag-related attributes and other attributes not covered by tags. Including tags improves the accuracy of the recommendations, but it becomes crucial to include tag-unrelated attributes as well. Hence, we capture the effect of model order, *K*. As seen in Figure 6.4, we keep the tag-related attributes constant to see how the result, i.e., values of NDCG and RMSE, vary with change in *K*.

6.5. Chapter Summary

The generated results have shown that the proposed framework provides an effective recommendation in the fruit domain. It can consider temporal factors like demand, supply, and seasonality associated with both the products and customers. The study has used a real-world dataset acquired from a prominent local boutique fruit retailer to carry out this task. To generate practical recommendations, the system has used RNN, which can take temporal changes in both user and item. However, RNN takes ratings as input, so the framework processes the point of sale data to generate ratings of items within a specific period by analyzing the customers' interactions with that particular item. This work is a proof of concept which can be extended to different domains and datasets.

Chapter 7

CONCLUSION AND FUTURE SCOPE

This chapter presents a comprehensive summary of the research work done. It includes the research summary of the work done in Section 7.1, and the limitations of the research study identified in Section 7.2. The chapter also presents the future aspects of the research work performed in Section 7.3 and how the study can help the future researchers in the said domain.

7.1. Research Summary

To fulfil the first objective, RO1, the research commenced with an extensive research survey in Recommender Systems. It was concurred that deep learning has become an inevitable primary module of any technical implementation and advancement and hence prior used simple machine learning algorithms are no longer used in the real-world applications of recommender systems. An in-depth analysis of 120 research papers implementing deep learning in recommender systems was performed and the various Machine Learning and Deep Learning models and techniques that can be and have been in the past implemented in Recommender Systems were studied. To further achieve this objective, a novel application of recommender systems was designed, wherein, the system provides food-wine pairing recommendations to the user. This study included the following:

- A novel framework to solve two-fold recommendation problem of food-wine parings
- Novel features were extracted using text mining and sentiment analysis
- We created and compiled two novel datasets for the process of feature extraction
- Results showed the resulting food-wine recommendations aligned with wine sommelier's food-wine recommendations

To fulfil the second objective, RO2, an ensemble system with Particle Swarm Optimization was proposed, which intelligently optimized the recommendations by identifying the best ensemble architecture for the data at hand. This study included the following:

- Proposed an AutoML framework for Ensemble Learning Recommendations (EnPSO: Ensemble with Particle Swarm Optimization)

- Employed evolutionary algorithm Particle Swarm Optimization (PSO) for finding the best model
- Analyzed EnPSO on three publicly available benchmark MovieLens datasets
- Implemented EnPSO on five benchmark machine learning techniques

In addition to this, in another implementation of this objective, ensemble learning was performed on deep learning techniques (En-DLR) and the most optimum ensemble was identified by applying genetic algorithm as the evolutionary algorithm.

To carry out the third research objective, RO3, state-of-the-art, deep learning recommenders were implemented on the Movielens dataset. A diverse set of base recommenders were selected based on matrix factorization, collaborative filtering, and graph-based approaches to create a well-rounded ensemble that can benefit from such diversity. The generated ensemble was a hierarchical model which combines results using various ensemble techniques in a bottom-up tree structure. Genetic programming was used to optimize the best ensemble. This study included the following:

- Proposed a recommender system to implement AutoML framework for Ensemble Learning (En-DLR: Ensemble based Deep Learning Recommender)
- Employed Genetic Algorithm to identify the most optimal model in the search space
- Generated conclusive results from the analysis on the benchmark MovieLens datasets
- Implemented the model on four benchmark deep learning techniques as base recommenders

To fulfil the fourth and final objective, RO4, the problem of data sparsity was remediated by using Recurrent Recommender Network (RRN) in the model. Other methods learn the parameters to be used in the system for model training purposes. But in the proposed model, the system learns the functions which identify the parameters. By doing this, the statistical power is shared across all data points and hence, no item suffers from cold start problem, rendering the system free of the problem of data sparsity, making the system a dense network. This study included the following:

- Created a novel recommendation framework for generating dynamic fruit recommendations using deep learning based LSTM network.

- Alleviated the problem of data sparsity with the implementation of Recurrent Recommender Network.
- Developed a dynamic recommender system to dynamically incorporate the temporal changes in fruit seasonality variations and user preferences.
- Used a real-world Point-Of-Sale dataset of a commercial fruit retailer for implementing the system.

7.2. Limitations of the Study

The research work done comprised of certain drawbacks and limitations which are discussed as follows:

- Creating an ensemble is a challenging task on its own. There are endless ways to combine techniques and their outputs. For an ill-defined model, combining the base techniques in the search space for the parameters can be very large and require extensive training data and a lot of time to optimize.
- Even with the right model, the search will still have a large number of dimensions across which the system is required to search for the optimal ensemble.
- For the food-wine recommender system, although extensive feature engineering was performed, most of the features are derived from reviews and are abstract, like flavor feature and aroma feature as there is no quantitative source for this data. It is possible to improve the quality of recommendations if data for the features become available quantitatively.
- The recommender system employed in food-wine pairwise recommender is based on content-based recommendation, and they are not studied it in a social network setting or multiple users. Such exploration can lead to a better hybrid recommendation model incorporating collaborative filtering, graph-based recommender systems, and much more.
- The significant challenges in creating a recommendation framework from real-world POS data are, firstly, the temporal dynamics concerning both user and item state. Secondly, the problem of data sparsity since a user interacts with only a limited number of SKUs is comparatively less than the total number of SKUs in the store. The third is in incorporating sales data in the recommendation framework. There are two primary reasons for consumers to switch to different fruits and thus bring a drastic change in their preferences:

- Shift in User Interest:

  User interest is crucial aspect that affects the sale of fruits. It is as simple as some people like apples and some people do not. Hence, another thing needs to be kept in mind that people's choices vary over time. It might so happen that a person is not a fan of kiwis, but after having a taste of succulent and sweet New Zealand kiwis, he develops a liking towards them.

- Seasonal Variation:

  The affinity of people towards fruits pertains to the temporal aspects as well. Not only is it due to the availability of such seasonal fruits, but it is also how those fruits affect the human body. For example, watermelon is thoroughly relished and enjoyed in summers due to its high water content, which upon consumption, increases the water content of the body and maintains the electrolyte balance.

7.3. Future Aspects

Following are the future aspects of the research work performed:

- Results for AutoML implementation can be improved by using more advanced models for the ensemble with hyper-parameter optimization.
- More complex evolutionary strategies can be considered for finding the best models. Apart from working on model generation, research on AutoML can also be done in data preparation, feature engineering, and model evaluation. Thus, creating a complete platform to make recommender systems very trivial to implement for the user by encapsulating all the complexities within an AutoML framework.
- For the food-wine recommender system, although extensive feature engineering is done, most of the features are derived from reviews and are abstract, like flavor feature and aroma feature as there is no quantitative source for this data. It is possible to improve the quality of recommendations if data for the features become available quantitatively.
- The recommender system employed in food-wine pairwise recommender is based on content-based recommendation, and it is not studied it in a social network setting or multiple users. Such exploration can lead to a better hybrid recommendation model incorporating collaborative filtering, graph-based recommender systems, and much more.

- For the En-DLR system, future work in this domain includes exploring non-hierarchical structures for the ensembles and different ways to optimize them. It is also essential to see how this framework performs across domains.
- The proposed dynamic fruit recommendation framework using POS data can be extended to cover all such retail settings where the retailer can plug in the readily available POS data that is available with them and, through it, can generate effective recommendations for their customers. Using better techniques, researchers can create an integrated recommender system that does not need to convert POS to rating and digest the data as it is effective and process it to generate recommendations. Researchers can also implement session-based recommendations as well to further improve the accuracy.

References

[1] K. jae Kim and H. Ahn, "A recommender system using GA K-means clustering in an online shopping market," *Expert Syst. Appl.*, vol. 34, no. 2, pp. 1200–1209, 2008, doi: 10.1016/j.eswa.2006.12.025.

[2] X. Zhang, H. Liu, X. Chen, J. Zhong, and D. Wang, "A novel hybrid deep recommendation system to differentiate user's preference and item's attractiveness," *Inf. Sci. (Ny).*, vol. 519, pp. 306–316, 2020, doi: 10.1016/j.ins.2020.01.044.

[3] H. N. Peterson J. J., Yahyah M., Lief K., "Predictive Distributions for Constructing the ICH Q8 Design Space, pp. 55-70, In Comprehensive Quality by Design for Pharmaceutical Product Development and Manufacture," 2017, doi: 10.1145/1143844.1143865.

[4] X. Zhu, "Semi-Supervised Learning Literature Survey Contents," *Sci. York*, vol. 10, no. 1530, p. 10, 2008, doi: 10.1.1.146.2352.

[5] P. Grant, "Assessment and Selection," *Bus. Giv.*, 2014, doi: 10.1057/9780230355033.0018.

[6] T. Degris, O. Sigaud, and P. H. Wuillemin, "Learning the structure of factored Markov decision processes in reinforcement learning problems," *ACM Int. Conf. Proceeding Ser.*, vol. 148, pp. 257–264, 2006, doi: 10.1145/1143844.1143877.

[7] R. Van Meteren and M. Van Someren, "Using Content-Based Filtering for Recommendation," *ECML/MLNET Work. Mach. Learn. New Inf. Age*, pp. 47–56, 2000, doi: 1011255743.

[8] Y. Koren and R. Bell, "Advances in Collaborative Filtering - Recommender Systems Handbook," *Recomm. Syst. Handb.*, pp. 145–186, 2011.

[9] R. M. Sallam, M. Hussein, and H. M. Mousa, "Improving collaborative filtering using lexicon-based sentiment analysis," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 2, pp. 1744–1753, 2022, doi: 10.11591/ijece.v12i2.pp1744-1753.

[10] O. Azeroual, "RecSys Pertaining to Research Information with Collaborative Filtering Methods : Characteristics and Challenges," 2022.

[11] V. Vekariya and G. R. Kulkarni, "Hybrid recommender systems: Survey and

experiments," *2012 2nd Int. Conf. Digit. Inf. Commun. Technol. its Appl. DICTAP 2012*, pp. 469–473, 2012, doi: 10.1109/DICTAP.2012.6215409.

[12] M. Baidada, K. Mansouri, and F. Poirier, "Hybrid Filtering Recommendation System in an Educational Context," *Int. J. Web-Based Learn. Teach. Technol.*, vol. 17, no. 1, pp. 1–17, 2021, doi: 10.4018/ijwltt.294573.

[13] J. M. Pak, "Hybrid Interacting Multiple Model Filtering for Improving the Reliability of Radar-Based Forward Collision Warning Systems," *Sensors*, vol. 22, no. 3, 2022, doi: 10.3390/s22030875.

[14] R. A. E. D. Ahmed, M. Fernández-Veiga, and M. Gawich, "Neural Collaborative Filtering with Ontologies for Integrated Recommendation Systems," *Sensors*, vol. 22, no. 2, pp. 1–26, 2022, doi: 10.3390/s22020700.

