RECOMMENDATION SYSTEMS USING DEEP LEARNING METHODS

A thesis Submitted
In Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

by

KIRTI JAIN
(2K18/Ph.D/CO/20)

Under the Supervision of

Prof. Rajni Jindal
Professor, Department of Computer Science and Engineering
DELHI TECHNOLOGICAL UNIVERSITY

Department of Computer Science and Engineering

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

November, 2024

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

CANDIDATE'S DECLARATION

I, Kirti Jain, hereby certify that the work which is being presented in the thesis entitled **"Recommendation Systems Using Deep Learning Methods"** in partial fulfillment of the requirements for the award of the Degree of Doctor of Philosophy, submitted in the Department of Computer Science & Engineering, Delhi Technological University is an authentic record of my own work carried out during the period from August 2018 to June 2024 under the supervision of Prof. Rajni Jindal.

The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Ms. Kirti Jain
2K18/Ph.D/CO/20
Department of Computer Science & Engineering
Delhi Technological University, Delhi-110042

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.

Signature of Supervisor **Signature of External Examiner**

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

CERTIFICATE BY THE SUPERVISOR

Certified that **Kirti Jain (**2K18/Ph.D/CO/20) has carried out her research work presented in this thesis entitled **"Recommendation Systems Using Deep Learning Methods"** for the award of **Doctor of Philosophy** from Department of Computer Science & Engineering, Delhi Technological University, Delhi, under my supervision. The thesis embodies results of original work, and studies are carried out by the student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Prof. Rajni Jindal
Professor
Department of Computer Science and Engineering
Delhi Technological University, Delhi -110042

ACKNOWLEDGEMENT

With a deep sense of gratitude, I wish to express my sincere thanks to my supervisor, Prof. Rajni Jindal for her immense help in planning and executing this thesis work in time. She consistently stood by me in all my difficult times helping me to do my research fruitfully. I would like to express my sincere gratitude to her for excellent guidance, inspiration and support throughout my research work. I am grateful to her for the time she has spent with me in discussions and giving valuable suggestions. Her enthusiasm, innovative suggestions, and deep insight into technical matters were instrumental in the conceptualization of this work. Working under her supervision was a great learning experience.

I owe a deep sense of gratitude to Prof. Vinod Kumar, HOD of the Department of Computer Science and Engineering, DRC Chairperson, SRC, DRC members, faculty members, research scholars and non-teaching staff of the department for their valuable support and time to time help.

Moreover, I am profoundly grateful to my parents, in-laws, husband (Shivam Jain), brother, bhabhi, brother-in-law and son (Jeevesh Jain) whose unconditional support and blessings have been a cornerstone of my achievements. I would also like to thank my colleague (Dr. Ankit Vidyarthi) and my friends (Dr. Sanjay Kumar and Dr. Dipty Tripathi) for their constant motivational support.

Their unwavering faith in my abilities, endless lessons, moral support and care have played an integral role in shaping my academic pursuits. Their presence in my life has been truly invaluable, and I am forever indebted to them for their unwavering support and belief in my abilities.

Ms. Kirti Jain
Roll No: 2K18/Ph.D/CO/20
Department of Computer Science & Engineering
Delhi Technological University, Delhi-110042

Recommendation Systems Using Deep Learning Methods

Kirti Jain

Abstract

Recommendation systems (RS) are quite useful these days as they offer the content according to the users' tastes and interests. They are used in every online platform like social media, e-commerce, streaming services, news and content websites, traveling sites, job portals, online advertising, food delivery and restaurant apps, online learning platforms, dating and marriage related apps and many more. Some examples of recommendations include friend suggestions and news feed content on Facebook, job recommendation and content sharing on LinkedIn, movie recommendation on Netflix, products recommendation on e-commerce sites, hotels and flights recommendation on traveling sites, courses and learning material recommendation on E-learning platforms like Coursera, Udemy. Recommendation systems are mainly of three types: Content based RS, Collaborative-Filtering based RS (CFRS) and Hybrid Systems. Content-based RS are related to only one individual user and recommend items related to items' description and the user's personal choice/interests. Whereas, Collaborative RS creates a matrix of user-item pairs, which has each users' ratings for liked items. Now, a user gets the recommendation of items based on his interests as well as the based on the others users' interests with the similar profiles. Both Content based RS and CFRS have their own disadvantages. So, to overcome these disadvantages, hybrid systems take individual output of both content based RS and CFRS and then combine these outputs to make recommendations.

With the recent advancements in technology, Deep Learning (DL) models handle RS effectively. They are capable of handling intricate structures of data like image, text, audio, video and learn complex patterns from this data. This ability of DL to handle high dimensional data and learn hierarchical representation of features makes it suitable to be applied in RS. DL time-series models like RNN, LSTM have the capability to capture users' dynamic interests and their evolving temporal preferences. Such models help in capturing the changing needs of the users and make the recommendations accordingly.

Although plenty of data is available to be processed, processing the entire dataset is a cumbersome process. Sampling is a way to select a subset of the entire dataset which contains all the attributes that the database can represent. Many sampling techniques are combined with DL models to sample the data as well as to improve the performance of the RS. In this research work, we have discussed six types of sampling methods used for recommender systems, namely, Bayesian Hierarchical Sampling, Negative Sampling, Thompson Sampling, Bernoulli Sampling, Gibbs Sampling and Bootstrap Sampling.

Also, before processing the data, we need to clean it up as it contains a certain amount of noise. Noise can be described as malicious, natural, structural and contextual. So, there is a need to filter this noise before making the data suitable for processing. Thus, we present a summary of various noise filtering methods such as supervised, semi-supervised, unsupervised, crisp, fuzzy and optimization techniques.

Various companies spent a lot of revenue on designing good recommendation models in order to boost up their sales. Now, after this design, the question arises, how well the RS is working. That means, we need to measure its quality using some suitable evaluation metrics. So, choosing the appropriate evaluation metrics amongst the various existing evaluation metrics is a challenging task and thus, it becomes important to know which metric is to be used while measuring the performance of the system.

This research work presents four contributions in the recommendations systems domain. Firstly, an exploration of various sampling and noise filtering methods used for RS is presented. Secondly, an application of negative sampling and Nuclear Physics optimization is proposed on tweets data to enhance the recommendation of movies. Thirdly, a hybrid model of BERT-LSTM is proposed to improve the performance of hashtag recommendation models. Lastly, various performance evaluation metrics are explored in the field of recommendation systems and a novel metric named Semantic Recommendation Score (SRS) is proposed.

List of Publications

Papers Published / Communicated in International Journals:

- Kirti Jain, Rajni Jindal, "NLP-enabled Recommendation of Hashtags for Covid based Tweets using Hybrid BERT-LSTM Model", Transactions on Asian and Low-Resource Language Information Processing, 2024. (SCIE, ACM, IF: 2, Published)
- Kirti Jain, Rajni Jindal, "Sampling and Noise Filtering Methods for Recommender Systems: A Literature Review", Engineering Applications of Artificial Intelligence, Vol. 122C, 2023. (SCIE, Elsevier, IF: 8, Published)
- Kirti Jain, Rajni Jindal, "Optimization-based Noise Filtering Among User-Centric Tweets to Improve Predictions in Recommendation System", Knowledge and Information Systems, (SCIE, Springer, IF: 2.5, Communicated)
- Rajni Jindal, Kirti Jain, "A Review on Recommendation Systems Using Deep Learning", International Journal of Scientific & Technology Research, ISSN: 2277-8616, Vol. 8, Issue 10, 2019. (Scopus Indexed, Published)

Papers Published / Presented in International Conferences:

- Kirti Jain, Rajni Jindal, "A Survey on Hashtag Recommendations", 27th Conference of the Open Innovations Association FRUCT, Italy, ISSN: 2305-7254, 7-8 September, 2020, (pp. 323-327). (Published)
- Kirti Jain, Rajni Jindal, "A Walkthrough of Various Accuracy Measurement Metrics for Recommendation Systems", International Conference on Emerging Technologies in Engineering and Science, India, ISSN: 1551-7616, 11-12 August, 2023. (Presented)
- Kirti Jain, Rajni Jindal, "Bridging Gap Between Semantic Understanding and Linguistic Quality in Recommender Systems: A Semantic Recommendation Score (SRS) for Evaluation", 4th International Conference on Computing and Communication Networks (ICCCNet), United Kingdom, 17-18 October, 2024. (Presented & Received Best paper Certificate)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

RS	Recommendation Systems
CFRS	Collaborative Filtering based Recommendation Systems
DL	Deep Learning
RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory
BERT	Bidirectional Encoder Representations from Transformers
SRS	Semantic Recommendation Score
TF-IDF	Term Frequency Inverse Document Frequency
ML	Machine Learning
DNN	Deep Neural Networks
MF	Matrix Factorization
NPO	Nuclear Physics Optimization
BELHASH	Bert Embedding based LSTM for Hashtag Recommendation
L2R	Learning-to-Rank
BBC	British Broadcasting Corporation
LDA	Linear Discriminant Analysis
IBTM	Incremental Biterm Topic Model
LIBRA	Learning Intelligent Book Recommending Agent
KNN	K-Nearest Neighbour
DCFM	Deep Collaborative Filtering model
CF	Collaborative Filtering
mDAE	Marginalized Denoising Auto Encoder
SSL	Semi-Supervised Learning
POI	Points of Interest
PACE	Preference and Context Embedding
CNN	Convolutional Neural Networks
DeepCoNN	Deep Cooperative Neural Networks
SVD	Singular Value Decomposition
DLMF	Deep Learning based Matrix Factorization
GRU	Gated Recurrent Units

OTT	Over-the-top
SVM	Support Vector Machine
MV-RNN	Multi-View Recurrent Neural Network
VGG	Visual Geometry Group
MACON	Memory Augmented Co-attention model
GCN	Graph Convolutional Network
TOAST	SenTiment enhanced multi-mOdal Attentive hashtag recommendation
MLP	Multi Layer Perceptron
Bi-LSTM	Bidirectional Long Short Term Memory
UHMAN	User Guided Hierarchical Multi-Head Attention Network
CGAT	Contextualized Graph Attention Network
MV-GAN	Multi View Graph Attention Network
FL	Federated Learning
BPR	Bayesian Personalized Ranking
FeSoG	Federated Social recommendation with Graph Neural Network
LBSN	Location-based social networks
EM	Expectation Maximization
GNN	Graph Neural Network
CDR	Cross-Domain Recommendation
CCDR	Contrastive Cross-Domain Recommendation
MNS	Mixed Negative Sampling
MAB	Multi-Armed Bandit
CTR	Click Through Rate
PB-MHB	Position Based Metropolis-Hastings Bandit
DGR	Deep Generative Ranking
RBM	Restricted Boltzmann Machine
AoP	Average over Popular
HHT	Hilbert-Huang Transform
EMD	Empirical Mode Decomposition
TIA	Target Item Analysis
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model

CCPS	Co-Clustering with Propensity Similarity
PAM	Partition Around Median
RIS	Randomness in Item Selection
DTEC	Dual Training Error based Correction approach
CARS	Context-Aware Recommender System
NDC	Noise Detection and Correction
RCFS	Relaxed Context Feature Sets
NORMA	Noise Resilient Matrix Approximation
GRS	Group Recommendation Systems
DCBM	Dynamic Coherence-Based Modeling
NNMG-FT	Natural Noise Management in Group Recommendation using Fuzzy Tools
CL	Contrastive Learning
DQN	Deep Q-Networks
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
CBoW	Continuous Bag-of-Words
NEL	Neutron Enrichment Level
EB_u	Enrichment Bound
MAP	Mean Average Precision
POS	Part of Speech
MLB	MultiLabelBinarizer
API	Application Programming Interface
URL	Uniform Resource Locator
HTML	Hypertext Markup Language
MAE	Mean Absolute Error
NMAE	Normalized Mean Absolute Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MRR	Mean Reciprocal Rank
CHR	Cumulative Hit Rate
BLEU	Bilingual Evaluation Understudy
ROUGE	Recall-Oriented Understudy for Gisting Evaluation

RDMA	Rating Deviation from Mean Agreement
SSADR-CoF	Semi-Supervised Attack Detection in Recommendation based on Co-Forest Algorithm
IRM-TIA	Item Relationship Mining - Target Item Analysis
SCoR	Social Collaborative Recommendation

CHAPTER 1

INTRODUCTION

Recommendation is all about suggesting the items to the user according to his tastes and preferences. Recommendation systems are the techniques that help the users get the items according to their needs. As the number of visitors on the Internet is growing day-by-day, it has become necessary to filter out information and present it according to users' preferences and taste of interests. Users can find products that suit their tastes with the aid of recommendation systems. They are crucial in presenting the most relevant results based on the interests and preferences of the users. In order to predict the next best content, these systems take into account users' dynamic choices in addition to their static interests. They are regarded as playlist generators for music and videos on Youtube [1] and Netflix [2]; they also recommend hashtags and facilitate connections between users on social media platforms like Facebook, Twitter [3]; they support e-commerce sites like Flipkart, Amazon [4], by indicating the next item a customer should buy based on item ratings. Also, these systems are becoming more and more useful in areas other than amusement and online shopping. They support the recommendation of individualized treatment plans and health-related content in the healthcare industry. Recommendation systems in education aid in customizing learning materials to meet the needs of each individual student, improving the educational process.