[15] M. Ben Ahmed and A. A. Boudhir, "Innovations in Smart Cities and Applications," *Proc. 2nd Mediterr. Symp. Smart City Appl.*, no. January, pp. 1–1046, 2018, doi: 10.1007/978-3-319-74500-8.

[16] H. Zhang, Y. Xiao, and Z. Bu, "Personalized book recommender system based on Chinese library classification," *Proc. - 2017 14th Web Inf. Syst. Appl. Conf. WISA 2017*, vol. 2018-Janua, pp. 127–131, 2018, doi: 10.1109/WISA.2017.42.

[17] J. Bobadilla, F. Serradilla, and A. Hernando, "Collaborative filtering adapted to recommender systems of e-learning," *Knowledge-Based Syst.*, vol. 22, no. 4, pp. 261–265, 2009, doi: 10.1016/j.knosys.2009.01.008.

[18] R. Katarya, "Movie recommender system with metaheuristic artificial bee," *Neural Comput. Appl.*, vol. 30, no. 6, pp. 1983–1990, 2018, doi: 10.1007/s00521-017-3338-4.

[19] A. Almahairi, K. Kastner, K. Cho, and A. Courville, "Learning distributed representations from reviews for collaborative filtering," *RecSys 2015 - Proc. 9th ACM Conf. Recomm. Syst.*, pp. 147–154, 2015, doi: 10.1145/2792838.2800192.

[20] A. M. Elkahky, Y. Song, and X. He, "A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems," *Proc. 24th Int. Conf. World Wide Web - WWW '15*, pp. 278–288, 2015, doi: 10.1145/2736277.2741667.

[21] S. Seo, J. Huang, H. Yang, and Y. Liu, "Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction," *Proc. Elev. ACM Conf. Recomm. Syst. - RecSys '17*, pp. 297–305, 2017, doi: 10.1145/3109859.3109890.

[22] T. Donkers, B. Loepp, and J. Ziegler, “Sequential User-based Recurrent Neural Network Recommendations,” *Proc. Elev. ACM Conf. Recomm. Syst. - RecSys ’17*, pp. 152–160, 2017, doi: 10.1145/3109859.3109877.

[23] B. Purkaystha, T. Datta, M. S. Islam, and Marium-E-Jannat, “Product recommendation: A deep learning factorization method using separate learners,” *20th Int. Conf. Comput. Inf. Technol. ICCIT 2017*, vol. 2018-Janua, pp. 1–5, 2018, doi: 10.1109/ICCITECHN.2017.8281852.

[24] L. Zheng, V. Noroozi, and P. S. Yu, “Joint Deep Modeling of Users and Items Using Reviews for Recommendation,” pp. 425–433, 2017, doi: 10.1145/3018661.3018665.

[25] T. K. Paradarami, N. D. Bastian, and J. L. Wightman, “A hybrid recommender system using artificial neural networks,” *Expert Syst. Appl.*, vol. 83, pp. 300–313, 2017, doi: 10.1016/j.eswa.2017.04.046.

[26] X. Wang, X. He, L. Nie, and T.-S. Chua, “Item Silk Road: Recommending Items from Information Domains to Social Users,” pp. 185–194, 2017, doi: 10.1145/3077136.3080771.

[27] H. Zhu *et al.*, “Learning Tree-based Deep Model for Recommender Systems,” 2018, doi: 10.1145/3219819.3219826.

[28] Y. Lu, R. Dong, and B. Smyth, “Coevolutionary Recommendation Model: Mutual Learning between Ratings and Reviews,” *Www*, pp. 773–782, 2018, doi: 10.1145/3178876.3186158.

[29] K. J. Oh, W. J. Lee, C. G. Lim, and H. J. Choi, “Personalized news recommendation using classified keywords to capture user preference,” *Int. Conf. Adv. Commun. Technol. ICACT*, pp. 1283–1287, 2014, doi: 10.1109/ICACT.2014.6779166.

[30] Y. Song, A. M. Elkahky, and X. He, “Multi-rate deep learning for temporal recommendation,” *SIGIR 2016 - Proc. 39th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 909–912, 2016, doi: 10.1145/2911451.2914726.

[31] V. Kumar, D. Khattar, S. Gupta, M. Gupta, and V. Varma, “User profiling based deep neural network for temporal news recommendation,” *IEEE Int. Conf. Data Min. Work. ICDMW*, vol. 2017-Novem, pp. 765–772, 2017, doi: 10.1109/ICDMW.2017.106.

[32] S. Cao, N. Yang, and Z. Liu, “Online news recommender based on stacked auto-encoder,” *Proc. - 16th IEEE/ACIS Int. Conf. Comput. Inf. Sci. ICIS 2017*, pp. 721–726, 2017, doi:

10.1109/ICIS.2017.7960088.

[33] K. Park, J. Lee, and J. Choi, “Deep Neural Networks for News Recommendations,” *Proc. 2017 ACM Conf. Inf. Knowl. Manag. - CIKM ’17*, pp. 2255–2258, 2017, doi: 10.1145/3132847.3133154.

[34] G. Shani *et al.*, “DRN: A Deep Reinforcement Learning Framework for News Recommendation,” *Proc. 2018 World Wide Web Conf. World Wide Web*, vol. 6, no. Sep, pp. 113–120, 2018, doi: 10.1145/3178876.3185994.

[35] B. Yi, X. Shen, Z. Zhang, J. Shu, and H. Liu, “Expanded autoencoder recommendation framework and its application in movie recommendation,” *Ski. 2016 - 2016 10th Int. Conf. Software, Knowledge, Inf. Manag. Appl.*, pp. 298–303, 2017, doi: 10.1109/SKIMA.2016.7916236.

[36] J. B. P. Vuurens, M. Larson, and A. P. De Vries, “Exploring deep space: Learning personalized ranking in a semantic space,” *ACM Int. Conf. Proceeding Ser.*, vol. 15-Septemb, pp. 23–28, 2016, doi: 10.1145/2988450.2988457.

[37] C. Zhao, J. Shi, T. Jiang, J. Zhao, and J. Chen, “Application of deep belief nets for collaborative filtering,” *2016 16th Int. Symp. Commun. Inf. Technol. Isc. 2016*, pp. 201–205, 2016, doi: 10.1109/ISCIT.2016.7751621.

[38] G. Sottocornola, F. Stella, M. Zanker, and F. Canonaco, “Towards a deep learning model for hybrid recommendation,” *Proc. Int. Conf. Web Intell. - WI ’17*, pp. 1260–1264, 2017, doi: 10.1145/3106426.3110321.

[39] S. M. Taheri and I. Irajian, “DeepMovRS: A unified framework for deep learning-based movie recommender systems,” *2018 6th Iran. Jt. Congr. Fuzzy Intell. Syst.*, pp. 200–204, 2018, doi: 10.1109/CFIS.2018.8336633.

[40] M. Fu, H. Qu, Z. Yi, L. Lu, and Y. Liu, “A Novel Deep Learning-Based Collaborative Filtering Model for Recommendation System,” *IEEE Trans. Cybern.*, pp. 1–13, 2018, doi: 10.1109/TCYB.2018.2795041.

[41] J. Yuan, W. Shalaby, M. Korayem, D. Lin, K. AlJadda, and J. Luo, “Solving Cold-Start Problem in Large-scale Recommendation Engines: {A} Deep Learning Approach,” *CoRR*, vol. abs/1611.0, pp. 1901–1910, 2016.

[42] W. Chen, X. Zhang, H. Wang, and H. Xu, “Hybrid deep collaborative filtering for job recommendation,” *2017 2nd IEEE Int. Conf. Comput. Intell. Appl. ICCIA 2017*, vol. 2017-

Janua, pp. 275–280, 2017, doi: 10.1109/CIAPP.2017.8167222.

[43] T. T. Nguyen and H. W. Lauw, “Collaborative Topic Regression with Denoising AutoEncoder for Content and Community Co-Representation,” *Proc. 2017 ACM Conf. Inf. Knowl. Manag. - CIKM ’17*, pp. 2231–2234, 2017, doi: 10.1145/3132847.3133128.

[44] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, “Collaborative filtering and deep learning based recommendation system for cold start items,” *Expert Syst. Appl.*, vol. 69, pp. 1339–1351, 2017, doi: 10.1016/j.eswa.2016.09.040.

[45] Y. Zhang, H. Yin, Z. Huang, X. Du, G. Yang, and D. Lian, “Discrete Deep Learning for Fast Content-Aware Recommendation,” *Proc. Elev. ACM Int. Conf. Web Search Data Min. - WSDM ’18*, pp. 717–726, 2018, doi: 10.1145/3159652.3159688.

[46] Y. K. Tan, X. Xu, and Y. Liu, “Improved Recurrent Neural Networks for Session-based Recommendations,” pp. 0–5, 2016, doi: 10.1145/2988450.2988452.

[47] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk, “Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations,” *Proc. 10th ACM Conf. Recomm. Syst. - RecSys ’16*, pp. 241–248, 2016, doi: 10.1145/2959100.2959167.

[48] A. Greenstein-Messica, L. Rokach, and M. Friedman, “Session-Based Recommendations Using Item Embedding,” *Proc. 22nd Int. Conf. Intell. User Interfaces - IUI ’17*, pp. 629–633, 2017, doi: 10.1145/3025171.3025197.

[49] S. P. Chatzis, P. Christodoulou, and A. S. Andreou, “Recurrent Latent Variable Networks for Session-Based Recommendation,” *Proc. 2nd Work. Deep Learn. Recomm. Syst. - DLRS 2017*, no. Dl, pp. 38–45, 2017, doi: 10.1145/3125486.3125493.

[50] M. Ruocco, O. S. L. Skrede, and H. Langseth, “Inter-Session Modeling for Session-Based Recommendation,” 2017, doi: 10.1145/3125486.3125491.

[51] X. Wang and Y. Wang, “Improving Content-based and Hybrid Music Recommendation using Deep Learning,” *Proc. ACM Int. Conf. Multimed. - MM ’14*, pp. 627–636, 2014, doi: 10.1145/2647868.2654940.

[52] P. Chiliguano and G. Fazekas, “Hybrid music recommender using content-based and social information,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2016-May, pp. 2618–2622, 2016, doi: 10.1109/ICASSP.2016.7472151.

[53] S. Oramas, O. Nieto, M. Sordo, and X. Serra, “A Deep Multimodal Approach for Cold-

start Music Recommendation," *Proc. 2nd Work. Deep Learn. Recomm. Syst. - DLRS 2017*, pp. 32–37, 2017, doi: 10.1145/3125486.3125492.

[54] M. Jiang, Z. Yang, and C. Zhao, "What to play next? A RNN-based music recommendation system," *Conf. Rec. 51st Asilomar Conf. Signals, Syst. Comput. ACSSC 2017*, vol. 2017-Octob, pp. 356–358, 2018, doi: 10.1109/ACSSC.2017.8335200.

[55] O. U. Florez, "Deep Learning of Semantic Word Representations to Implement a Content-Based Recommender for the RecSys Challenge'14," 2014, pp. 199–204.