Additionally, recommendation systems are essential in helping users find fresh and pertinent content, which increases user enjoyment and involvement in the field of content creation and consumption. They have become a crucial component of contemporary digital platforms, greatly improving user experience. These systems are capable of making recommendations for goods, services, or content that closely match personal preferences by examining consumer habits and choices. These systems work best when they can process large amounts of data and provide precise, personalized recommendations that increase user participation and engagement. With personalized experiences across a variety of domains, recommendation technologies

promise to become even more essential to our everyday online experiences as they continue to advance.

1.1 Phases of Recommendation Process

Recommendation process involves three phases as described in fig.1.1.

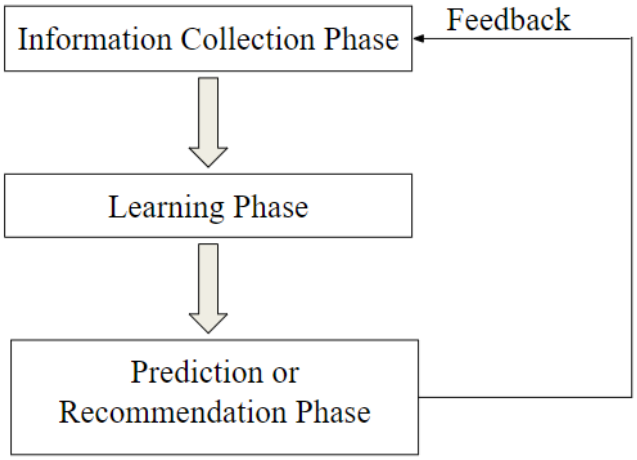

Fig.1.1: Different Phases of Recommendation Systems

1.1.1 Information Gathering Phase

This stage is in charge of gathering data about users in order to create user profiles, which contain information about users' characteristics, online activities, and the sites they access. To accomplish this, feedback in the form of implicit, explicit or hybrid feedback is gathered.

1. **Explicit Feedback**- In this model, users are asked directly via the application's layout for details regarding the ratings of the products they are considering. This model requires users to exert additional effort in order to provide rating knowledge.
2. **Implicit Feedback** - This stage gathers consumer reviews immediately for the products they are keen on. This is accomplished by keeping an eye on various user actions, like browsing through past purchases, spending time on certain websites, and clicking buttons during specific events. This model is less accurate even though it doesn't require users to exert additional effort.

3. **Hybrid Feedback** - The benefits of both explicit and implicit feedback models are combined in the hybrid feedback model. This is accomplished by having a hidden verification on the user's explicit comments.

1.1.2 The Learning Stage

Appropriate algorithms are used in this stage to identify user features from the information gathering phase's data.

1.1.3 Phase of Prediction and Recommendation

In the end, this stage anticipates the items the user might prefer and then suggests them. Its output is forwarded to the information gathering stage as feedback.

1.2 Recommendation Techniques

Fig.1.2 describes the various techniques used in recommendation systems.

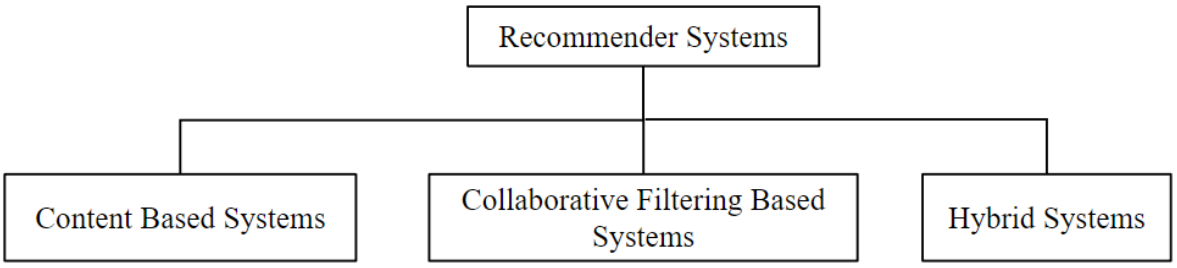

Fig.1.2: Various Techniques used in Recommendation Systems

1.2.1 Content-Based RS

This approach bases its recommendations on characteristics that have been taken from the item's content that the user has previously accessed. For example, in an online shopping site, suppose one likes t-shirts and denim, then the system will capture the user preferences like style, material, color, and then recommends similar items from new collections. It is suitable for recommending news articles and web pages. It uses Term Frequency Inverse Document Frequency (tf-idf) statistical model and Neural Networks, Decision Trees and the Naive Bayes Classifier as Probabilistic Models.

Disadvantages of Content-Based Filtering:

- Recommendations for Limited Cross-Domain Use - Content-based solutions usually work in only one sector or subject space, like music, movies, or literature. Their capacity to deliver a variety of recommendations to users with a wide range of interests is limited as they fail to recommend products across multiple domains or content areas.
- Limited Content Analysis- This limitation results from the system's heavy reliance on item metadata, including tags, descriptions, and other attributes. In order for the recommendation system to function properly, the metadata must be comprehensive, accurate, and reflect the qualities of the items.

1.2.2 Collaborative Filtering based RS (CFRS)

This method works by constructing a user-item matrix and then comparing users' profiles with similar interests for recommending the items. The matrix contains ratings of the items given by that user. Now, a user gets the recommendation of items based on his interests as well as the based on the others users' interests with the similar profiles. For example, if a user is watching cricket on YouTube, he may get recommendations for football as well. This is because another user watching cricket, has interests for football as well. Fig.1.3 explains the user-item matrix. CFRS is of two types - Memory based and Model based.

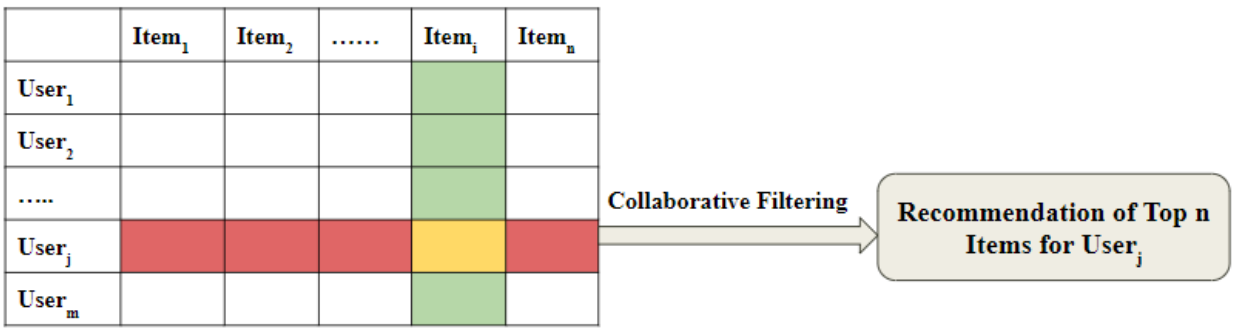

Fig.1.3: User-Item Matrix for Collaborative Filtering

a) Techniques based on Memory

There are two approaches to applying memory-based techniques: item-based and user-based.

- Item-Based Collaborative Filtering - This model suggests products to a user based on how well they match other products the user has already used. Before recommending an item for an active user, it considers the ratings of all items that are comparable to the one being recommended. Compared to user-based collaborative filtering, this approach is more scalable and reliable, but it may have trouble with new or sparse items.

- User-Based Collaborative Filtering: In order to ascertain user similarity, this model contrasts user ratings on the same item. The item is then recommended for that user if a user with similar interests to the active user purchases it.

b) Techniques based on Model

Based on an active user's past ratings of items, this model recommends an item using a variety of machine learning or data mining algorithms.

Model-Based Recommendation Systems employ the following algorithms:

1) Association Rules
2) Decision Trees
3) Clustering
4) Artificial Neural Networks

Disadvantages of Collaborative Filtering:

- Cold-Start Issue: This issue arises when a user accesses the website for the first time or when a new item is introduced. At that point, the database contains no historical records from which to recommend items for that user.
- Sparsity Problem: This issue arises when a product receives extremely few ratings from users. As a result, it might occasionally be challenging to suggest

that product to a user because it is impossible to identify consumer similarities from the user-item matrix.

1.2.3 Hybrid Systems

This method leverages the advantages of both collaborative filtering and content based recommendation techniques in order to enhance the efficiency of the system and thus increase the performance. It is done as the combination of various models will be more effective than individual models in recommending an item to a user. This can be done by embedding some content based model into collaborative filtering model or by embedding some collaborative filtering content into content based model. These systems combine the predictions of an independent content result and a collaborative recommender system through a voting scheme. The hybrid model of RS is depicted in fig.1.4.

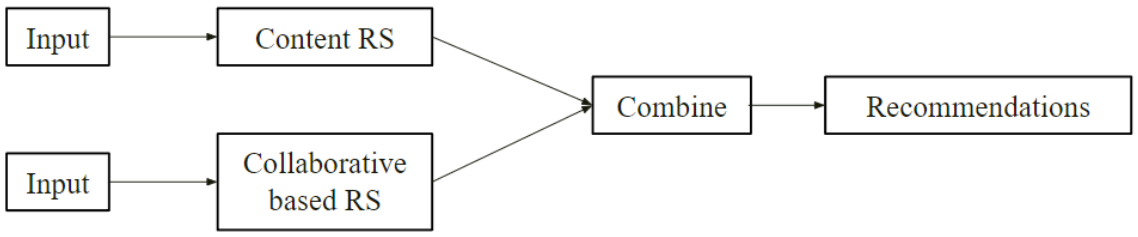

Fig.1.4: Hybrid model of recommendation Systems

1.3 Introduction to Deep Learning

With the growing technology in the field of data management and processing, Deep Learning (DL) is an advancement to Machine Learning (ML). A prominent example of DL is Deep Neural Networks (DNN). In DNN, there are a number of interconnections in between the nodes giving rise to the multiple hidden layers in between the input and the output layers. The different layers of DNN process the input data, identifies the patterns hidden in the data and forward that information to the subsequent layers for further processing. These Deep Neural Networks provide a level of abstraction in identifying the various patterns in the dataset. Also, DNN automatically does the task of feature extraction by themselves in comparison to manual work in ML.

DL has attained good popularity and attention because of its nature to identify various hidden patterns in the data. They are capable of handling intricate structures of data like image, text, audio, video and learn complex patterns from this data. This ability of DL to handle high dimensional data and learn hierarchical representation of features makes it suitable to be applied in a number of fields such as healthcare [5], computer vision [6], Natural Language Processing [7] and many more [8-10]. Some of the applications where DL is widely used include classification, image processing, object detection, sentiment analysis, medical diagnosis, machine translation.

1.4 Challenges in Recommendation Systems

Matrix Factorization (MF) was traditionally used in recommender systems. It constructs a user-item interaction matrix which consists of ratings for the items given by the users and then it discovers the latent factors from this ratings matrix and then maps the items and the users against those factors. In order to uncover the hidden patterns of users and items, this collaborative filtering technique splits the matrix representing users and items into two reduced-rank matrices. This makes the recommendations possible for users about their items of interest. A few MF drawbacks are listed below, along with how DL fixes them.

- **Limited Representation** - MF fails to represent complex and non-linear patterns in the data. It also fails to represent high-dimensional data in an effective way to find out the hidden features from that data. On the other hand, DL handles intricate data structures like video, audio, images, text and thus it is able to represent high dimensional data.
- **Cold-Start Problem** - When a new user enters the system or a new item is introduced, then MF fails to identify the similarity of that new user with the other users or the latent features of the new item to predict its ratings. So, here MF fails in answering the question, “what to recommend to a new user” or “to whom to recommend a new item”. But, DL captures users’ demographics and item attributes and learns about the preferences of the new user or the similarities of the new item to other items.
- **Feature Extraction** - MF requires features to be extracted manually, which

leads to time consumption or wastage of resources, whereas, DL has the ability to automatically extract the latent features from the data.

- **Scalability and Efficiency** - MF fails in dealing with large datasets, leading to increased computational costs and much awaited training time, while, DL has the ability to handle large volumes of data effectively and thus it is able to make personalized recommendations to users.

1.5 Role of Deep Learning in Recommender Systems

There are plenty of abilities that make DL suitable for recommendation systems. Few of them are described below:

- **Feature Learning -** DL has the ability to automatically extract the latent features from the data. This makes it suitable to extract the latent features from users' profiles to identify the similarities among them. It also extracts features for identifying patterns using item attributes.
- **Personalization** - DL has the ability to handle intricate data structures and thus it is able to make personalized recommendations to users by looking at their reviews, button clicks, likes/dislikes, ratings for an item.
- **Temporal Dynamics** - DL time-series models like RNN, LSTM have the capability to capture users' dynamic interests and their evolving temporal preferences. Such models help in capturing the changing needs of the users and make the recommendations accordingly.

1.6 Significance of the Research

In this section, we describe the significance of our research in context to recommendation systems.

1.6.1 Motivation

- As the information on the Internet is increasing, users find it difficult to look for the right information.
- Recommendation systems have emerged as the powerful tool to filter information and present it in accordance with users' preferences and taste of interests.

- With the advancements in deep learning techniques, RS have evolved to work in a much more precise and accurate manner.
- This research is beneficial for society as RS not only saves time in helping the users get their prescribed information, but increases the user engagement and satisfaction with the system.

1.6.2 Sampling and Noise Filtering

In the era of online business, many e-commerce sites have evolved which recommend items according to one's needs and interests. Plenty of data is available to be processed to make the recommender systems work effectively and efficiently. But, processing the entire dataset is a cumbersome process. So, there is a need to select a part of the data to be processed easily. Sampling is a way to select a subset of the entire dataset which contains all the attributes that the database can represent. It is important to understand which type of sampling method is suitable for a particular application in recommender systems. Thus, there is a need to study various sampling methods previously used in recommender systems. Also, before processing the data, we need to clean it up as it contains a certain amount of noise. This noise is described as malicious or natural or structural or contextual. Malicious noise is implicitly inserted in the system to alter the behavior of the system. Natural noise enters the systems unknowingly due to the reluctance of users in giving proper ratings to the items. Structural noise arises due to the inconsistencies and irregularities in the structure of data format. Contextual noise is due to the changing contexts of the users like time, place, mood etc. So, there is a need to filter these types of noise before making the data suitable for processing.

1.6.3 Evaluation Metrics

Implementing the appropriate model for your problem of recommendation is the first challenging task. But, choosing the right metric to evaluate that recommender model is another challenge. Sometimes, it may happen that selecting the wrong metric to evaluate the model may give worse results even if the model is correctly designed. So, it is important to know which metric is to be used while measuring the performance of the system.

1.7 Research Gaps and Objectives

The following research gaps are identified:

- Most of the studies have not discussed about the sampling methods. There is a need to explore some sampling methods used for recommendation purposes.
- Most of the studies have done data preprocessing which makes the data suitable to be processed for recommendation tasks. But, many studies have not taken care of the noise entering the system. Thus, there is a need to explore methods for noise filtration to improve the results.
- Most of the studies include methods to capture the semantics of the words in a sentence. But, the context of the word is not taken care of. So, a deep learning based approach can be explored to capture the context along with the semantics of the words.
- Most of the studies have evaluated their work using Recall, Precision and F1-Score to measure the accuracy of their systems. But very few studies have taken care of linguistic related metrics. So, other accuracy measurement metrics can be explored.

The research objectives are as follows:

1. To explore various sampling methods used in recommendation systems.
2. To critically analyze various noise filtering methods used for recommendation systems.
3. To develop a suitable model using a deep learning approach to capture the contextual relationships between words in a sentence for text based recommender systems.
4. To explore various accuracy measurement metrics in recommendation systems.

1.8 Organization of the Thesis

This section presents the thesis's structure, which is divided into the following seven chapters:

Chapter 1: Introduction

This chapter introduces the recommendation systems, deep learning techniques, challenges of RS, how DL overcomes these challenges, motivation leading to this research, research gaps and objectives.

Chapter 2: Literature Review

This chapter presents a brief description of the existing work related to recommendation systems.

This work has resulted in the publication of the following papers:

- Rajni Jindal, Kirti Jain, “A Review on Recommendation Systems Using Deep Learning”, International Journal of Scientific & Technology Research, ISSN: 2277-8616, Vol. 8, Issue 10, 2019. (Scopus Indexed)
- Kirti Jain, Rajni Jindal, “A Survey on Hashtag Recommendations”, 27th Conference of the Open Innovations Association FRUCT, Italy, ISSN: 2305-7254, 7-8 September, 2020, (pp. 323-327).

Chapter 3: Sampling and Noise Filtering Methods

This chapter explains the needs of sampling methods as well as various noise filtering methods that are used for recommendation systems. This chapter includes a comprehensive survey of the above mentioned two essential parts of RS.

This work has resulted in the publication of the following paper:

- Kirti Jain, Rajni Jindal, “Sampling and Noise Filtering Methods for Recommender Systems: A Literature Review”, Engineering Applications of Artificial Intelligence, Vol. 122C, 2023. (SCIE, Elsevier, IF: 8)

Chapter 4: NPO based Movie Recommendation Model

This chapter covers a detailed description of the proposed movie recommendation model using negative sampling and Nuclear Physics Optimization (NPO) to remove irrelevant tweets.

This work has resulted in the communication of the following paper:

- Kirti Jain, Rajni Jindal, “Optimization-based Noise Filtering Among User-Centric Tweets to Improve Predictions in Recommendation System”, Knowledge and Information Systems, (SCIE, Springer, IF: 2.5, Communicated)

Chapter 5: Proposed model for Hashtag Recommendation

This chapter describes the proposed model for hashtag recommendation which is based on the hybridization of BERT and LSTM.

This work has resulted in the publication of the following paper:

- Kirti Jain, Rajni Jindal, “NLP-enabled Recommendation of Hashtags for Covid based Tweets using Hybrid BERT-LSTM Model”, Transactions on Asian and Low-Resource Language Information Processing, 2024. (SCIE, ACM, IF: 2)

Chapter 6: Evaluation Metrics

This chapter mainly describes the various evaluation metrics that are or can be used for recommendation systems. It also introduces a novel metric named Semantic Recommendation Score (SRS) to evaluate the language based models.

This work has resulted in the presentation of the following papers:

- Kirti Jain, Rajni Jindal, “A Walkthrough of Various Accuracy Measurement Metrics for Recommendation Systems”, International Conference on Emerging Technologies in Engineering and Science, India, ISSN: 1551-7616, 11-12 August, 2023.
- Kirti Jain, Rajni Jindal, “Bridging Gap Between Semantic Understanding and Linguistic Quality in Recommender Systems: A Semantic Recommendation Score (SRS) for Evaluation”, 4th International Conference on Computing and Communication Networks (ICCCNet), United Kingdom, 17-18 October, 2024. (Received Best paper Certificate)

Chapter 7: Conclusion

The research work's conclusion and future scope are presented in the final chapter. The significance of our suggested models for different recommendation systems is discussed in this chapter. It also draws attention to the social effects of recommendation system research.

1.9 Contributions of the Thesis

The thesis contributions are as follows:

1. A comprehensive survey of various sampling methods as well as noise filtering methods is done. This survey helps in understanding which type of sampling method is suitable for a particular application in recommender systems and how the data needs to be cleaned before processing.
2. After the critical analysis of sampling and noise filtering methods is done, a movie recommendation system is proposed, which is based on training the model with negative examples and removing the noise with Nuclear Physics Optimization.
3. Covid-19 based tweets are sampled to capture the contextual relationships between words in a sentence. These tweets are evaluated by a proposed model named BELHASH for hashtag recommendations. This model works on BERT-LSTM layers.
4. Choosing the appropriate evaluation metrics amongst the various existing evaluation metrics is a challenging task and thus, it becomes important to know which metric is to be used while measuring the performance of the system. Therefore, a study of various evaluation metrics is made which are used for Recommendation systems. We also introduce a novel metric named Semantic Recommendation Score (SRS) to evaluate the language based models.

CHAPTER 2

LITERATURE REVIEW

A review of the research on the recommendation systems is given in this chapter. Key studies have been enumerated and summarized, and they are provided below in two sections:

- The work on traditional recommender systems is covered in the first section.
- The different deep learning techniques applied to recommender systems are covered and summarized in the second section.

2.1 Traditional Recommender Systems

By offering users tailored suggestions, recommendation systems are essential in tackling the problem of information overload. Conventional recommendation systems have been thoroughly researched and used in a wide range of industries, including social media, content streaming platforms, e-commerce, and entertainment. These systems are designed to help users find relevant content like movies, music, products or articles by using their past interactions and preferences as a basis.

This literature review delves into the fundamental ideas, workings, uses cases, advantages, and disadvantages of conventional recommendation techniques. Understanding these systems' foundations will help us explore more sophisticated methods, like deep learning, in the subsequent sections.

An overview of content-based recommendation systems and an exploration of the fundamentals of content-based filtering and its use in recommendation systems can be found in the study in [11]. It highlights how important it is to comprehend item attributes and user preferences based on content features. Discussions of several content-based recommendation system approaches, including feature extraction methods, similarity metrics and user modeling strategies are included in this study. It also includes text-based datasets like news articles, movie descriptions or product attributes, and illustrates practical uses of content-based filtering. Two drawbacks of

content-based systems - Overspecialization and the cold-start issue, are also discussed in this study.

Tagging is one of the effective tools that has emerged to help users locate, arrange, and comprehend online entities. Similar to this, recommender systems let users quickly browse through sizable item collections. The automation found in recommenders as well as the adaptability and conceptual clarity found in tagging systems may be provided by algorithms that combine tags and recommenders. The paper [12] investigates tagommenders, recommender systems that forecast users' inclinations for objects by utilizing their deduced tag preferences. It displays algorithms for inferring tag preferences based on user interactions with tags and movies. To predict the movie ratings of 995 MovieLens users, these algorithms are tested using their inferred tag preferences.