[56] D. Kim, C. Park, J. Oh, and H. Yu, "Deep hybrid recommender systems via exploiting document context and statistics of items," *Inf. Sci. (Ny).*, vol. 417, pp. 72–87, 2017, doi: 10.1016/j.ins.2017.06.026.

[57] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional Matrix Factorization for Document Context-Aware Recommendation," *Proc. 10th ACM Conf. Recomm. Syst. - RecSys '16*, pp. 233–240, 2016, doi: 10.1145/2959100.2959165.

[58] T. Bansal, D. Belanger, and A. McCallum, "*Ask the GRU*," *Proc. 10th ACM Conf. Recomm. Syst. - RecSys '16*, pp. 107–114, 2016, doi: 10.1145/2959100.2959180.

[59] X. Wang *et al.*, "Dynamic Attention Deep Model for Article Recommendation by Learning Human Editors' Demonstration," *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. - KDD '17*, pp. 2051–2059, 2017, doi: 10.1145/3097983.3098096.

[60] C. Lei, D. Liu, W. Li, Z.-J. Zha, and H. Li, "Comparative Deep Learning of Hybrid Representations for Image Recommendations," pp. 2545–2553, 2016, doi: 10.1109/CVPR.2016.279.

[61] J. Wang and K. Kawagoe, "Ukiyo-e Recommendation based on Deep Learning For Learning Japanese Art and Culture," *Proc. 2017 Int. Conf. Inf. Syst. Data Min. - ICISDM '17*, pp. 119–123, 2017, doi: 10.1145/3077584.3077612.

[62] L. Peska and H. Trojanova, "Towards Recommender Systems for Police Photo Lineup," *Proc. 2nd Work. Deep Learn. Recomm. Syst. - DLRS 2017*, pp. 19–23, 2017, doi: 10.1145/3125486.3125490.

[63] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, "On Deep Learning for Trust-Aware Recommendations in Social Networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 5, pp. 1164–1177, 2017, doi: 10.1109/TNNLS.2016.2514368.

[64] Q. V. Dang and C. L. Ignat, "DTrust: A Simple Deep Learning Approach for Social

Recommendation," *Proc. - 2017 IEEE 3rd Int. Conf. Collab. Internet Comput. CIC 2017*, vol. 2017-Janua, pp. 209–218, 2017, doi: 10.1109/CIC.2017.00036.

[65] D. Rafailidis and F. Crestani, "Recommendation with Social Relationships via Deep Learning," *Proc. ACM SIGIR Int. Conf. Theory Inf. Retr. - ICTIR '17*, pp. 151–158, 2017, doi: 10.1145/3121050.3121057.

[66] A. Tomar, F. Godin, B. Vandersmissen, W. De Neve, and R. Van De Walle, "Towards Twitter hashtag recommendation using distributed word representations and a deep feed forward neural network," *Proc. 2014 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2014*, pp. 362–368, 2014, doi: 10.1109/ICACCI.2014.6968557.

[67] Y. Zuo, J. Zeng, M. Gong, and L. Jiao, "Tag-aware recommender systems based on deep neural networks," *Neurocomputing*, vol. 204, pp. 51–60, 2016, doi: 10.1016/j.neucom.2015.10.134.

[68] Z. Xu, C. Chen, T. Lukasiewicz, Y. Miao, and X. Meng, "Tag-Aware Personalized Recommendation Using a Deep-Semantic Similarity Model with Negative Sampling," *Proc. 25th ACM Int. Conf. Inf. Knowl. Manag. - CIKM '16*, pp. 1921–1924, 2016, doi: 10.1145/2983323.2983874.

[69] F. Wang, Y. Qu, L. Zheng, C. T. Lu, and P. S. Yu, "Deep and Broad Learning on Content-Aware POI Recommendation," *Proc. - 2017 IEEE 3rd Int. Conf. Collab. Internet Comput. CIC 2017*, vol. 2017-Janua, pp. 369–378, 2017, doi: 10.1109/CIC.2017.00054.

[70] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-Aware Hierarchical Collaborative Deep Learning for POI Recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 11, pp. 2537–2551, 2017, doi: 10.1109/TKDE.2017.2741484.

[71] B. Xia, Y. Li, Q. Li, and T. Li, "Attention-based recurrent neural network for location recommendation," *Proc. 2017 12th Int. Conf. Intell. Syst. Knowl. Eng. ISKE 2017*, vol. 2018-Janua, pp. 1–6, 2018, doi: 10.1109/ISKE.2017.8258747.

[72] V. Huck-Fries, F. Wiegand, K. Klinker, M. Wiesche, and H. Krcmar, "Reranking-based Recommender System with Deep Learning," *Inform. 2017*, pp. 585–596, 2017, doi: 10.18420/in2017.

[73] H. A. M. Hassan, "Personalized Research Paper Recommendation using Deep Learning," *Proc. 25th Conf. User Model. Adapt. Pers. - UMAP '17*, pp. 327–330, 2017, doi: 10.1145/3079628.3079708.

[74] E. Smirnova and F. Vasile, “Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks,” 2017, doi: 10.1145/3125486.3125488.

[75] M. Verma and D. Ganguly, “LiRME: Locally interpretable ranking model explanation,” *SIGIR 2019 - Proc. 42nd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 1281–1284, 2019, doi: 10.1145/nnnnnnn.nnnnnnn.

[76] C. Hongliang and Q. Xiaona, “The video recommendation system based on DBN,” *Proc. - 15th IEEE Int. Conf. Comput. Inf. Technol. CIT 2015, 14th IEEE Int. Conf. Ubiquitous Comput. Commun. IUCC 2015, 13th IEEE Int. Conf. Dependable, Auton. Se*, pp. 1016–1021, 2015, doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.154.

[77] P. Covington, J. Adams, and E. Sargin, “Deep Neural Networks for YouTube Recommendations,” *ACM Conf. Recomm. Syst.*, pp. 191–198, 2016, doi: 10.1145/2959100.2959190.

[78] H. Lee, Y. Ahn, H. Lee, S. Ha, and S. Lee, “Quote Recommendation in Dialogue using Deep Neural Network,” *Proc. 39th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR ’16*, pp. 957–960, 2016, doi: 10.1145/2911451.2914734.

[79] J. Tan, X. Wan, and J. Xiao, “A Neural Network Approach to Quote Recommendation in Writings,” *Proc. 25th ACM Int. Conf. Inf. Knowl. Manag. - CIKM ’16*, pp. 65–74, 2016, doi: 10.1145/2983323.2983788.

[80] H. Zhang, H. Yang, T. Huang, and G. Zhan, “DBNCF: Personalized courses recommendation system based on DBN in MOOC environment,” *Proc. - 2017 Int. Symp. Educ. Technol. ISET 2017*, pp. 106–108, 2017, doi: 10.1109/ISET.2017.33.

[81] X. Wang, Y. Zhang, S. Yu, X. Liu, Y. Yuan, and F. Y. Wang, “E-learning recommendation framework based on deep learning,” *2017 IEEE Int. Conf. Syst. Man, Cybern. SMC 2017*, vol. 2017-Janua, pp. 455–460, 2017, doi: 10.1109/SMC.2017.8122647.

[82] P. Li, Z. Wang, Z. Ren, L. Bing, and W. Lam, “Neural Rating Regression with Abstractive Tips Generation for Recommendation,” pp. 345–354, 2017, doi: 10.1145/3077136.3080822.

[83] S. Maheshwary and H. Misra, “Matching Resumes to Jobs via Deep Siamese Network,” *Companion Proc. Web Conf. 2018*, pp. 87–88, 2018, doi: 10.1145/3184558.3186942.

[84] S. Jaradat, “Deep Cross-Domain Fashion Recommendation,” *Proc. Elev. ACM Conf.*

Recomm. Syst. - RecSys '17, pp. 407–410, 2017, doi: 10.1145/3109859.3109861.

[85] S. Jiang, Y. Wu, and Y. Fu, "5 Deep Bidirectional Cross-Triplet Embedding for Online Clothing Shopping," *ACM Trans. Multimed. Comput. Commun. Appl. Artic.*, vol. 14, no. 22, pp. 1–22, 2018, doi: 10.1145/3152114.

[86] G. H. Sack, "Human cell transformation by simian virus 40—A review," *In Vitro*, vol. 17, no. 1, pp. 1–19, Jan. 1981, doi: 10.1007/BF02618025.

[87] J. Zahalka, S. Rudinac, and M. Worring, "Interactive multimodal learning for venue recommendation," *IEEE Trans. Multimed.*, vol. 17, no. 12, pp. 2235–2244, 2015, doi: 10.1109/TMM.2015.2480007.

[88] T. Gao, X. Li, Y. Chai, and Y. Tang, "Deep learning with consumer preferences for recommender system," *2016 IEEE Int. Conf. Inf. Autom. IEEE ICIA 2016*, no. August, pp. 1556–1561, 2017, doi: 10.1109/ICInfA.2016.7832066.

[89] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Recurrent coevolutionary latent feature processes for continuous-time recommendation," *ACM Int. Conf. Proceeding Ser.*, vol. 15-Septemb, pp. 29–34, 2016, doi: 10.1145/2988450.2988451.

[90] V. Dominguez, P. Messina, D. Parra, D. Mery, C. Trattner, and A. Soto, "Comparing Neural and Attractiveness-based Visual Features for Artwork Recommendation," *Proc. 2nd Work. Deep Learn. Recomm. Syst. - DLRS 2017*, pp. 55–59, 2017, doi: 10.1145/3125486.3125495.

[91] H. Soh, S. Sanner, M. White, and G. Jamieson, "Deep Sequential Recommendation for Personalized Adaptive User Interfaces," *Proc. 22nd Int. Conf. Intell. User Interfaces - IUI '17*, pp. 589–593, 2017, doi: 10.1145/3025171.3025207.

[92] S. T. Jishan and Y. Wang, "Audience Activity Recommendation Using Stacked-LSTM Based Sequence Learning," *Proc. 9th Int. Conf. Mach. Learn. Comput. - ICMLC 2017*, pp. 98–106, 2017, doi: 10.1145/3055635.3056606.

[93] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, and L. Sun, "Dual-regularized matrix factorization with deep neural networks for recommender systems," *Knowledge-Based Syst.*, vol. 145, pp. 1–14, 2018, doi: 10.1016/j.knosys.2018.01.003.

[94] W. Yuan, C. Li, D. Guan, G. Han, and A. M. Khattak, "Socialized healthcare service recommendation using deep learning," *Neural Comput. Appl.*, vol. 30, no. 7, pp. 2071–2082, 2018, doi: 10.1007/s00521-018-3394-4.

[95] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger, “DeepSurv: Personalized Treatment Recommender System Using A Cox Proportional Hazards Deep Neural Network,” pp. 1–12, 2018, doi: 10.1186/s12874-018-0482-1.

[96] N. Nassar, A. Jafar, and Y. Rahhal, “A novel deep multi-criteria collaborative filtering model for recommendation system,” *Knowledge-Based Syst.*, vol. 187, pp. 1–13, 2020, doi: 10.1016/j.knosys.2019.06.019.