In order to increase users' interests in specific news on Twitter, the authors of the study [13] tackle the challenge of social media tagging to a news feed as a Learning-To-rank problem. The hashtag relevance is illustrated using the L2R Method. In this context, news reports function as queries and hashtags as documents. The process of recommendation includes two stages: first, every story is linked to a hashtag stream to generate potential tags through a pre-ranking step; next, L2R is utilized to assess every possible tag's relevance and suggest particular hashtags. The dataset is made up of seven organizations' RSS news feeds: The Journal, RTE, Irish Independent, Irish Times, Irish , Irish Examiner, BBC and Reuters. Additionally, this study suggests a path for further research into how social media tagging affects digital story Recognition and monitoring.

The authors of the research [14] present a learning-to-rank method called Hashtagger+ for real-time social media tagging of Twitter newsfeed. Prior to creating the Content-Tag Attribute Vector, key phrases are obtained from news reports and potential hashtags are pulled from pertinent tweets. This attribute vector then incorporates Hashtagger+ to suggest tags. In the future, this method can be used for mining text as well as monitoring and analysis of stories of interest.

Hashtags on Twitter are suggested based on a combination of users' changing desires and live content [15]. The User-Tweet LDA Model is used to discover users' evolving goals. To find hidden topics—that is, topics that are currently being used with Real-time Twitter feeds—another model called the Incremental Biterm Topic Model (IBTM) is employed. The User-IBTM model performs social media tagging tasks by utilizing these two models.

A content-based recommendation method emphasizing book recommendations based on textual descriptions is presented in [16]. Their approach uses machine learning techniques to analyze user preferences and book content through text categorization. Their algorithm, LIBRA (Learning Intelligent Book Recommending Agent), creates customized recommendations based on the interests of each user by utilizing features that are taken from book descriptions. The authors conduct the experiments on a dataset of book descriptions from Amazon.com to show the effectiveness of their approach.

One crucial application in the field of information filtering is recommender systems. The authors of [17] present a novel probabilistic factor analysis framework that seamlessly integrates the preferences of trusted friends and users. They refer to the formulation of the social trust constraints on the recommender systems in this framework as the "Social Trust Ensemble". Factorization of the user-item matrix involves learning user features through the application of matrix factorization. The experiments are conducted on the Epinions dataset.

The volume of work involved in conventional collaborative filtering systems rises with increasing user base. The authors of the work [18] looked into item-based collaborative filtering strategies. Item-based methods begin by calculating item-item similarities using the user-item relationship represented using a matrix to ascertain the connections between different items. These relationships are then used to compute recommendations for users in an indirect manner.

In order to model user preferences, recommender systems passively track various forms of user behavior, such as past purchases, viewing habits, and browsing activity. In the study [19], authors pinpoint the distinct characteristics of implicit feedback

datasets. They suggest interpreting the data as indicating both positive and negative preferences with widely differing degrees of confidence. This results in a factor model that is specifically designed for recommenders of implicit feedback. Additionally, they also propose a scalable optimization process that increases linearly in the size of the data. A television show recommender system effectively employs this algorithm.

Another study [20] suggests a single probabilistic framework for combining content-based and collaborative recommendation systems into a hybrid model. When content and collaborative methods are combined with conventional Expectation Maximization (EM) learning algorithms, global probabilistic models frequently cause severe overfitting in the sparse data scenarios. However, the three-way relationship data between items, users and product content is included in this study. This study demonstrates that sparsity can frequently be overcome by using secondary content information. Researchlndex, a library of computer science publications, is used for the experiments. The results indicate that suitable mixture models with secondary data yield much higher-quality recommendations than k-nearest neighbors (kNN). Compared to local approaches like KNN, global probabilistic models enable more general inferences.

2.2 Deep Learning Approaches for Recommender Systems

Deep Learning models have the ability to identify various hidden patterns in the data and thus, they are capable of handling intricate structures of data like image, text, audio, video and learn complex patterns from this data. This ability of DL to handle high dimensional data and learn hierarchical representation of features makes it suitable to be applied in the field of recommendation systems. DL models extract the latent features from users' profiles to identify the similarities among them. They also extract features for identifying patterns using item attributes. They make personalized recommendations to users by looking at their reviews, button clicks, likes/dislikes, ratings for an item. They also capture users' dynamic interests and their evolving temporal preferences, thus making personalized recommendations.

The Deep Collaborative Filtering Model (DCFM) is an integrated model that combines collaborative filtering models with Deep Learning (DL) algorithms. This is done to fix issues with cold start and sparsity problems. One version of DCFM is used in [21], where the sparsity issue of collaborative filtering (CF) is solved by utilizing Bayesian Stacked Denoising Autoencoders to build a hierarchical Bayesian model known as Collaborative Deep Learning. Three datasets—Netflix, CiteULike-t and CiteULike-a—are used in the experiments. The findings demonstrate improved performance after combining DL for information on content with CF for matrix of ratings.

The difficulties with data sparsity and matrix decomposition (matrix factorization) issues that arise with collaborative filtering are highlighted in another study [22]. By integrating CF's matrix decomposition with the Marginalized Denoising Auto Encoder (mDAE), it marginalizes the gap between CF and DL. It outperforms other approaches in terms of response prediction and movie and book recommendations, and it uncovers hidden variables for recommendations on videos from both side information and user-item scores.

Recommendations on YouTube are generated by integrating CF and Neural Networks [23]. This integration is used to predict how long users will watch a video according to whether they clicked on it (positive) or not (negative). This method boosts the efficiency and recent viewing activities by taking into account the characteristics of time-sensitive attributes of those videos.

In another work [24], CF is paired with Semi-Supervised Learning (SSL) to address the issue of information scarcity in recommending the Points of Interest (POI). POI recommendation is also termed as "location-based recommendation" or "geospatial recommendation. To evaluate user-POI interactions, a novel framework named, Preference and Context Embedding (PACE) is developed. It integrates CF and SSL and is based on neural networks. This work is tested with the Gowalla and Yelp check-in datasets, yielding positive outcomes.

Two-Phase Regularization for matrix factorization utilizing deep neural networks is presented in [25]. This involves accumulating CNN and gated RNN to create

complex neural networks. Here, textual information for item recommendations is first extracted, and then MF is combined with deep neural networks to create a latent image of items and users. 4 Datasets are used for the experiments - Amazon Instant Video, Apps for Android, Kindle Store, and Yelp.

A content based recommendation algorithm is developed using CNN model for recommending learning resources [26]. CNN is utilized to forecast grading scores between students and learning resources based on text information in learning resources. In this approach, Language Model is employed for its input to train CNN. Latent Factor Model with L1 norm is employed for its output. The famous Book-Crossing dataset is used for this purpose. The proposed algorithm is feasible in recommending new and unpopular learning resources.

CNNs are also widely used in Social Networks. CNN incorporates a local attention channel which encodes a few trigger words and represents embeddings of those words; and a global channel which encodes all the words and represents embedding of the entire microblog to perform hashtag recommendation tasks [27] in social networks. The results reveal that the proposed method outperforms the other methods which consider only local or global information.

Stackoverflow dataset is used for the study in [28], where word embedding is used to represent user profiles and their posted questions. CNN is then used to recommend experts based on their profiles to best respond to a question that was just posted in Community Question Answering. The findings show that CNN outperforms this experiment in comparison to other methods such as TF-IDF, LDA, Structural Topic Model.

In another study [29], a variation of CNN, Deep Cooperative Neural Networks (DeepCoNN), makes use of valuable information written in reviews of users and reviews for items for recommendation systems. DeepCoNN consists of 2 parallel neural networks. One of them is responsible for learning user behaviours by analyzing reviews written by users and other is responsible for learning items by analyzing reviews written for items. The experiments are conducted on 3 Datasets - Yelp (Restaurant Reviews), Amazon (Product Reviews) and Beer (Beer Reviews).

On the Yelp and Beer datasets, the suggested algorithm DeepCoNN produced gains of 8.5% and 7.6%, respectively. It exceeded all baselines on Amazon, improving by an average of 8.7%. The suggested model achieves an overall 8.3% improvement across the three datasets.

One variation of autoencoders is implemented in [30] where a model based on autoencoders is proposed to reduce the sparsity problem of Collaborative Filtering models. Here, Collaborative Filtering is converted to Supervised Learning. Its 5 main steps are: matrices creation (ratings of users for items are specified), data normalization (missing ratings of the matrix are filled with zeros), Autoencoder Architecture selection (adjusting the user item matrix so that it gets fit within the range of Autoencoders' activation function), generating supervised dataset (features of user and items are mapped) and applying regressor (regressor model is trained using the generated supervised learning data, for prediction purpose). Extraction, mapping and prediction are 3 main parts of this approach. Singular Value Decomposition (SVD) is used for the extraction part but it is unable to fetch the features that are not linear. So, to resolve this issue, Stacked Denoising Autoencoder is used. In the mapping phase, ratings of users for items are specified. In the prediction phase, supervised Learning is applied for making a model to learn. This approach also has a limitation - as the number of new users increases after a threshold, the quality of the system degrades. So, to determine this threshold is still a task of future work of this study.

A novel model Autorec is proposed in [31] which is based on an autoencoder framework for video recommendation. The authors contend that AutoRec is superior to the traditional approaches to CF in terms of representation and computational efficiency. The experiments are conducted on 2 datasets - MovieLens and Netflix. The proposed model outperforms the existing neural network model for Collaborative Filtering.

The problem of Trustworthiness of CFRS is highlighted in the study [32]. Trust Information of users is helpful for new users (cold start users) on social networks (Epinions and Flixster). To overcome this issue, a model called Deep Learning based

Matrix Factorization (DLMF) is proposed which is used for trust-based recommendations within social networks and surpasses the other cutting-edge techniques on social networks recommendations.

Another work in the field of social networks is implemented in [33]. In this work, autoencoders are used to tag users' profiles to extract deep features from them. In this way, user recommendation performance is increased by updating users' profile by this proposed method. This experiment is conducted on 2 website datasets - Last.fm and Del.icio.us. The proposed algorithm outperforms CF in terms of precision, recall and rank score.

One limitation of Collaborative Topic Regression is highlighted in [34] as the learned representation of items may not be effective. So, to overcome this issue, Stacked Denoising Autoencoder is used which is combined with Probabilistic Matrix Factorization to further extend it to Relational Stacked Denoising Autoencoder for improving the performance of tag recommendation. This work is implemented on 3 datasets: CiteULike-a, CiteULike-t and MovieLens.

Few existing video recommendation methods consider that users' interests are static. The work proposed in [35] resolves this problem by considering dynamic users' interests for videos. It uses RNN and considers three factors named: (1) Video semantic embedding which represents videos according to their content information, (2) User Interest modeling which represents users' choice of playing the video, (3) User Relevance Mining which provides additional attributes for improving the performance of recommendation. This work takes real time and dynamic interests of users’ on Google+ website - cross network dataset.

Application of RNNs along with GRUs (GRU4REC) are widely used in session-based recommendations. The combination of KNN with GRU4REC is proposed in the work of [36]. In an anonymous session, KNN is used to evaluate how well the next item is recommended. Two datasets are utilized in the work. First dataset are the music playlists from the platforms - 8tracks.com, artofthemix.org and last.fm. The second dataset is an open online shopping dataset from the TMall

competition. The results reveal that the combination of KNN with GRU4REC is effective in improving recommendation.

Recommendations are also done on long session based data instead of short term based sessions. The work proposed in [37] has used RNN combined with GRU for item-to-item recommendation in long term based sessions. A ranking loss function, mini-batch based output sampling and session-parallel mini-batches are added to GRU to increase its performance. The 2 datasets used in the work are - RecSys Challenge 2015 and YouTube like OTT video service platform. The proposed method outperforms all the baselines used earlier.

Another work [38] proposes hashtag recommendation. This work includes four steps: in the first step, skip-gram model generates word embeddings; the second and third phases deploys CNN for composition of sentences and RNN for composition of tweets respectively; and then finally in the fourth step, classification of hashtag is done. Traditional methods like SVM or TF-IDF ignore semantic related information in tweets aad thus making them inefficient. But the proposed model overcomes this drawback and improves the efficiency of the tag recommendation model.

A more robust model is proposed in [39] to tackle the problem of cold start for items. It involves the integration of Marginalized Denoising Auto Encoder (mDAE) along with Multi-View Recurrent Neural Network (MV-RNN). This integration comes out to be quite powerful in fetching the hidden patterns from textual data and images related to the items. The authors take 3 characteristics into account: fusion by reconstruction, fusion by addition and concatenation. The proposed model proves to be a great one for handling the problem of cold start for items.

The study's authors [40] use text and image data from microblogs to suggest hashtags on the Twitter dataset. They put forth a model of the co-attention mechanism that uses both images and text. Features are extracted from images using VGGNet. Text feature extraction is done via LSTM. A single-layer Softmax classifier is used to recommend tags after both are combined using a co-attention mechanism.

The method suggested in [41] is effective for suggesting tags on the Chinese microblogging platform "Sina Weibo." The microblogs are associated with news in the form of brief messages, and their content is categorized according to the five Ws: how, Where, When, What and Why. Following this, hashtags related to these five features are recommended. This work is divided into four sections: filtering spam from microblogs, dealing with common online terms and phrases, categorizing blogs into phrases, and lastly, recommendations of tags associated with the above five features. Authors link each word in a microblog to one of the five words, and then they suggest hashtags.

Authors of the study [42] develop an approach using LSTM which incorporates selective sentence-level attention for reducing noise. Additionally, it takes into account time-related data when recommending hashtags in SINA Weibo microblogs. After giving each sentence a weight, the noisy data is eliminated using selective sentence-level attention. Next, a Softmax pooling layer is used to recommend hashtags based on temporal information.

The study [43] proposes the Memory Augmented Co-attention model (MACON), for the hashtag recommendation task. Here, it is advised to use both text and image hashtags. Text and image features are learned using the LSTM, which is a co-attention neural network. Additionally, a memory unit is used to learn from users' tagging history. This is accomplished in two steps: firstly, a user-based random sampling of past posts by users along with the hashtags associated with those posts is performed; secondly, users' historical posts are used to learn about their tagging habits, and the current post is connected to a new tag. In the future of this study, user-based temporal sampling or community-based random sampling (which takes into account the posts of users' friends) may be investigated.

The authors of the study [44] present DeepTagRec, a deep learning model that leverages a heterogeneous network of user-tag relationships in addition to the information contained in the body and title of StackOverflow's questions. This information is represented using the Gated Recurrent Units (GRU) model. GRU produces the encoding of this information into a sequence of words using word2vec.

In heterogeneous networks, the modeling of user-tag connections is captured by the node2vec model. Tag prediction and recommendation are achieved by concatenating word2vec and node2vec.

In a different study [45], Graph Convolutional Network (GCN) is used to recommend personalized hashtags by analyzing the graph involving interactions among three nodes: micro-videos, tags and the user. Additionally, it makes use of a mechanism for attention to filter out unnecessary information that hashtags and users receive from micro-videos. This model takes into account the users' tastes for publishing content and their individual knowledge of hashtags. The publicly available YFCC100M dataset and Instagram dataset are used for the experiments. Compared to previous works that are mentioned in this study, one limitation of this work is that there is no significant improvement in accuracy even after filtering out the noise.

The authors of the study [46] present a novel method for hashtag recommendation of Vine micro-videos. They propose a model for senTiment enhanced multi-mOdal Attentive hashtag recommendation (TOAST). It considers content features and sentiment from three distinct multimodalities—text, audio, and video. For extracting sentiment features, Multi-Layer Perceptrons (MLP) are utilized, while Bi-directional LSTMs (Bi-LSTMs) are employed for extracting content features for every modality (text, audio, and video). Then, hashtag recommendations are made using the suggested model, TOAST. This work provides a path forward for sentiment hashtag recommendations in text modality by integrating emojis.

In a different study [47], the authors recommend tags for Musical.ly micro-videos. This recommendation is based on profiles of users and their past hashtags history. The authors propose the model - User Guided Hierarchical Multi-Head Attention Network (UHMAN), which integrates individual profiles and their past hashtags history. This model is employed to pay attention to micro-video representations, including customer profiles, at both the image and video levels.

A novel recommendation framework called Contextualized Graph Attention Network (CGAT) is proposed in [48] for an entity in Knowledge Graph. It utilizes an entity's graph context data, both local and non-local. To extract the local context

data, it makes advantage of network attention methods related to users. Additionally, by extracting the entity's non-local context using the skewed randomly generated sampling approach and an RNN, the relationship within an entity and its distant contextual counterparts is modeled by CGAT. To record the individual preferences of the user for certain items, a mechanism is drawn to focus on item-specific attributes. This approach simulates the connection of a target item with the related items retrieved from the user's previous actions.

When compared to other traditional products (such as literature, entertainment, and groceries), travel-related products are typically visited due to the time and cost involved. Also the decision for selecting the right travel product gets affected by the arrival and departure times, location, and cost. Thus, to resolve the above mentioned issues, [49] proposes a model, Multi View Graph Attention Network (MV-GAN) for Travel Recommendation. To analyze the product and user representations from all perspectives, attention networks are constructed at the vertex and edge hierarchies.

The Point-of-Interest Recommendation System (POI-RS) seeks to uncover potential venues that users may find appealing. Federated Learning (FL) is introduced into POI-RS for privacy protection in numerous works. However, they are unable to ensure recommendation performance due to limited data in POI-RS. Furthermore, model training in FL is easily rendered ineffective by check-in geographic factors. Therefore, the study [50] suggests a novel sequential information-based (FedSR) framework for POI-RS in order to address the above mentioned issues. A multi-task framework in the FedSR is constructed using Contrastive Learning and uses a data augmentation method based on spatial relationships among POIs. For the recommendation task, Bayesian Personalized Ranking (BPR) optimization is applied.

Regarding the social endorsement task, which is challenging because of its specifications for privacy protection, personalization and heterogeneity, the authors of the study [51] design a federated learning recommender system. ‘Federated Social recommendation with Graph Neural Network (FeSoG)’ is a novel model

proposed by them in order to achieve this goal. To address heterogeneity, FeSoG first uses relational attention and aggregation. Second, in order to maintain personalization, FeSoG uses local data to infer user embeddings. Finally, to improve training and preserve privacy, the suggested model uses item sampling in conjunction with pseudo-labeling techniques.

Location-based social networks (LBSNs), which are essential for proposing the next Point-of-Interest (POI), have become widely popular as a result of the growth of location-based services. For the next POI recommendation, the study [52] presents an attention-based fusion framework and a modified node2Vec. Using a modified node2vec algorithm, first the raw data is preprocessed to extract the pertinent information before presenting the feature vectors for users and locations. The attention-based framework is then applied to these feature vectors. These features are then used to produce balanced and properly labeled datasets that are categorized according to predetermined time intervals. These datasets are then used to train different classifiers, which are then combined in a weighted way to create a better recommendation system based on fusion.

Most of the studies referenced in this literature review have not discussed about the sampling and noise filtering methods. So, we have done an exploration of various sampling and noise filtration methods to improve the results in the next chapter. An implementation is also done for movie recommendations using suitable sampling method and an optimization method for noise filtering in chapter 4. Most of the studies include methods to capture the semantics of the words in a sentence. But, the context of the word is not taken care of. So, our deep learning based approach for hashtag recommendation has taken care of this gap in chapter 5. Also, most of the studies have evaluated their results using the commonly used evaluation metrics such as Recall, Precision and F1-Score to measure the accuracy of their models. Other accuracy measurement metrics are explored in chapter 6 so as to know which metric is best applicable in which situation.

CHAPTER 3

SAMPLING AND NOISE FILTERING METHODS

Many e-commerce sites that make product recommendations based on user interests and needs have emerged in the age of online commerce. There is an abundance of data that can be analyzed to improve the effectiveness and efficiency of recommender systems. However, processing the complete dataset takes a long time. Thus, a portion of the data must be chosen in order for it to be processed quickly. Sampling is a technique used to choose a portion of the dataset that includes every attribute that the database is capable of storing. Knowing which kind of sampling technique is best for a given recommender system application is crucial. As a result, research into different sampling techniques previously applied to recommender systems is necessary.

Additionally, since the data contains some noise, we must clean it up before processing it. The system can be subtly programmed with malicious noise to change its behaviour. Also, because users are reluctant to assign accurate ratings to the items, natural noise inadvertently finds its way into the systems. Apart from malicious and natural noise, there exists structural and contextual noise as well. Therefore, in order to prepare the data for processing, these kinds of noise must be filtered out.

3.1 Introduction

It takes a lot of work to process all the data for recommendation systems. However, handling a portion of it will simplify and ease matters. Sampling is a technique where an analysis is carried out by selecting a subset of observations from a given set of all the observations. Sampling in research is the process of taking a subset of relevant data from the whole dataset. It gives us a way to run our experiments on a subset of the data instead of the full dataset, as computing on larger datasets will needlessly increase computational time and cost. Thus, a sample that is adequate for the experiments to be carried out is taken from the dataset based on the requirements.

In this chapter, six sampling techniques have been examined: Gibbs sampling, Thompson sampling, Bernoulli sampling, Negative sampling, Bayesian Hierarchical sampling, and Bootstrap sampling.

When choosing a sample from a population, there may be some noise in the sample. Thus, we must filter out noise from the gathered dataset. Therefore, research on noise filtration techniques for recommendation systems is necessary. Recommender Systems suggest related products to users as they peruse the products of their choice. Certain investors or manufacturers would prefer that their goods receive higher user ratings than those of other brands. Therefore, they might add fictitious profiles and rate their own products, raising the product's rating. Additionally, the high ratings for this product may fool a sincere customer. However, users have the ability to rate items arbitrarily. This adds additional types of noise to the database.

O'Mahony [53] explains that noise is divided into two categories: malicious noise and natural noise. The inconsistent ways in which users rate or comment on the products they have bought or found appealing give rise to naturally occurring noise. The noise that subtly introduces bias into the system is known as malicious noise. Recommender systems are easily susceptible to malicious attacks because of their open nature. A malicious noise is added with the intention of raising (Push attack) or demonizing (Nuke attack) a particular product. This type of noise involves inserting biased profiles into a database in order to change how recommender systems behave. Thus, we provide an overview of the studies related to these noise.

Apart from these two types of noise, there exists structural and contextual noise as well. Structural Noise refers to the inconsistencies and irregularities in the data structure or format. Contextual Noise refers to the dynamic needs and behavior of the user due to the change in the context like time, location and moods. In order to handle these types of noise, advanced techniques such as optimization algorithms are applied to the recommendation systems. Optimization improves the quality of the recommendation systems. It focusses on refining the performance of RS by reducing the impact of noise through mathematical and computational methods. Thus, we provide an overview of a few studies that use optimization techniques to

reduce/eliminate the impact of noise while measuring the performance of the recommender systems.

3.2 Sampling Methods

This section offers a review of studies that discuss different recommender system sampling techniques.

3.2.1 Bayesian Hierarchical Sampling

Hierarchical Sampling is employed in observational situations when data is collected at many spatial scales. It overcomes the limitations of clustering algorithms, such as their inability to handle huge datasets and high sensitivity to noise. Hierarchical Sampling is suitable for large datasets and is noise resistant [54].