[97] H. Wang and D. Yeung, “Towards Bayesian Deep Learning: A Framework and Some Existing Methods,” *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3395–3408, Dec. 2016, doi: 10.1109/TKDE.2016.2606428.

[98] T. Bai, J.-R. Wen, J. Zhang, and W. X. Zhao, “A Neural Collaborative Filtering Model with Interaction-based Neighborhood,” *Proc. 2017 ACM Conf. Inf. Knowl. Manag. - CIKM ’17*, pp. 1979–1982, 2017, doi: 10.1145/3132847.3133083.

[99] W. Zhang, F. Liu, L. Jiang, and D. Xu, “Recommendation based on collaborative filtering by convolution deep learning model based on label weight nearest neighbor,” *Proc. - 2017 10th Int. Symp. Comput. Intell. Des. Isc. 2017*, vol. 2, pp. 504–507, 2018, doi: 10.1109/ISCID.2017.235.

[100] J. Liu and D. Wang, “PHD : A Probabilistic Model of Hybrid Deep Collaborative Filtering for Recommender Systems,” *Acml*, pp. 1–16, 2017.

[101] Y. Tay, A. T. Luu, and S. C. Hui, “Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking,” pp. 729–739, 2017, doi: 10.1145/3178876.3186154.

[102] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, “A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems,” *Proc. 31st AAAI Conf. Artif. Intell.*, pp. 1309–1315, 2017, doi: 10.1103/PhysRevLett.93.077207.

[103] H. J. Xue, X. Y. Dai, J. Zhang, S. Huang, and J. Chen, “Deep matrix factorization models for recommender systems,” *IJCAI Int. Jt. Conf. Artif. Intell.*, pp. 3203–3209, 2017, doi: 10.24963/ijcai.2017/447.

[104] R. Catherine and W. Cohen, “TransNets: Learning to Transform for Recommendation,” pp. 288–296, 2017, doi: 10.1145/3109859.3109878.

[105] G. Preethi, P. V. Krishna, M. S. Obaidat, V. Saritha, and S. Yenduri, “Application of Deep Learning to Sentiment Analysis for recommender system on cloud,” *IEEE CITS 2017 - 2017 Int. Conf. Comput. Inf. Telecommun. Syst.*, pp. 93–97, 2017, doi:

10.1109/CITS.2017.8035341.

[106] J. Serrà and A. Karatzoglou, “Getting deep recommenders fit: Bloom embeddings for sparse binary input/output networks,” pp. 279–287, 2017, doi: 10.1145/3109859.3109876.

[107] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, “Collaborative Filtering and Deep Learning Based Hybrid Recommendation for Cold Start Problem,” *Proc. - 2016 IEEE 14th Int. Conf. Dependable, Auton. Secur. Comput. DASC 2016, 2016 IEEE 14th Int. Conf. Pervasive Intell. Comput. PICom 2016, 2016 IEEE 2nd Int. Conf. Big Data*, pp. 874–877, 2016, doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2016.149.

[108] H. Wang, N. Wang, and D.-Y. Yeung, “Collaborative Deep Learning for Recommender Systems,” pp. 1235–1244, 2014, doi: 10.1145/2783258.2783273.

[109] H. Wang and D.-Y. Yeung, “Towards Bayesian Deep Learning: A Survey,” vol. 28, no. 12, pp. 3395–3408, 2016, doi: 10.1109/TKDE.2016.2606428.

[110] W. Lee, K. Song, and I.-C. Moon, “Augmented Variational Autoencoders for Collaborative Filtering with Auxiliary Information,” *Proc. 2017 ACM Conf. Inf. Knowl. Manag. - CIKM ’17*, pp. 1139–1148, 2017, doi: 10.1145/3132847.3132972.

[111] Q. Li, X. Zheng, and X. Wu, “Collaborative Autoencoder for Recommender Systems,” pp. 305–314, 2017, doi: 10.1145/3097983.3098077.

[112] H.-T. Cheng *et al.*, “Wide & Deep Learning for Recommender Systems,” 2016, doi: 10.1145/2988450.2988454.

[113] X. He and T.-S. Chua, “Neural Factorization Machines for Sparse Predictive Analytics,” pp. 355–364, 2017, doi: 10.1145/3077136.3080777.

[114] G. Zheng *et al.*, “DRN: A deep reinforcement learning framework for news recommendation,” *Web Conf. 2018 - Proc. World Wide Web Conf. WWW 2018*, vol. 2, pp. 167–176, 2018, doi: 10.1145/3178876.3185994.

[115] T. Ebesu and Y. Fang, “Neural Citation Network for Context-Aware Citation Recommendation,” *Proc. 40th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR ’17*, pp. 1093–1096, 2017, doi: 10.1145/3077136.3080730.

[116] C. Chen, P. Zhao, L. Li, J. Zhou, X. Li, and M. Qiu, “Locally Connected Deep Learning Framework for Industrial-scale Recommender Systems,” *Proc. 26th Int. Conf. World Wide Web Companion - WWW ’17 Companion*, pp. 769–770, 2017, doi: 10.1145/3041021.3054227.

[117] A. Da'u, N. Salim, I. Rabiu, and A. Osman, "Recommendation system exploiting aspect-based opinion mining with deep learning method," *Inf. Sci. (Ny).*, vol. 512, pp. 1279–1292, 2020, doi: 10.1016/j.ins.2019.10.038.

[118] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–35, 2019, doi: 10.1145/3285029.

[119] J. Bobadilla, S. Alonso, and A. Hernando, "Deep learning architecture for collaborative filtering recommender systems," *Appl. Sci.*, vol. 10, no. 7, 2020, doi: 10.3390/app10072441.

[120] H. Zarzour, Z. A. Al-Sharif, and Y. Jararweh, "RecDNNing: A recommender system using deep neural network with user and item embeddings," *2019 10th Int. Conf. Inf. Commun. Syst. ICICS 2019*, pp. 99–103, 2019, doi: 10.1109/IACS.2019.8809156.

[121] F. Fessahaye *et al.*, "T-RECSYS: A Novel Music Recommendation System Using Deep Learning," *2019 IEEE Int. Conf. Consum. Electron. ICCE 2019*, 2019, doi: 10.1109/ICCE.2019.8662028.

[122] H. Lee and J. Lee, "Scalable deep learning-based recommendation systems," *ICT Express*, vol. 5, no. 2, pp. 84–88, 2019, doi: 10.1016/j.icte.2018.05.003.

[123] P. Nimirthi, P. Venkata Krishna, M. S. Obaidat, and V. Saritha, "A framework for sentiment analysis based recommender system for agriculture using deep learning approach," *SpringerBriefs Appl. Sci. Technol.*, pp. 59–66, 2019, doi: 10.1007/978-981-13-1456-8_5.

[124] H. F. Nweke, Y. W. Teh, M. A. Al-garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Syst. Appl.*, vol. 105, pp. 233–261, 2018, doi: 10.1016/j.eswa.2018.03.056.

[125] L. Zheng, "A Survey and Critique of Deep Learning on Recommender Systems," no. September, p. 31, 2016, doi: 10.1145.

[126] A. Da'u, N. Salim, I. Rabiu, and A. Osman, "Weighted aspect-based opinion mining using deep learning for recommender system," *Expert Syst. Appl.*, vol. 140, 2020, doi: 10.1016/j.eswa.2019.112871.

[127] S. Wang, C. Huang, J. Li, Y. Yuan, and F. Y. Wang, "Decentralized construction of

knowledge graphs for deep recommender systems based on blockchain-powered smart contracts," *IEEE Access*, vol. 7, pp. 136951–136961, 2019, doi: 10.1109/ACCESS.2019.2942338.

[128] Z. Huang, J. Tang, G. Shan, J. Ni, Y. Chen, and C. Wang, "An Efficient Passenger-Hunting Recommendation Framework with Multitask Deep Learning," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7713–7721, 2019, doi: 10.1109/JIOT.2019.2901759.

[129] P. Vincent and H. Larochelle, "Extracting and Composing Robust Features with Denoising.pdf," pp. 1096–1103, 2008.

[130] X. Zhang, J. Zhong, and K. Liu, "Wasserstein autoencoders for collaborative filtering," *Neural Comput. Appl.*, 2020, doi: 10.1007/s00521-020-05117-w.

[131] X. Deng and F. Huangfu, "Collaborative Variational Deep Learning for Healthcare Recommendation," *IEEE Access*, vol. 7, pp. 55679–55688, 2019, doi: 10.1109/ACCESS.2019.2913468.

[132] Y. Pan, F. He, and H. Yu, "Learning social representations with deep autoencoder for recommender system," *World Wide Web*, vol. 23, no. 4, pp. 2259–2279, 2020, doi: 10.1007/s11280-020-00793-z.

[133] B. Saravanan, V. Mohanraj, and J. Senthilkumar, "A fuzzy entropy technique for dimensionality reduction in recommender systems using deep learning," *Soft Comput.*, vol. 23, no. 8, pp. 2575–2583, 2019, doi: 10.1007/s00500-019-03807-9.

[134] Y. Guan, Q. Wei, and G. Chen, "Deep learning based personalized recommendation with multi-view information integration," *Decis. Support Syst.*, vol. 118, no. August 2018, pp. 58–69, 2019, doi: 10.1016/j.dss.2019.01.003.

[135] K. Wang, L. Xu, L. Huang, C. D. Wang, and J. H. Lai, "SDDRS: Stacked Discriminative Denoising Auto-Encoder based Recommender System," *Cogn. Syst. Res.*, vol. 55, pp. 164–174, 2019, doi: 10.1016/j.cogsys.2019.01.011.

[136] R. Damaševičius and L. Zailskaitė-Jakštė, "Usability and Security Testing of Online Links: A Framework for Click-Through Rate Prediction Using Deep Learning," *Electron.*, vol. 11, no. 3, 2022, doi: 10.3390/electronics11030400.

[137] Y. Meng, C. Lu, M. Jin, J. Xu, X. Zeng, and J. Yang, "A weighted bilinear neural collaborative filtering approach for drug repositioning," *Brief. Bioinform.*, vol. 23, no. 2, 2022, doi: 10.1093/bib/bbab581.

[138] Y. Zhang, C. Yin, Q. Wu, Q. He, and H. Zhu, “Location-Aware Deep Collaborative Filtering for Service Recommendation,” *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. PP, pp. 1–12, 2019, doi: 10.1109/tsmc.2019.2931723.

[139] M. T. Ahamed and S. Afroge, “A Recommender System Based on Deep Neural Network and Matrix Factorization for Collaborative Filtering,” *2nd Int. Conf. Electr. Comput. Commun. Eng. ECCE 2019*, pp. 1–5, 2019, doi: 10.1109/ECACE.2019.8679125.

[140] R. Feinman, “A Deep Belief Network Approach to Learning Depth From Optical Flow,” pp. 1–14.

[141] A. G. C. Pacheco, R. A. Krohling, and C. A. S. da Silva, “Restricted Boltzmann machine to determine the input weights for extreme learning machines,” *Expert Syst. Appl.*, vol. 96, pp. 77–85, 2018, doi: 10.1016/j.eswa.2017.11.054.