Tourist destinations have been recommended using Hierarchical Sampling [55,56]. In these implied works, user preferences for various demographic attributes are obtained. These attributes are "Travel Season", "Travel Interest", and "Travel Method". [55] employs Bayesian Personalized Ranking (BPR) to find the semantic aspects of several images. Then, a novel model is created by combining BPR with Hierarchical Sampling. The authors of [56] predicted user ratings using SVD++. The combination of Hierarchical Sampling and SVD++ is then used to recommend tourism attractions.

A further approach suggested in [57] states that the system uses knowledge from other users to recommend products to a new user via a Bayesian Hierarchical Sampling model. In this study, the weights are sampled at random for each user, and the ratings are sampled at random for each item using the Normal distribution. The experiments are carried out on datasets from MovieLens, Netflix and Reuters.

3.2.2 Negative Sampling

It is critical to train the model with both positive and negative examples in user-item recommendation systems [58]. Positive examples can be easily gathered through the user's interaction with the products. Negative examples are products that the user does not interact with. As a result, the method of gathering negative examples is known as negative sampling.

[59] shows that negative sampling was important in recommending social friends and foes using the Social Pairwise Deep Learning model's social ranking method. The authors of this work use negative sampling in the back propagation step to optimize the loss function for selecting unobserved items (negative examples) of users' trust and distrust. [60] proposes yet another work on user-item representations using negative sampling. It proposes a contrastive learning module based on GNN to learn user item representations in a self-supervised manner.

Recent research has also used negative sampling in recommender systems [61-65]. The authors of [61] propose a novel Contrastive Cross-Domain Recommendation (CCDR) model that improves traditional Cross-Domain Recommendation (CDR) systems. CDR has a data sparseness limitation in the candidate generation phase, which is overcome by CCDR. With the help of negative sampling, this paper also introduces both intra and inter domain contrastive learning.

A new model for a heterogeneous multi-domain recommendation system is proposed in another study[62]. The model in this work uses negative sampling to retrieve heterogeneous data from multiple recommender system source domains. The authors of [63] apply two-tower frameworks based on neural networks using a variation of negative sampling known as Mixed Negative Sampling (MNS). MNS leverages item recommendation systems by using batch and uniform negative samples. The results show that MNS outperforms the other baseline models.

Conversational recommender systems also use negative sampling [64]. In this work, the pre-training step combines both item-based (historical data) and attribute-based

(conversational data) preference sequences. The negative sampler, which generates high quality negative samples, later improves learning performance.

Another method [65] employs Matrix Factorization (MF) in collaborative filtering recommender systems to represent latent user and item features in a shared feature space. Neural Embedding Collaborative Filtering is a model that combines MF and neural embedding. Auto-encoders are used in this model to generate embedding vectors. Negative sampling is used to represent latent features in a regression model that is combined with these vectors. The inner product is then applied to user-item latent features to define their correlations. This model is tested on MovieLens and Pinterest datasets, and the results show that the system is quite accurate.

3.2.3 Thompson Sampling

Thompson sampling is used to solve the Multi-Armed Bandit (MAB) problem. According to the MAB problem [66], if there are N machines and the user has to find the machine with the best reward, how would the user do that? Although there are a number of algorithms [67] to solve the MAB problem, such as the Epsilon Greedy Approach, Boltzmann Exploration, Pursuit Algorithms, and Upper Confidence Bounds, Thompson Sampling outperforms the others [68]. Posterior sampling is another name for Thompson sampling. It is a Bayesian algorithm that is randomized. It is used to select the model based on the likelihood of it receiving the best rewards [69].

Occasionally, recommender systems are incapable of capturing users' dynamic contexts. As a result, the authors of [70] devise a novel interactive recommendation system capable of capturing and presenting users' dynamic behavioral context. Thompson sampling is used in this case, and rewards (feedback) are collected after each interaction of the user with the system. These rewards are used as an input to determine the user's next preferred model.

Although recommender systems use Click Through Rate (CTR), they still suffer from CTR underestimation due to changing item ratings [71]. As a result, bandit solutions are used to solve this problem. Thompson Sampling is used in this work,

which results in greater accuracy. Conversational recommender systems use bandit solutions such as Thompson Sampling to interact with users via natural language [72]. Such systems ask users whether they like or dislike a particular item. As a result, the systems are highly responsive and dynamic.

Recommender systems also employ Thompson Sampling to ascertain the impact of offline parameters on online performance. The results show that the influence of offline parameters on online performance diminishes with time [73]. Position Based Metropolis-Hastings Bandit (PB-MHB), a unique bandit-based solution that makes use of the Thompson Sampling framework, is proposed in another work [74]. This framework is used to present different items on a web page at appropriate places. The outcomes demonstrate that this approach enhances recommendation performance.

3.2.4 Bernoulli Distribution Sampling

One particular type of poisson sampling is called Bernoulli sampling. In Poisson Sampling, the odds of selecting each item can vary, but every item in Bernoulli Sampling has an equal chance of being chosen. Bernoulli distribution sampling has been used in the study of several recommendation systems. Genre information [75], which employs Bernoulli distribution sampling and takes into consideration three important properties: genre coverage, genre redundancy, and size awareness, increases the diversity of recommendation systems.

Enhancing the ranking of recommendation systems is another use for the Bernoulli distribution [76]. This work proposes a Deep Generative Ranking (DGR) for this purpose. Using the Bernoulli distribution, DGR generates feedback and produces a ranking list for each user's interacted and non-interacted items. A different study [77] looks at how dynamically people interact with recommendation systems. In order to address the heterogeneous user preferences that emerge during user interaction with the recommendation systems, this work also employs the Bernoulli distribution.

A novel method called Bernoulli Matrix factorization was presented in [78] to give recommender systems prediction and reliability. It has a number of advantages over the other methods currently in use, including the fact that it is based on classification

rather than regression models and does not rely on outside sources for reliability. The MovieLens, FilmTrust, and MyAnimeList datasets are used to evaluate it, and the findings show that it provides improved reliability.

Bernoulli sampling is also used in [79], where memory-intensive embedding representations in recommender systems are replaced with mixed dimension embeddings to save space and memory. This mixed dimension embedding shortens the training period while increasing efficiency. Bernoulli Distribution sampling also promotes long-term user engagement with recommender systems [80]. This method carefully monitors user clicks (both current and future clicks) and return behaviors for the optimization purpose. Experiments on Yahoo News demonstrate the effectiveness of the suggested strategy.

3.2.5 Gibbs Sampling

One variation of the Bayesian sampling method is Gibbs Sampling. Recommendation engines play a significant role in e-commerce by offering products based on user preferences. However, they have issues with cold start and sparsity. In [81], a novel approach is proposed to circumvent such issues by combining social relations with user-rated items. The user and item feature vectors are sampled using Gibbs Sampling in this method. When compared to other baseline models, the results show that this fusion produces recommendation results that are more accurate.

Another method, the Bernoulli Restricted Boltzmann Machine (BRBM) [82], creates Joke-Reader Segmentation based on the preference patterns and recommends jokes using Gibbs sampling. BRBM has 100 visible nodes and 20 hidden nodes in this method. Gibbs sampling determines the values of hidden nodes in the first step, resulting in a vector size of 20. The values of the visible nodes are computed using Gibbs sampling in the second step, resulting in a vector size of 100. This procedure, known as 20-step Contrastive Divergence, is carried out 20 times.

Gibbs sampling is also suitable for handling streaming data and big data in real-world recommender systems [83]. The authors of this study propose a novel approach, Online Bayesian Inference for Collaborative Topic Regression. In order to handle flowing information and large amounts of data in practical recommender systems, this method applies Gibbs sampling.

3.2.6 Bootstrap Sampling

Another sampling technique known as bootstrap sampling, allows for the selection of an object once and again in the future. This method has become increasingly important in deep learning recently. Medical recommendation system [84] that uses Fourier Transformations to predict heart diseases is one area in which bootstrap sampling is applied. In this case, numerous training datasets are created via bootstrap sampling, and the necessary prediction is then achieved by applying three algorithms—Support Vector Machines, Artificial Neural Networks and Naive Bayes—to these generated datasets.

Decision Trees are used by more medical recommendation systems based on Chronic Disease Diagnosis [85,86] to estimate the disease's risk and make recommendations correspondingly. Decision Trees combine to create a Random Forest, and each node's features are chosen using Bootstrap Sampling. One can generate an unbiased estimate of the classification error by utilizing Bootstrap sampling. The medical records of historical patients from the Middle East are used to evaluate this work.

Web-based Bootstrap Recommendation systems can function more efficiently when sampling is used [87]. Personalized recommendation systems can occasionally malfunction due to intermittent changes in user and item repositories as well as a cold-start issue. The results of this study's experiments on online advertising and news recommendation show that employing a bandit strategy boosts the effectiveness of recommender systems.

Bootstrap Sampling is also utilized by ensemble techniques [88,89]. In the proposed work [88], online bagging—an ensemble technique—is used to give the recommender systems a dynamic and responsive quality. Recommender systems

must be quick enough to suggest items based on the dynamic nature of users since large amounts of data are available. In order to train the model for this purpose, bootstrap samples are used in an online bagging method.

An additional work that uses an ensemble approach and is based on ordered item sequences was suggested in [89]. Using this method, one can create ordered sequences of novelty and popular items by comparing the rating patterns of attackers and authentic profiles. After that, each user's item rating series is created. This method makes use of bootstrap sampling techniques to create training sets. This training set is used to train the decision tree so that it can eventually identify the phony profiles.

3.2.7 Comparison of all Sampling Methods

Table 3.1 shows strengths and limitations of the above discussed sampling techniques.

Table 3.1: Strengths and Limitations of the above discussed Sampling Techniques.

Sampling Methods	**Strengths**	**Limitations**
Bayesian Hierarchical Sampling	It models complex dependencies between users and items.	It takes time to converge and thus makes inaccurate or unreliable recommendations.
Negative Sampling	It helps to reduce noise by separating positive (relevant) and negative (irrelevant) examples and thus makes accurate recommendations.	It is less effective for infrequent words or sparse datasets.

Thompson Sampling	Balances exploration (trying out new items) exploitation (recommending known items) trade off.	It is computationally expensive in comparison to other simpler heuristic-based approaches.
Bernoulli Distribution sampling	It is used where feedback is binary like whether the user liked / clicked / purchased a recommended item or not.	Difficult to manage exploration - exploitation tradeoff.
Gibbs Sampling	For sufficient iterations, it converges to the true posterior distribution of the model parameters and thus makes accurate recommendations.	It is hard to work with Gibbs Sampling when variables have strong dependencies among them, as it takes a long convergence time when the data is huge.
Bootstrap Sampling	It is simpler in function and predicts accurate results.	It is ineffective for small datasets.

3.3 Noise Filtering Methods

Noise is mainly classified as Natural, Malicious, Structural and Contextual [53, 156, 157]. Natural Noise arises because of unintentional wrong ratings of the users for the rated products, whereas malicious noise is an intentional attempt to make the recommender system biased by inserting false user profiles to biased ratings for specific products. Structural Noise refers to the inconsistencies and irregularities in the data structure or format. Contextual Noise refers to the dynamic needs and behavior of the user due to the change in the context like time, location and moods. Fig.3.1 shows the different methods handling different noises. We provide a brief survey of various noise filtering methods on recommendation systems in the below sections.