[142] J. Schmidhuber, “Deep Learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.

[143] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” *Neural Comput.*, vol. 18, pp. 1527–1554, 2006, doi: 10.1162/neco.2006.18.7.1527.

[144] L. Luo, S. Zhang, Y. Wang, and H. Peng, “An alternate method between generative objective and discriminative objective in training classification Restricted Boltzmann Machine,” *Knowledge-Based Syst.*, vol. 144, pp. 144–152, 2018, doi: 10.1016/j.knosys.2017.12.032.

[145] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, no. October 2016, pp. 11–26, 2017, doi: 10.1016/j.neucom.2016.12.038.

[146] B. Hidasi and A. Karatzoglou, “Recurrent Neural Networks with Top-k Gains for Session-based Recommendations,” pp. 370–371, 2017, doi: 10.1145/3269206.3271761.

[147] A. Da’U and N. Salim, “Sentiment-Aware Deep Recommender System with Neural Attention Networks,” *IEEE Access*, vol. 7, pp. 45472–45484, 2019, doi: 10.1109/ACCESS.2019.2907729.

[148] K. Tejas D, N. Karthik R., S. Ardavan, and T. Joshua B., “Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation,” *Conf. Neural Inf. Process. Syst.*, 2016, doi: 10.1162/NECO.

[149] Y. Song, J. Wang, T. Lukasiewicz, Z. Xu, and M. Xu, “Diversity-Driven Extensible

Hierarchical Reinforcement Learning," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 4992–4999, 2019, doi: 10.1609/aaai.v33i01.33014992.

[150] J. Zhang, B. Hao, B. Chen, C. Li, H. Chen, and J. Sun, "Hierarchical Reinforcement Learning for Course Recommendation in MOOCs," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 435–442, 2019, doi: 10.1609/aaai.v33i01.3301435.

[151] X. He, Z. He, J. Song, Z. Liu, Y. G. Jiang, and T. S. Chua, "NAIS: Neural Attentive Item Similarity Model for Recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2354–2366, 2018, doi: 10.1109/TKDE.2018.2831682.

[152] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," *Int. Conf. Inf. Knowl. Manag. Proc.*, vol. Part F1318, pp. 1419–1428, 2017, doi: 10.1145/3132847.3132926.

[153] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous Deep Q-Learning with Model-based Acceleration," *Int. Conf. Mach. Learn. New York*, vol. 48, 2016.

[154] M. Hessel *et al.*, "Rainbow: Combining Improvements in DQN," *Thirty-Second AAAI Conf. Artif. Intell.*, pp. 3215–3222, 2018.

[155] H. Van Hasselt, "Double Q-learning," *Adv. Neural Inf. Process. Syst. 23 24th Annu. Conf. Neural Inf. Process. Syst. 2010, NIPS 2010*, pp. 1–9, 2010.

[156] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*, pp. 1–21, 2016.

[157] R. S. Sutton, "Learning to Predict by the Methods of Temporal Differences," *Mach. Learn. Proc. 1991*, pp. 9–44, 1988, doi: https://doi.org/10.1007/BF00115009.

[158] M. G. Bellemare, W. Dabney, and R. Munos, "A Distributional Perspective on Reinforcement Learning," *arXiv*, Jul. 2017.

[159] M. Fortunato *et al.*, "Noisy networks for exploration," *arXiv*, pp. 1–21, 2017.

[160] C. Chen *et al.*, "Reinforcement learning for user intent prediction in customer service bots," *SIGIR 2019 - Proc. 42nd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 1265–1268, 2019, doi: 10.1145/3331184.3331370.

[161] D. Liu and C. Yang, "A Deep Reinforcement Learning Approach to Proactive Content Pushing and Recommendation for Mobile Users," *IEEE Access*, vol. 7, pp. 83120–83136, 2019, doi: 10.1109/ACCESS.2019.2925019.

[162] M. Lim, A. Abdullah, and N. Jhanjhi, "Performance optimization of criminal network

hidden link prediction model with deep reinforcement learning," *J. King Saud Univ. - Comput. Inf. Sci.*, no. xxxx, 2019, doi: 10.1016/j.jksuci.2019.07.010.

[163] Z. Liu, C. Yao, H. Yu, and T. Wu, "Deep reinforcement learning with its application for lung cancer detection in medical Internet of Things," *Futur. Gener. Comput. Syst.*, vol. 97, pp. 1–9, 2019, doi: 10.1016/j.future.2019.02.068.

[164] Y. Lei, Z. Wang, W. Li, and H. Pei, "Social attentive deep Q-network for recommendation," *SIGIR 2019 - Proc. 42nd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 1189–1192, 2019, doi: 10.1145/3331184.3331302.

[165] T. H. Dodd, "Journal of Restaurant & Foodservice Marketing Techniques to Increase Impulse Wine Purchases in a Restaurant Setting," no. April 2015, pp. 37–41, doi: 10.1300/J061v02n01.

[166] S. Pettigrew and S. Charters, "Consumers' expectations of food and alcohol pairing," *Br. Food J.*, vol. 108, no. 3, pp. 169–180, 2006, doi: 10.1108/00070700610650990.

[167] E. Ginon, G. Ares, S. Issanchou, L. H. E. dos S. Laboissière, and R. Deliza, "Identifying motives underlying wine purchase decisions: Results from an exploratory free listing task with Burgundy wine consumers," *Food Res. Int.*, vol. 62, pp. 860–867, 2014, doi: 10.1016/j.foodres.2014.04.052.

[168] L. Sirieix, H. Remaud, L. Lockshin, L. Thach, and T. Lease, "Determinants of restaurant's owners/managers selection of wines to be offered on the wine list," *J. Retail. Consum. Serv.*, vol. 18, no. 6, pp. 500–508, 2011, doi: 10.1016/j.jretconser.2011.06.012.

[169] R. J. Harrington, *Food & wine pairing: a sensory experience*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2008.

[170] A. Eschevins, A. Giboreau, P. Julien, and C. Dacremont, "From expert knowledge and sensory science to a general model of food and beverage pairing with wine and beer," *Int. J. Gastron. Food Sci.*, vol. 17, no. June 2018, p. 100144, 2019, doi: 10.1016/j.ijgfs.2019.100144.

[171] R. Gawel, A. Oberholster, and I. L. Francis, "A 'Mouth-feel Wheel': Terminology for communicating the mouth-feel characteristics of red wine," *Aust. J. Grape Wine Res.*, vol. 6, no. 3, pp. 203–207, 2000, doi: 10.1111/j.1755-0238.2000.tb00180.x.

[172] R. Koone, R. J. Harrington, M. Gozzi, and M. McCarthy, "The role of acidity, sweetness, tannin and consumer knowledge on wine and food match perceptions," *J. Wine Res.*, vol.

25, no. 3, pp. 158–174, 2014, doi: 10.1080/09571264.2014.899491.

[173] L. Briand and C. Salles, “Taste perception and integration,” *Flavor From Food to Behav. Wellbeing Heal.*, no. February 2017, pp. 101–119, 2016, doi: 10.1016/B978-0-08-100295-7.00004-9.

[174] R. J. Harrington and R. Hammond, “The impact of wine effervescence levels on perceived palatability with salty and bitter foods,” *J. Foodserv. Bus. Res.*, vol. 12, no. 3, pp. 234–246, 2009, doi: 10.1080/15378020903158509.

[175] M. Goel and G. Bagler, “Computational gastronomy: A data science approach to food,” *J. Biosci.*, vol. 47, no. 1, pp. 1–10, 2022, doi: 10.1007/s12038-021-00248-1.

[176] R. J. Harrington and H. S. Seo, “The Impact of Liking of Wine and Food Items on Perceptions of Wine–Food Pairing,” *J. Foodserv. Bus. Res.*, vol. 18, no. 5, pp. 489–501, 2015, doi: 10.1080/15378020.2015.1093455.

[177] J. Simon, *Wine with Food*. New York: Simon & Schuster, 1997.

[178] G. Wolf, “Detecting wine taste using Recommender Systems,” 2021.

[179] Y. Y. Ahn, S. E. Ahnert, J. P. Bagrow, and A. L. Barabási, “Flavor network and the principles of food pairing,” *Sci. Rep.*, vol. 1, pp. 1–7, 2011, doi: 10.1038/srep00196.

[180] R. J. Harrington, M. McCarthy, and M. Gozzi, “Perceived match of wine and cheese and the impact of additional food elements: A preliminary study,” *J. Foodserv. Bus. Res.*, vol. 13, no. 4, pp. 311–330, 2010, doi: 10.1080/15378020.2010.524541.

[181] R. J. Harrington, “Defining Gastronomy Identity,” *J. Culin. Sci. Technol.*, vol. 4, no. 2/3, pp. 129–152, 2008, doi: 10.1300/J385v04n02.

[182] R. J. Harrington, “The Wine and Food Pairing Process: Using Culinary and Sensory Perspectives,” *J. Culin. Sci. Technol.*, vol. 4, no. October 2014, pp. 101–112, 2005, doi: 10.1300/J385v04n01.

[183] J. Ye, “Cosine similarity measures for intuitionistic fuzzy sets and their applications,” *Math. Comput. Model.*, vol. 53, no. 1–2, pp. 91–97, 2011, doi: 10.1016/j.mcm.2010.07.022.

[184] I. Borg and P. Groenen, “Scalar Products and Euclidean Distances,” in *Modern multidimensional scaling*, no. 18, 1997, pp. 301–319.

[185] G. Gupta and R. Katarya, “Research on Understanding the Effect of Deep Learning on User Preferences,” *Arab. J. Sci. Eng.*, 2020, doi: 10.1007/s13369-020-05112-2.

[186] B. Zoph and Q. V. Le, “Neural Architecture Search with Reinforcement Learning,” *Cvpr2019*, no. Section 2, pp. 1–11, Nov. 2016.

[187] C. Lili, “Recommender algorithms based on boosting ensemble learning,” *Int. J. Smart Sens. Intell. Syst.*, vol. 8, no. 1, pp. 368–386, 2015, doi: 10.21307/ijssis-2017-763.

[188] K. Yu, A. Schwaighofer, V. Tresp, W. Ma, and H. Zhang, “Collaborative ensemble learning: Combining collaborative and content-based information filtering via hierarchical bayes,” *UAI’03 Proc. Ninet. Conf. Uncertain. Artif. Intell.*, pp. 353–360, 2003, doi: 10.1.1.9.7268.

[189] A. Schclar, A. Tsikinovsky, L. Rokach, A. Meisels, and L. Antwarg, “Ensemble methods for improving the performance of neighborhood-based collaborative filtering,” *RecSys’09 - Proc. 3rd ACM Conf. Recomm. Syst.*, pp. 261–264, 2009, doi: 10.1145/1639714.1639763.

[190] A. Fortes and M. Manzato, “Ensemble learning in recommender systems: Combining multiple user interactions for ranking personalization,” *WebMedia 2014 - Proc. 20th Brazilian Symp. Multimed. Web*, pp. 47–54, 2014, doi: 10.1145/2664551.2664556.