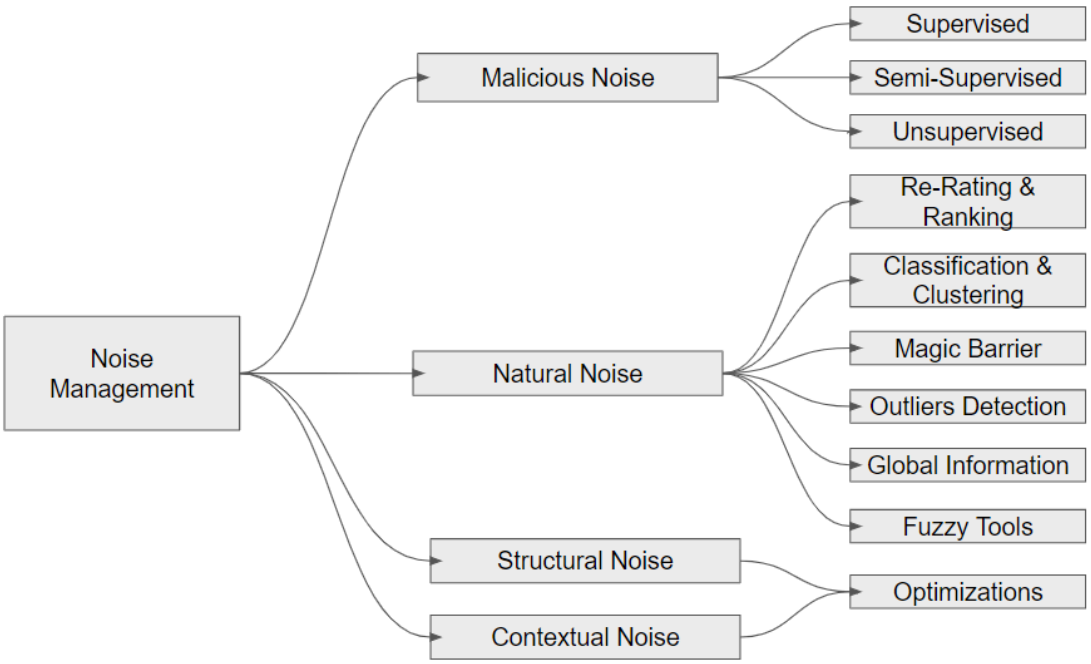

Fig.3.1: Different methods handling different noises

3.4 Malicious Noise

Several methods have been developed to identify malicious noise / shilling attacks. We have reviewed some studies on supervised [89-99], semi-supervised [100-104], and unsupervised [105-115] methods for identifying this kind of noise. While unsupervised approaches involve different kinds of clustering, supervised approaches primarily involve different kinds of classification, and semi-supervised approaches combine elements of both supervised and unsupervised methods.

3.4.1 Supervised Methods

Because recommender systems are open, they are susceptible to attack / malicious noise. In order to identify and eliminate this noise, recommender system research is done. Burke et al. [90] provide one such study in which a number of attributes that can be obtained from the user profile are examined. This study demonstrates that classification techniques can be considered superior to other generalized detection models when used with attributes collected from the profiles of attackers. According to [91], user influence is an additional component. Instead of using proven attack

detection techniques to the entire user population, their approach focuses on influential people. This study also demonstrates the connection between the attacker and the user's susceptibility to persuasion.

A number of criteria were put forth by Chirita et al. [92] to assess rating patterns and identify malicious noise. The authors forecast the likelihood that a user will be an attacker based on the following metrics:

1. The standard deviation of a user's rating - It is the variation between the user's rating and his average rating for a specific item.
2. Degree of Agreement with Other Users - It is defined as the departure of a user's ratings from the average ratings of each item.
3. Average Similarity - The degree of similarity between a user's ratings and the top K nearest neighbors is the average similarity.
4. Number of Prediction Differences - It is defined as the number of prediction changes of the system for each user if the user is removed from the system.
5. Rating Deviation from Mean Agreement (RDMA) - This metric measures how users agree or disagree with a collection of target items and the inverse rating frequency of those items.

The primary goal of a different suggested technique [93] is to enhance the identification of Average over Popular Items (AoP) attacks. This method extracts the AoP attack features using the Term Frequency Inverse Document Frequency (TF-IDF). Subsequently, the SVM model is trained on the training set to produce SVM-based classifiers. This classifier is used to identify AoP attacks.

Another supervised method that makes use of the Random Forest Classifier to identify malicious noise is suggested in [94]. There are three stages to this work. Attribute extraction is covered in the first stage. Training and test datasets are used to create a classifier in the second stage. Using the classifier created in the second stage, the final detection of bogus profiles is carried out in the third stage. The MovieLens dataset is used to assess this method.

Three classification techniques—k nearest neighbor (kNN), C4.5, and SVM—are used in another supervised approach [95] to find the malicious noise. With this

method, the qualities that characterize fraudulent profiles are identified. The SVM classifier performs best with this model. Moreover, the MovieLens dataset is used for the experimentation purpose.

In another work [96], user-user and user-item interactions are used to extract features from phony profiles even if the attacker modifies their attack strategy. This method considers label information in addition to user-user and user-item interactions. The Bayesian model, which is added for feature learning, uses the label information to provide user implicit features. The Amazon and MovieLens datasets are used to test this model.

HHT-SVM, a technique for detecting online malicious attacks, is presented in [97]. It solves the issue of profile injection attack detection in batch mode, which calls for processing the complete ratings database. It combines the Support Vector Machine (SVM) with the Hilbert-Huang Transform (HHT). Initially, this model generates the ratings series for each rated item for a specific user. Then, features based on the Hilbert spectrum are extracted for malicious profiles using Empirical Mode Decomposition (EMD). Then, using the features that were extracted, SVM is used to identify profile injection attacks. The system performs well, according to the evaluation on the MovieLens dataset.

The issue of class imbalance in attack detection algorithms occasionally affects SVM [98]. Therefore, in order to address the issue of class unbalance in imbalance datasets, a method called SVM-TIA (Target-Item Analysis), utilizes a novel over-sampling technique called borderline-SMOTE. The suggested method takes the rating matrix and pulls out the attributes. The procedure is then split into two stages. Using the attributes, the classifier created by the Borderline-SMOTE in the first phase finds suspicious profiles. Subsequently, in the next stage, the suspicious profiles employ the rating matrix to scrutinize the target items, and ultimately, the outcomes are identified. A higher precision rate is found when using this method on the MovieLens dataset.

In [99], an additional supervised method for collaborative filtering recommender systems is put forth. On the basis of the attack model, classification models which are

used in feature extraction, extracts eighteen features from user profiles. Re-scale boosting algorithms (RBoosting) and AdaBoost are the strategies employed here. For the purpose of identifying malicious profiles, this work employs RAdaBoost, a re-scaled version of AdaBoost. The MovieLens dataset is used for experiments, and the results show that AdaBoost enhances system performance when compared to more conventional classification models like SVM, kNN, and others.

In [89], a novel ensemble approach (EMDSA-OIS) is proposed to detect malicious noise based on ordered item sequences. This method examines the attackers' and real profiles' rating patterns. The authors of this work are able to understand the difference through which ordered item sequences are generated. After that, each user's item rating series is created. Additionally, this method suggests six characteristics to describe the attackers. The rating series of the items is first used to extract two features, from which mutual information is combined to extract the remaining four features. This method makes use of bootstrap sampling techniques to create training sets. This training set is used to train the decision tree so that it can eventually identify the phony profiles.

All of the supervised techniques covered above are compiled in Table 3.2, which also displays the various datasets used to test the noise filtering techniques and the evaluation metrics.

Table 3.2: Summary of all the discussed supervised approaches

S.No	**Publications**	**Noise filtering methods**	**Datasets**	**Accuracy measurement tools**
1.	[90]	Classification Methods	MovieLens	Precision, Recall
2.	[91]	KNN	MovieLens	Precision, Recall
3.	[92]	Metrics for detecting rating patterns	MovieLens	Rating Deviation from Mean Agreement (RDMA)

4.	[93]	SVM	MovieLens	Precision, Recall
5.	[94]	Random Forest Classifier	MovieLens	Precision, Recall
6.	[95]	kNN, C4.5 and SVM	MovieLens	Mean Absolute Error (MAE)
7.	[96]	Bayesian model	Amazon & MovieLens	Precision, Recall, F1-measure
8.	[97]	HHT-SVM	MovieLens	Specificity, Sensitivity, and Precision
9.	[98]	SVM-TIA	MovieLens	Precision, Recall
10.	[99]	RAdaBoost	MovieLens	Classification Error, Detection Rate, False Alarm Rate
11.	[89]	Decision tree	MovieLens	Precision, Recall, F1-measure

3.4.2 Semi-supervised Methods

Semi-supervised methods have the benefit of utilizing unlabeled data found in recommender systems. Semi-supervised Shilling Attack Detection, or Semi-SAD, is one such semi-supervised method that is used in [100]. This method uses labeled as well as unlabelled user profiles. It utilizes an augmented Expectation Maximization, or EM-λ, on unlabeled profiles after training the Naive Bayes classifier model on labeled profiles. The trained model performs better with this method. It is tested using the MovieLens dataset and contrasted with both supervised and unsupervised models. The findings show that compared to supervised and unsupervised models, this semi-supervised method is more effective at identifying shilling attacks.

Zhang et al.'s [101] group detection approach is based on Semi-Supervised Learning using Spammer Group Detection (Semi-SGD). This method utilizes data that has not been labeled. This method uses the Expected Maximization (EM) algorithm to incorporate unlabeled data after the Naive Bayes classifier has been trained on labeled data. This methodology enhances spammer group detection techniques and is assessed using datasets from Amazon.cn.

Another study [102] utilizes unlabeled data to increase recommender system attack detection precision rate. A variation of the Co-Forest algorithm (introduced in [103]) is proposed, called Semi-Supervised Attack Detection in Recommendation based on Co-Forest Algorithm (SSADR-CoF). Rather than using a small number of features to train a single classifier, this model trains multiple classifiers using a series of features. Two models for rating behavior and window splitting, are used to extract these features related to user rating behavior. First, a set of classifiers is trained with labeled user profiles using these extracted features. After that, unlabeled user profiles are labeled using these classifiers. Subsequently, the classifiers are updated to increase accuracy using the user profiles that have already been labeled and those that have not.

Gaussian Mixture Model (GMM) and Modified Support Vector Machine (MSVM) are used in another semi-supervised study that was presented in [104]. This lowers the dimensionality of the data and identifies the malicious noise on the MovieLens dataset. There are two stages to this proposed project. First, the rating matrix is used to analyze both real and fake user profiles. Next, attributes are extracted. After analyzing these attributes, MSVM generates a Classifier Generation to identify potentially suspicious profiles. In order to identify the last group of phony profiles, the suspicious profile set is examined using GMM in the second phase.

All of the semi-supervised techniques covered above are compiled in Table 3.3. It also includes the evaluation metrics and various datasets used to test the noise filtering techniques that each paper has identified above.

Table 3.3: Summary of all the discussed semi-supervised approaches

S.No	Publications	Noise filtering methods	Datasets	Accuracy measurement tools
1.	[100]	Semi-SAD	MovieLens	Specificity, Sensitivity
2.	[101]	Semi-SGD	Amazon	Precision, Recall and F1-Measure
3.	[102]	SSADR-CoF	Amazon & MovieLens	Precision, Recall, AUC (Area Under ROC Curve)
4.	[104]	Modified SVM, Gaussian Mixture Model (GMM)	MovieLens	Precision, Recall

3.4.3 Unsupervised Methods

Utilizing an unsupervised method using the Hidden Markov Model (HMM) and Hierarchical Clustering is one way to identify malicious users for shilling attacks [105]. This method uses HMM to calculate the degree of suspicion after analyzing each user's rating behaviors. After that, users are grouped according to their suspicions using hierarchical clustering, which helps to identify the malicious users. Experiments conducted on the MovieLens and Netflix datasets demonstrate a significant improvement in the accuracy measure.