[191] Y. Hao, P. Zhang, and F. Zhang, “Multiview Ensemble Method for Detecting Shilling Attacks in Collaborative Recommender Systems,” *Secur. Commun. Networks*, vol. 2018, 2018, doi: 10.1155/2018/8174603.

[192] T. Srikanth and M. Shashi, “A scalable ensemble architecture for collaborative filtering in recommender systems,” *Int. J. Appl. Eng. Res.*, vol. 11, no. 7, pp. 5103–5109, 2016.

[193] T. Ayaki, H. Yanagimoto, and M. Yoshioka, “Recommendation from access logs with ensemble learning,” *Artif. Life Robot.*, vol. 22, no. 2, pp. 163–167, 2017, doi: 10.1007/s10015-016-0346-x.

[194] A. F. Da Costa and M. G. Manzato, “Exploiting multimodal interactions in recommender systems with ensemble algorithms,” *Inf. Syst.*, vol. 56, pp. 120–132, 2016, doi: 10.1016/j.is.2015.09.007.

[195] M. Tiemann and S. Pauws, “Towards ensemble learning for hybrid music recommendation,” in *Proceedings of the 2007 ACM conference on Recommender systems - RecSys ’07*, 2007, no. May 2014, p. 177, doi: 10.1145/1297231.1297265.

[196] M. Tiemann and S. Pauws, “Towards ensemble learning for hybrid music recommendation,” in *Proceedings of the 2007 ACM conference on Recommender systems*

- *RecSys '07*, 2007, no. October, p. 177, doi: 10.1145/1297231.1297265.

[197] A. Sharafati, S. B. H. S. Asadollah, and M. Hosseinzadeh, "The potential of new ensemble machine learning models for effluent quality parameters prediction and related uncertainty," *Process Saf. Environ. Prot.*, vol. 140, pp. 68–78, 2020, doi: 10.1016/j.psep.2020.04.045.

[198] M. F. Tahir, C. Haoyong, K. Mehmood, N. A. Larik, A. Khan, and M. S. Javed, "Short Term Load Forecasting Using Bootstrap Aggregating Based Ensemble Artificial Neural Network," *Recent Adv. Electr. Electron. Eng. (Formerly Recent Patents Electr. Electron. Eng.*, vol. 13, no. 7, pp. 980–992, 2019, doi: 10.2174/2213111607666191111095329.

[199] S. Shamshirband, E. Jafari Nodoushan, J. E. Adolf, A. Abdul Manaf, A. Mosavi, and K. wing Chau, "Ensemble models with uncertainty analysis for multi-day ahead forecasting of chlorophyll a concentration in coastal waters," *Eng. Appl. Comput. Fluid Mech.*, vol. 13, no. 1, pp. 91–101, 2019, doi: 10.1080/19942060.2018.1553742.

[200] M. J. Alizadeh, E. Jafari Nodoushan, N. Kalarestaghi, and K. W. Chau, "Toward multi-day-ahead forecasting of suspended sediment concentration using ensemble models," *Environ. Sci. Pollut. Res.*, vol. 24, no. 36, pp. 28017–28025, 2017, doi: 10.1007/s11356-017-0405-4.

[201] R. Homsi *et al.*, "Precipitation projection using a CMIP5 GCM ensemble model: a regional investigation of Syria," *Eng. Appl. Comput. Fluid Mech.*, vol. 14, no. 1, pp. 90–106, 2020, doi: 10.1080/19942060.2019.1683076.

[202] C. L. Wu and K. W. Chau, "Prediction of rainfall time series using modular soft computingmethods," *Eng. Appl. Artif. Intell.*, vol. 26, no. 3, pp. 997–1007, 2013, doi: 10.1016/j.engappai.2012.05.023.

[203] Z. Chen *et al.*, "Beyond Point Estimate: Inferring Ensemble Prediction Variation from Neuron Activation Strength in Recommender Systems," *WSDM 2021 - Proc. 14th ACM Int. Conf. Web Search Data Min.*, pp. 76–84, 2021, doi: 10.1145/3437963.3441770.

[204] Y. Lee and K.-J. Kim, "Product Recommender Systems using Multi-Model Ensemble Techniques," *J. Intell. Inf. Syst.*, vol. 19, no. 2, pp. 39–54, 2013, doi: 10.13088/jiis.2013.19.2.039.

[205] A. Lommatzsch, "Real-time news recommendation using context-aware ensembles," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes*

Bioinformatics), vol. 8416 LNCS, pp. 51–62, 2014, doi: 10.1007/978-3-319-06028-6_5.

[206] H. Liang, X. Tao, Y. Xu, R. Nayak, and Y. Li, "Connecting users and items with weighted tags for personalized item recommendations," p. 51, 2010, doi: 10.1145/1810617.1810628.

[207] M. Jamali and M. Ester, "*TrustWalker*: a random walk model for combining trust-based and item-based recommendation," *KDD '09 Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. data Min.*, pp. 397–406, 2009, doi: citeulike-article-id:5151320.

[208] M. Nilashi, K. Bagherifard, M. Rahmani, and V. Rafe, "A recommender system for tourism industry using cluster ensemble and prediction machine learning techniques," *Comput. Ind. Eng.*, vol. 109, pp. 357–368, 2017, doi: 10.1016/j.cie.2017.05.016.

[209] M. Nilashi, O. Ibrahim, and K. Bagherifard, "A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques," *Expert Syst. Appl.*, vol. 92, pp. 507–520, 2018, doi: 10.1016/j.eswa.2017.09.058.

[210] C. F. Tsai and C. Hung, "Cluster ensembles in collaborative filtering recommendation," *Appl. Soft Comput. J.*, vol. 12, no. 4, pp. 1417–1425, 2012, doi: 10.1016/j.asoc.2011.11.016.

[211] W. Zhang, H. Zou, L. Luo, Q. Liu, W. Wu, and W. Xiao, "Predicting potential side effects of drugs by recommender methods and ensemble learning," *Neurocomputing*, vol. 173, pp. 979–987, 2016, doi: 10.1016/j.neucom.2015.08.054.

[212] R. Logesh, V. Subramaniyaswamy, D. Malathi, N. Sivaramakrishnan, and V. Vijayakumar, "Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method," *Neural Comput. Appl.*, vol. 5, 2018, doi: 10.1007/s00521-018-3891-5.

[213] M. G. Manzato *et al.*, "Mining unstructured content for recommender systems: An ensemble approach," *Inf. Retr. J.*, vol. 19, no. 4, pp. 378–415, 2016, doi: 10.1007/s10791-016-9280-8.

[214] A. Mosavi, F. Sajedi Hosseini, B. Choubin, M. Goodarzi, A. A. Dineva, and E. Rafiei Sardooi, "Ensemble Boosting and Bagging Based Machine Learning Models for Groundwater Potential Prediction," *Water Resour. Manag.*, vol. 35, no. 1, pp. 23–37, 2021, doi: 10.1007/s11269-020-02704-3.

[215] E. K. Ampomah, Z. Qin, G. Nyame, and F. E. Botchey, "Stock market decision support

modeling with tree-based adaboost ensemble machine learning models," *Inform.*, vol. 44, no. 4, pp. 477–489, 2020, doi: 10.31449/INF.V44I4.3159.

[216] S. J. Park, C. U. Kang, and Y. C. Byun, "Extreme gradient boosting for recommendation system by transforming product classification into regression based on multi-dimensional word2vec," *Symmetry (Basel).*, vol. 13, no. 5, 2021, doi: 10.3390/sym13050758.

[217] H. E. Kiziloz, "Classifier ensemble methods in feature selection," *Neurocomputing*, vol. 419, no. September, pp. 97–107, 2021, doi: 10.1016/j.neucom.2020.07.113.

[218] S. Pervaiz, Z. Ul-Qayyum, W. H. Bangyal, L. Gao, and J. Ahmad, "A Systematic Literature Review on Particle Swarm Optimization Techniques for Medical Diseases Detection," *Comput. Math. Methods Med.*, vol. 2021, 2021, doi: 10.1155/2021/5990999.

[219] H. Wu, K. Yue, Y. Pei, B. Li, Y. Zhao, and F. Dong, "Collaborative Topic Regression with social trust ensemble for recommendation in social media systems," *Knowledge-Based Syst.*, vol. 97, pp. 111–122, 2016, doi: 10.1016/j.knosys.2016.01.011.

[220] M. G. Vozalis and K. G. Margaritis, "Applying SVD on item-based filtering," in *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, 2005, vol. 3, no. 3, pp. 464–469, doi: 10.1109/ISDA.2005.25.

[221] B. Marlin, "Modeling user rating profiles for collaborative filtering," *Adv. Neural Inf. Process. Syst.*, 2004.

[222] D. Praveen Kumar, T. Amgoth, and C. S. R. Annavarapu, "Machine learning algorithms for wireless sensor networks: A survey," *Inf. Fusion*, vol. 49, pp. 1–25, 2019, doi: 10.1016/j.inffus.2018.09.013.

[223] M. E. Lopes, "Estimating the algorithmic variance of randomized ensembles via the bootstrap," *Ann. Stat.*, vol. 47, no. 2, pp. 1088–1112, 2019, doi: 10.1214/18-AOS1707.

[224] N. C. Oza, "Online Bagging and Boosting," pp. 2340–2345, 2006, doi: 10.1109/icsmc.2005.1571498.

[225] J. Vinagre, A. M. Jorge, and J. Gama, "Online bagging for recommender systems," *Expert Syst.*, vol. 35, no. 4, pp. 1–13, 2018, doi: 10.1111/exsy.12303.

[226] S. Al-Stouhi and C. K. Reddy, "Adaptive boosting for transfer learning using dynamic updates," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6911 LNAI, no. PART 1, pp. 60–75, 2011, doi: 10.1007/978-3-642-23780-5_14.

[227] L. Zheng, L. Li, W. Hong, and T. Li, “PENETRATE: Personalized news recommendation using ensemble hierarchical clustering,” *Expert Syst. Appl.*, vol. 40, no. 6, pp. 2127–2136, 2013, doi: 10.1016/j.eswa.2012.10.029.

[228] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, “A Survey on Ensemble Learning for Data Stream Classification,” *ACM Comput. Surv.*, vol. 50, no. 2, pp. 1–36, 2017, doi: 10.1145/3054925.

[229] K. Bauman, B. Liu, and A. Tuzhilin, “Aspect Based Recommendations,” *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. - KDD ’17*, pp. 717–725, 2017, doi: 10.1145/3097983.3098170.

[230] S. Kant and T. Mahara, “Merging user and item based collaborative filtering to alleviate data sparsity,” *Int. J. Syst. Assur. Eng. Manag.*, vol. 9, no. 1, pp. 173–179, 2018, doi: 10.1007/s13198-016-0500-9.

[231] C. Paper, “Cosine Similarity Metric Learning for Face for Face Veri cation,” *Accv*, no. September, pp. 709–720, 2015, doi: 10.1007/978-3-642-19309-5.

[232] E. Shmueli and T. Tassa, “Secure Multi-Party Protocols for Item-Based Collaborative Filtering,” *Proc. Elev. ACM Conf. Recomm. Syst. - RecSys ’17*, pp. 89–97, 2017, doi: 10.1145/3109859.3109881.

[233] J. Wang, A. P. De Vries, and M. J. T. Reinders, “Unifying user-based and item-based collaborative filtering approaches by similarity fusion,” *Proc. Twenty-Ninth Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, vol. 2006, pp. 501–508, 2006, doi: 10.1145/1148170.1148257.

[234] A. Beygelzimer, E. Hazan, S. Kale, and H. Luo, “Online gradient boosting,” *Adv. Neural Inf. Process. Syst.*, vol. 2015-Janua, pp. 2458–2466, 2015.

[235] M. T. Alam *et al.*, “Comparative Analysis of Machine Learning based Filtering Techniques using MovieLens dataset,” *Procedia Comput. Sci.*, vol. 194, pp. 210–217, 2021, doi: 10.1016/j.procs.2021.10.075.

[236] Q. Yao *et al.*, “Taking Human out of Learning Applications: A Survey on Automated Machine Learning,” pp. 1–20, 2018.

[237] M.-A. Zöller and M. F. Huber, “Benchmark and Survey of Automated Machine Learning Frameworks,” 2019.

[238] X. He, K. Zhao, and X. Chu, “AutoML: A Survey of the State-of-the-Art,” 2019.

[239] C. Troussas, A. Krouska, C. Sgouropoulou, and I. Voyiatzis, “Ensemble learning using fuzzy weights to improve learning style identification for adapted instructional routines,” *Entropy*, vol. 22, no. 7, 2020, doi: 10.3390/E22070735.

[240] Z. Zhu, Z. Wang, D. Li, Y. Zhu, and W. Du, “Geometric Structural Ensemble Learning for Imbalanced Problems,” *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1617–1629, 2020, doi: 10.1109/TCYB.2018.2877663.

[241] A. A. Feitosa Neto and A. M. P. Canuto, “An exploratory study of mono and multi-objective metaheuristics to ensemble of classifiers,” *Appl. Intell.*, vol. 48, no. 2, pp. 416–431, 2018, doi: 10.1007/s10489-017-0982-4.

[242] T. Gressling, *84 Automated machine learning*. 2020.

[243] P. Kordík, J. Černý, and T. Frýda, “Discovering predictive ensembles for transfer learning and meta-learning,” *Mach. Learn.*, vol. 107, no. 1, pp. 177–207, Jan. 2018, doi: 10.1007/s10994-017-5682-0.

[244] L. Kotthoff, C. Thornton, H. H. Holger, F. Hutter, and K. Leyton-Brown, “Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA,” *J. Mach. Learn. Res.*, pp. 1–5, 2017, doi: 10.1088/0953-4075/40/9/S11.

[245] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, “Automatic frankensteining: Creating complex ensembles autonomously,” *Proc. 17th SIAM Int. Conf. Data Mining, SDM 2017*, pp. 741–749, 2017, doi: 10.1137/1.9781611974973.83.

[246] R. A. Da Silva, A. M. D. P. Canuto, C. A. D. S. Barreto, and J. C. Xavier, “Automatic Recommendation Method for Classifier Ensemble Structure Using Meta-Learning,” *IEEE Access*, vol. 9, no. August, pp. 106254–106268, 2021, doi: 10.1109/ACCESS.2021.3099689.

[247] D. S. C. Nascimento, A. M. P. Canuto, and A. L. V. Coelho, “An empirical analysis of meta-learning for the automatic choice of architecture and components in ensemble systems,” *Proc. - 2014 Brazilian Conf. Intell. Syst. BRACIS 2014*, pp. 1–6, 2014, doi: 10.1109/BRACIS.2014.12.

[248] W. A. Albukhanajer, Y. Jin, and J. A. Briffa, “Classifier ensembles for image identification using multi-objective Pareto features,” *Neurocomputing*, vol. 238, pp. 316–327, 2017, doi: 10.1016/j.neucom.2017.01.067.

[249] J. R. Rice, “The Algorithm Selection Problem,” *Adv. Comput.*, vol. 15, no. C, pp. 65–118,

1976, doi: 10.1016/S0065-2458(08)60520-3.

[250] L. Rendell and H. Cho, “Empirical Learning as a Function of Concept Character,” *Mach. Learn.*, vol. 5, no. 3, pp. 267–298, 1990, doi: 10.1023/A:1022651406695.

[251] R. D. KING, C. FENG, and A. SUTHERLAND, “STATLOG: COMPARISON OF CLASSIFICATION ALGORITHMS ON LARGE REAL-WORLD PROBLEMS,” *Appl. Artif. Intell.*, vol. 9, no. 3, pp. 289–333, May 1995, doi: 10.1080/08839519508945477.

[252] K. Smith, F. Woo, V. Ciesielski, and R. Ibrahim, “Modelling the relationship between problem characteristics and data mining algorithm performance using neural networks,” *Intell. Eng. Syst. Through Artif. Neural Networks*, vol. 11, pp. 356–362, 2001.

[253] S. Ali and K. A. Smith-Miles, “Improved support vector machine generalization using normalized input space,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4304 LNAI, no. 2, pp. 362–371, 2006, doi: 10.1007/11941439_40.

[254] S. Shilbayeh and S. Vadera, “Feature selection in meta learning framework,” *Proc. 2014 Sci. Inf. Conf. SAI 2014*, pp. 269–275, 2014, doi: 10.1109/SAI.2014.6918200.

[255] M. Reif, F. Shafait, M. Goldstein, T. Breuel, and A. Dengel, “Automatic classifier selection for non-experts,” *Pattern Anal. Appl.*, vol. 17, no. 1, pp. 83–96, 2014, doi: 10.1007/s10044-012-0280-z.

[256] A. Filchenkov and A. Pendryak, “Datasets meta-feature description for recommending feature selection algorithm,” *Proc. Artif. Intell. Nat. Lang. Inf. Extr. Soc. Media Web Search Fruct Conf. AINL-ISMW Fruct 2015*, vol. 7, pp. 11–18, 2016, doi: 10.1109/AINL-ISMW-FRUCT.2015.7382962.

[257] R. A. Da Silva, A. M. D. P. Canuto, C. A. D. S. Barreto, and J. C. Xavier, “Automatic Recommendation Method for Classifier Ensemble Structure Using Meta-Learning,” *IEEE Access*, vol. 9, pp. 106254–106268, 2021, doi: 10.1109/ACCESS.2021.3099689.

[258] Y. H. Liu *et al.*, “Modification on the tribological properties of ceramics lubricated by water using fullerenol as a lubricating additive,” *Sci. China Technol. Sci.*, vol. 55, no. 9, pp. 2656–2661, 2012, doi: 10.1007/s11431-012-4938-y.

[259] A. Kalousis, “Algorithm Selection via Meta-Learning,” *Univ. Geneva, Genebra*, p. 283, 2002.

[260] R. R. Parente, A. M. P. Canuto, and J. C. Xavier, “Characterization measures of ensemble

systems using a meta-learning approach," *Proc. Int. Jt. Conf. Neural Networks*, 2013, doi: 10.1109/IJCNN.2013.6707016.

[261] R. A. Da Silva, A. M. D. P. Canuto, J. C. Xavier Junior, and T. B. Ludermir, "Using Meta-learning in the Selection of the Combination Method of a Classifier Ensemble," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2018-July, 2018, doi: 10.1109/IJCNN.2018.8489586.

[262] L. Raquel and B. Moreira, "Online ensembles of local recommendation models," 2021.

[263] R. Wang, Z. Wu, J. Lou, and Y. Jiang, "Attention-based dynamic user modeling and Deep Collaborative filtering recommendation," *Expert Syst. Appl.*, vol. 188, no. February, p. 116036, 2022, doi: 10.1016/j.eswa.2021.116036.

[264] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," *Int. Conf. Inf. Knowl. Manag. Proc.*, vol. 19-23-Oct-, pp. 811–820, 2015, doi: 10.1145/2806416.2806527.

[265] S. Li, Q. Liu, J. Dai, W. Wang, X. Gui, and Y. Yi, "Adaptive-Weighted Multiview Deep Basis Matrix Factorization for Multimedia Data Analysis," *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021, doi: 10.1155/2021/5526479.

[266] R. van den Berg, T. N. Kipf, and M. Welling, "Graph Convolutional Matrix Completion," 2017.

[267] N. Jankowski, W. Duch, and K. Grąbczewski, *Meta-Learning in Computational Intelligence*, no. 1. 2014.

[268] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002, doi: 10.1162/106365602320169811.

[269] J. A. Müller, A. G. Ivachnenko, and F. Lemke, "GMDH algorithms for complex systems modelling," *Math. Comput. Model. Dyn. Syst.*, vol. 4, no. 4, pp. 275–316, 1998, doi: 10.1080/13873959808837083.

[270] M. W. Browne, "Cross-validation methods. Journal of Mathematical Psychology," *J. Math. Psychol.*, vol. 44, no. 1, pp. 108–132, 2000.

[271] P. Kordik and J. Cerny, "Building predictive models in two stages with meta-learning templates optimized by genetic programming," in *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)*, 2014, no. April, pp. 1–8, doi:

10.1109/CIEL.2014.7015740.

[272] A. W. Moore and M. S. Lee, “Efficient Algorithms for Minimizing Cross Validation Error,” in *Machine Learning Proceedings 1994*, Elsevier, 1994, pp. 190–198.

[273] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *J. Mach. Learn. Res.*, vol. 18, pp. 1–52, 2018.

[274] R. I. McKay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O’neill, “Grammar-based Genetic programming: A survey,” *Genet. Program. Evolvable Mach.*, vol. 11, no. 3–4, pp. 365–396, 2010, doi: 10.1007/s10710-010-9109-y.

[275] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier, “Meta-Learning by Landmarking Various Learning Algorithms,” *Proc. Seventeenth Int. Conf. Mach. Learn. ICML2000*, vol. 951, no. 2000, pp. 743–750, 2000.

[276] C. Vijaya Kumar Reddy, D. Sreeramulu, and M. Raghunath, “Antioxidant activity of fresh and dry fruits commonly consumed in India,” *Food Res. Int.*, vol. 43, no. 1, pp. 285–288, 2010, doi: 10.1016/j.foodres.2009.10.006.

[277] J. You, Y. Wang, A. Pal, P. Eksombatchai, C. Rosenberg, and J. Leskovec, “Hierarchical Temporal Convolutional Networks for Dynamic Recommender Systems,” *ACM Ref. Format*, Apr. 2019.

[278] M. M. Tanjim, H. A. Ayuubi, and G. W. Cottrell, “DynamicRec: A Dynamic Convolutional Network for Next Item Recommendation,” *Int. Conf. Inf. Knowl. Manag. Proc.*, pp. 2237–2240, 2020, doi: 10.1145/3340531.3412118.

[279] T. Thanthriwatta and D. S. Rosenblum, “Improving Dynamic Recommendation using Network Embedding for Context Inference,” *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI*, vol. 2020-Novem, pp. 205–210, 2020, doi: 10.1109/ICTAI50040.2020.00041.

[280] D. Puspitaningrum, J. Fernando, E. Afriando, F. P. Utama, R. Rahmadini, and Y. Pinata, “Finding Local Experts for Dynamic Recommendations Using Lazy Random Walk,” *arXiv*, Jul. 2020, doi: 10.1109/CITSM47753.2019.8965354.

[281] C. Estaquio, N. Druesne-Pecollo, P. Latino-Martel, L. Dauchet, S. Hercberg, and S. Bertrais, “Socioeconomic Differences in Fruit and Vegetable Consumption among Middle-Aged French Adults: Adherence to the 5 A Day Recommendation,” *J. Am. Diet. Assoc.*, vol. 108, no. 12, pp. 2021–2030, 2008, doi: 10.1016/j.jada.2008.09.011.

[282] S. M. Rink *et al.*, “Self-Report of fruit and vegetable intake that meets the 5 a day recommendation is associated with reduced levels of oxidative stress biomarkers and increased levels of antioxidant defense in premenopausal women,” *J. Acad. Nutr. Diet.*, vol. 113, no. 6, pp. 776–785, 2013, doi: 10.1016/j.jand.2013.01.019.

[283] D. Benton and H. A. Young, “Role of fruit juice in achieving the 5-a-day recommendation for fruit and vegetable intake,” *Nutr. Rev.*, vol. 77, no. 11, pp. 829–843, 2019, doi: 10.1093/nutrit/nuz031.

[284] A. Naska *et al.*, “Fruit and vegetable availability among ten European countries: How does it compare with the ‘five-a-day’ recommendation?,” *Br. J. Nutr.*, vol. 84, no. 4, pp. 549–556, 2000, doi: 10.1017/s0007114500001860.

[285] Z. Zhu, J. Kim, T. Nguyen, A. Fenton, and J. Caverlee, “Fairness among New Items in Cold Start Recommender Systems,” *SIGIR 2021 - Proc. 44th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 767–776, 2021, doi: 10.1145/3404835.3462948.

[286] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, “Recurrent Recommender Networks,” *Proc. Tenth ACM Int. Conf. Web Search Data Min. - WSDM ’17*, pp. 495–503, 2017, doi: 10.1145/3018661.3018689.

[287] C. Cumby, A. Fano, R. Ghani, and M. Krema, “Predicting customer shopping lists from point-of-sale purchase data,” *KDD-2004 - Proc. Tenth ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 402–409, 2004, doi: 10.1145/1014052.1014098.

[288] C. Sun, R. Gao, and H. Xi, “Big data based retail recommender system of non E-commerce,” *5th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2014*, 2014, doi: 10.1109/ICCCNT.2014.6963129.

[289] Y. Koren, “Collaborative filtering with temporal dynamics,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’09*, 2009, p. 447, doi: 10.1145/1557019.1557072.

[290] F. Landi, L. Baraldi, M. Cornia, and R. Cucchiara, “Working Memory Connections for LSTM,” *Neural Networks*, 2021, doi: 10.1016/j.neunet.2021.08.030.

[291] D. Sánchez-Moreno, A. B. Gil González, M. D. Muñoz Vicente, V. F. López Batista, and M. N. Moreno García, “A collaborative filtering method for music recommendation using playing coefficients for artists and users,” *Expert Syst. Appl.*, vol. 66, pp. 1339–1351, 2016, doi: 10.1016/j.eswa.2016.09.019.

[292] S. Roweis and Z. Ghahramani, “A Unifying Review of Linear Gaussian Models,” *Neural Comput.*, vol. 11, no. 2, pp. 305–345, Feb. 1999, doi: 10.1162/089976699300016674.

[293] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[294] S. Sedhain, A. K. Menony, S. Sannery, and L. Xie, “AutoRec: Autoencoders meet collaborative filtering,” *WWW 2015 Companion - Proc. 24th Int. Conf. World Wide Web*, pp. 111–112, 2015, doi: 10.1145/2740908.2742726.

[295] T. Chen *et al.*, “MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems,” pp. 1–6, 2015.

[296] R. Salakhutdinov and A. Mnih, “Probabilistic matrix factorization,” *Adv. Neural Inf. Process. Syst. 20 - Proc. 2007 Conf.*, pp. 1–8, 2009.

[297] R. Jayashree and A. Christy, “Improving the enhanced recommended system using Bayesian approximation method and normalized discounted cumulative gain,” *Procedia Comput. Sci.*, vol. 50, pp. 216–222, 2015, doi: 10.1016/j.procs.2015.04.057.

[298] C. J. Willmott and K. Matsuura, “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” *Clim. Res.*, vol. 30, no. 1, pp. 79–82, 2005, doi: 10.3354/cr030079.

List of Publications

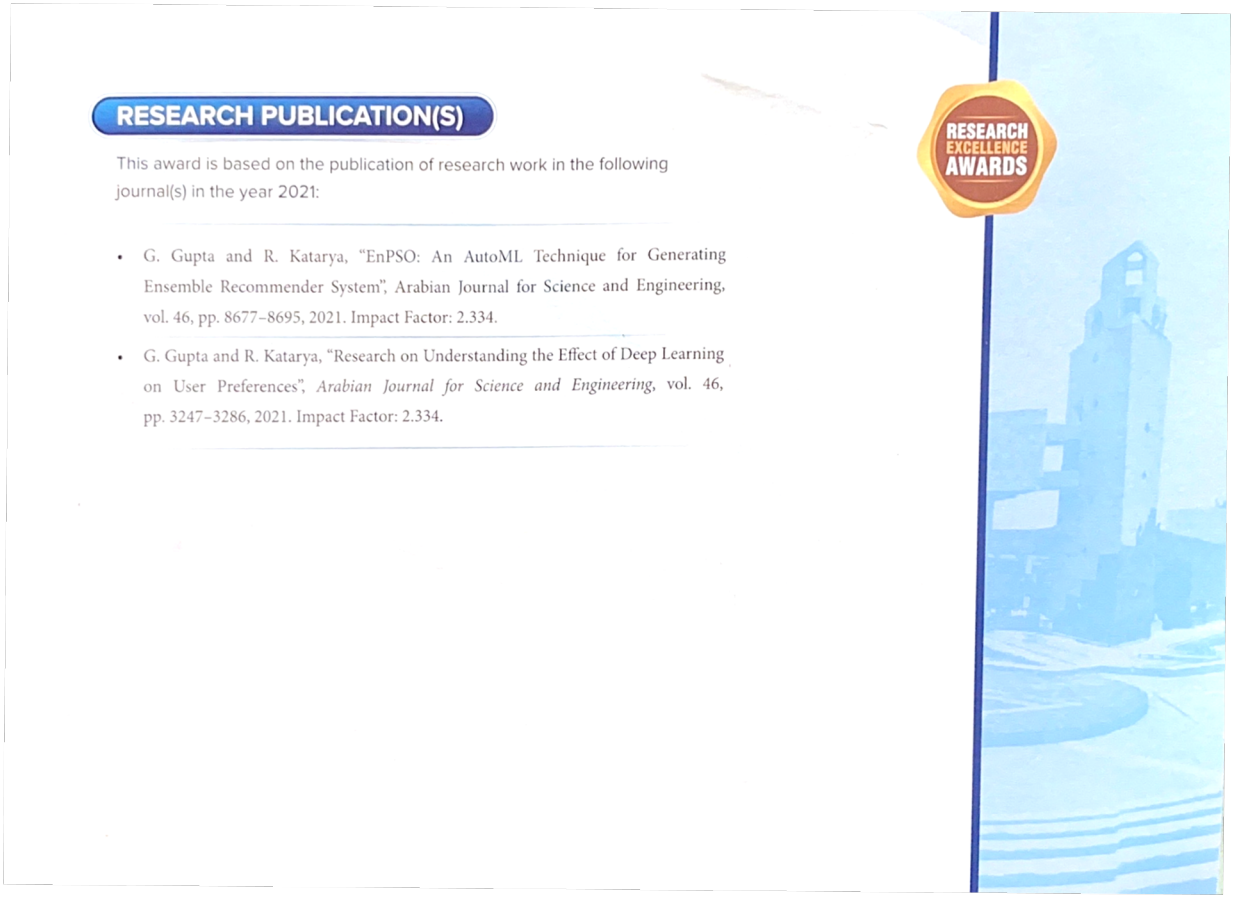
Journals

1. Gupta, G., & Katarya, R. (2021). Research on understanding the effect of deep learning on user preferences. Arabian Journal for Science and Engineering, 46(4), 3247-3286. (**Published, SCIE, IF: 2.3)**
2. Gupta, G., & Katarya, R. , A Computational Approach Towards Food- Wine Recommendations **(Communicated)**
3. Gupta, G., & Katarya, R. (2021). EnPSO: An AutoML technique for generating ensemble recommender system. Arabian Journal for Science and Engineering, 46(9), 8677-8695. **(Published, SCIE, IF: 2.3)**
4. Gupta, G., & Katarya, R. , En-DLR: Generating Recommendations with AutoML and Deep Learning **(Communicated)**
5. Gupta, G., & Katarya, R. , A Novel Approach To Alleviate Data Sparsity And Generate Dynamic Fruit Recommendations From Point-Of-Sale Data **(Major Revision, SCIE, IF: 1.5)**

International Conferences

1. Gupta, G., & Katarya, R. (2018, June). A study of recommender systems using Markov decision process. In 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 1279-1283). **IEEE**.
2. Gupta, G., & Katarya, R. (2021, May). A study of deep reinforcement learning based recommender systems. In 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC) (pp. 218-220). **IEEE**.
3. Gupta, G., & Katarya, R. (2019, November). Recommendation analysis on item-based and user-based collaborative filtering. In 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT) (pp. 1-4). **IEEE**

Research Excellence Award



RESEARCH PUBLICATION(S)

This award is based on the publication of research work in the following journal(s) in the year 2021:

- G. Gupta and R. Katarya, "EnPSO: An AutoML Technique for Generating Ensemble Recommender System", Arabian Journal for Science and Engineering, vol. 46, pp. 8677–8695, 2021. Impact Factor: 2.334.
- G. Gupta and R. Katarya, "Research on Understanding the Effect of Deep Learning on User Preferences", *Arabian Journal for Science and Engineering*, vol. 46, pp. 3247–3286, 2021. Impact Factor: 2.334.

Biography

Ms. Garima Gupta is currently designated as a Senior Research Fellow, a Ph.D. research scholar in the Department of Computer Science, Delhi Technological University, Delhi, India. She has completed her M.Tech in Computer Science from Indraprastha Institute of Information and Technology, Delhi. She has completed an undergraduate degree B.Tech in Computer Science from Guru Gobind Singh Indraprastha University, Delhi. She has published various research papers in SCIE/IEEE/SCOPUS indexed International Conferences/Journals. She is UGC-NET qualified. She has more than 3 years of experience in the IT industry. Her research area of interest includes Artificial Intelligence, Data mining, Machine Learning, Deep Learning, and Natural Language Processing. She is currently doing her research on recommender systems using computational intelligence techniques. She was also awarded the eminent “Commendable Research Award” in 2022 from Delhi Technological University, Delhi, India.