In [106], an additional unsupervised method is suggested that makes use of the co-clustering with propensity similarity model (CCPS). In order to identify shilling attacks, CCPS is a soft co-clustering technique that uses a user propensity similarity method. This method is experimented on the MovieLens and Jester datasets.

Item Relationship Mining - Target Item Analysis (IRM-TIA) [107], an alternative method, focuses on detection issues in real-world unlabeled datasets. There are three

phases to this work, which is based on target items and the item relationships matrix. Features are taken out of the item relationship matrix in the first stage. Then, using the features that were extracted in the first step, a group of suspicious users is created in the second phase. Subsequently, the third stage involves pinpointing the target items through an analysis of the actions of dubious users in relation to these items, and lastly, identifying the attackers from this dubious group. The MovieLens and Amazon reviews datasets are used for experiments, and the findings demonstrate the success of this strategy at identifying the attackers.

Malicious attack outliers are also found in the study [108]. The authors of this study employ a categorization method that specifies the characteristics to ascertain if a user is authentic or fraudulent. After that, they cluster the dataset into authorized and unauthorized users using the k-means clustering algorithm to find outliers. Push and nuke attacks were also used by the authors to assess their work on the Epinions dataset. The outcomes demonstrate the high accuracy of their work.

An unsupervised three-stage method is proposed by the authors of [109] to identify abnormal ratings in collaborative filtering recommender systems. Using user profiles as a guide, an undirected user-user graph is first constructed. Graph mining techniques are used in the second stage to find user similarities in order to optimize the graph. In order to distinguish a portion of real users from suspicious ones, an analogous analysis is also performed on the optimized graph. In order to identify the phony profiles, the remaining real users are then further weeded out in the third stage by examining target items. The MovieLens dataset is used for experiments, and the findings demonstrate the effectiveness of this strategy in comparison to other approaches discussed in this paper.

Another unsupervised method [110] uses special ratings, user activity and item popularity to eliminate ratings that are too sparse. To find the attackers, Target Item Analysis is then coupled with clustering on the remaining ratings. In [111], an alternative method for identifying outliers in e-commerce recommender systems is put forth. To find outliers, it employs two methods: the Clustering-based Partition Around Median (PAM) algorithm and K-Nearest Neighbors. The MovieLens dataset

is taken into account when assessing this work. Based on the findings, kNN provides significantly more accurate results than the PAM method.

In [112], one method for group recommendations is presented. A group of items is the focus of group recommendation as opposed to a single item. This strategy suggests using an unsupervised detection technique called De-TIA (Target Item Analysis) to find unusual profiles in group recommendations. It makes use of the Degree of Similarity (DegSim) metric, which calculates how dissimilar normal and aberrant profiles are from one another. The MovieLens, Netflix, and Eachmovie datasets are used to assess this method. The outcomes demonstrate the effectiveness of this work in enhancing the detection procedure.

Hurried Attacks are a new kind of shilling attacks. In these attacks, phony user profiles are quickly created in order to assign random ratings to the products. According to C. Panagiotakis et al. [113], the Hurried Attack's outliers are eliminated. The user-item rating matrix and user similarity serve as the foundation for this suggested system. There are three steps to this method. Initially, sparse entries—that is, users who rate relatively few items and items rated by relatively few users—are eliminated. Four characteristics are taken into account in the second step to distinguish the profiles of abnormal and real users. The user-item rating matrix and synthetic coordinates of the Social Collaborative Recommendation system (SCoR), (introduced in [114]), are used to compute these attributes. The K-means clustering algorithm, an unsupervised method, is then used in the third step to find the malicious profiles. The MovieLens dataset, which is used in this work, demonstrates the high performance that the suggested method provides.

A new feature called Randomness in Item Selection (RIS) is introduced in another approach [115], to identify malicious users and abnormal profiles. This approach suggests three approaches. The first approach calculates the likelihood that a user is malevolent using an unsupervised method. The second strategy also uses an unsupervised K-means clustering algorithm to select malicious profiles on its own. The third approach—a supervised technique based on random forest—is suggested

on the labeled dataset. Experiments are conducted using the MovieLens and Netflix datasets, demonstrating the excellent performance of the suggested techniques.

Table 3.4 provides an overview of all the unsupervised techniques covered above. It displays the various datasets that are used to test the noise filtering techniques and the evaluation metrics.

Table 3.4: Summary of all the discussed unsupervised approaches

S.No	**Publications**	**Noise filtering methods**	**Datasets**	**Accuracy measurement tools**
1.	[105]	HMM, Hierarchical clustering	MovieLens & Netflix Datasets	Precision, Recall, F1-measure
2.	[106]	Soft co-clustering	MovieLens & Jester Datasets	Mean Absolute Error (MAE)
3.	[107]	Item relationship and target item(s) -TIA	MovieLens & Amazon Review Datasets	Precision, Recall
4.	[108]	Classification + k-means Clustering	Epinions Dataset	Precision, Recall
5.	[109]	Graph mining	MovieLens	Detection Rate, False Alarm Rate
6.	[110]	TIA-clustering	MovieLens, Amazon & TripAdvisor Datasets	Accuracy, Normalized Mutual Information (NMI), Purity, Detection Rate and False Alarm Rate
7.	[111]	kNN and Clustering based PAM	MovieLens	Accuracy

8.	[112]	De-TIA	MovieLens, Netflix & Eachmovie Datasets	Detection Rate, False Positive Rate
9.	[113]	Unlabeled Data - K-Means clustering, Labeled Data - Random Forest Classifier	MovieLens	Precision, Recall, F1-measure
10.	[115]	Unlabeled Data - K-Means clustering, Labeled Data - Random Forest Classifier	MovieLens & Netflix Datasets	Precision, Recall, Specificity

3.5 Natural Noise

Users unintentionally giving their items incorrect or false ratings causes natural noise to appear in the database. This occurs as a result of users clicking without reading the appropriate feedback questions because they are unwilling to spend the time providing feedback. Since it is not entered with the intention of skewing the system, it differs from malicious noise. The noise that enters the system is an inadvertent act. To deal with the natural noise, various strategies have been put forth. A review of studies deploying various strategies on noise management has been done in subsequent sections.

3.5.1 Crisp Management

A review of the literature on the papers pertaining to Crisp Management Techniques is given in this section.

3.5.1.1 Re-Rating & Ranking

The idea that using user ratings as ground truth to predict unknown ratings of items is strongly rejected by Amatriain et al. [116]. The authors claim that users may be inconsistent in their ratings and comments. In order to quantify the noise in user ratings that could be the result of inconsistencies, they present a study on a dataset of movie reviews. They examine the ratings that users have left for various items over the course of one to fifteen days. The findings indicate that users' ratings are inconsistent because their personal preferences fluctuate over time.

Re-rating is another method for reducing natural noise in recommendation databases [117]. Re-rating is a technique that asks users to rate their previously rated items (items that they either liked or bought) one more time. Optimal ratings are chosen to be re-rated because it is not feasible to force every user to re-rate every item. This greatly increases the accuracy of the recommendations.

A different strategy [118] uses interactive recommender systems, letting users adjust their own ratings to enhance MovieLens and Netflix Datasets' functionality. A new rating is predicted by O'Mahony et al. [53] for every user-item pair. To identify noisy ratings, this rating is contrasted with ratings that were previously collected from the user through feedback. This method uses a set of real user profiles along with a memory-based collaborative filtering technique to detect noise.

To enhance recommendation performance, the study [119] suggests a model called Dual Training Error based Correction approach (DTEC). Its foundation lies in fixing users' and training set items' mistakes. It is applied to the test set once the error has been reduced to zero. The outcomes demonstrate that the recommendations were more accurate.

A different study, referenced in [120], creates a unique model known as RCFS-CARS. This refers to the Context-Aware Recommender System (CARS) with Noise Detection and Correction (NDC) with Relaxed Context Feature Sets (RCFS). The user ratings provided for every item are used to identify noise. Any discrepancy

between the user, the item, and the rating value is identified as noise in the rating. The RCFS-CARS technique is then used to correct for the noise.

Noisy ratings are a problem for collaborative filtering recommender systems (CFRS) based on Matrix Approximation (MA). In [121], a method known as Noise Resilient Matrix Approximation (NORMA), is proposed as an adaptive weighting strategy to address the issue of noisy ratings. By reducing the number of noisy rating learning steps, this technique improves the performance of the MA-based CFRS.

Each of the aforementioned strategies focuses on specific recommendations. Castro et al. [122] suggest using natural noise management in group recommendations to control noise at various rating levels and lower prediction error. Research is done using the MovieLens and Netflix datasets, and the findings indicate that the recommendation system performs more accurately.

A method that makes use of the users' ranking of the items is suggested in [123]. Users typically give popular items high ratings. However, occasionally, such items receive a lower rating. Sincere users inadvertently assign lower ratings to these kinds of products. Therefore, those who are providing fictitious ratings must be eliminated from the database. Therefore, if a user gives a rank for a popular item that is lower than a threshold value, that user is considered noisy.

3.5.1.2 Classification & Clustering

A strategy used in [124] is predicated on identifying a user as an expert and offers three techniques for adjusting the noisy ratings. The first approach focuses on the weighted average rating that various experts have given the same item. The second approach centers on the expert's weighted average rating of various items. The mean value of the above two methods is used as the third method. The entire approach increases the accuracy of the MovieLens dataset by concentrating on item attributes.

Enhancing the volume and caliber of user data is the main goal of another study that has been suggested in [125]. The Transfer Latent Factor Model, a novel framework based on user ratings, is proposed. This method divides the users into three

categories: heavy, medium and light, based on behavior data. The user's ratings are used to determine this grouping. Noise is identified and eliminated for heavy users, while noise is identified and corrected for light users but no processing is done for medium users.

A different method is applicable to ratings of noisy and sparse data [126]. To identify and correct noisy data, this approach first divides users and items into three classes: weak, average, and strong. Subsequently, the second phase involves utilizing sparse, noise-free data in conjunction with the Bhattacharya coefficient to forecast unrated items and suggest desired items to users.

Group Recommendation Systems (GRS) [127] are becoming more and more popular these days on social networking sites such as Facebook, Twitter. Users who share similar interests form groups. Therefore, we must remove group members who have different interests or traits before we can suggest a group to any user. These users are referred to as noise because of their divergent interests. This study deploys Decision trees that are created for GRS after hierarchical clustering is utilized to eliminate this noise. With this method, we obtain 73% accurate results.

3.5.1.3 Magic Barrier

The goal of the study in [128], based on user ratings, is to raise recommender systems' magic barrier. The magic barrier is a measure of whether or not a recommender system's accuracy can be increased. The authors of [128] assess their method using recommender systems for movies. The differences in user ratings are assessed while analyzing the movie dataset. These scores are employed to forecast the system's accuracy.

Another definition of the magic barrier is the recommender system's lower bound on error [129]. This error is referred to as noise that influences users' rating preferences. According to the work suggested in [129], the magic barrier and user coherence are connected. This work distinguishes between easy users (lower magic barrier) and difficult users (higher magic barrier) using this coherence factor. The findings show that users with high coherence have lower recommendation error.

A different study using the magic barrier is discussed in [130], where it is stated that changing the user profile may cause the recommendations to be incorrect. In order to achieve this, this work suggests a coherence-based method called Dynamic Coherence-Based Modeling (DCBM), which eliminates from the user profile any items that are not pertinent to the user. The results show that the recommendation system is now more accurate after removing such items.

3.5.1.4 Outliers Detection

Datasets that don't fit into the intended dataset are known as outliers. According to Li et al. [131], natural noise is the noise that users unintentionally introduce due to their variety of personalities. It recognizes users who are noisy but not malicious and who rate comparable products differently. The foundation of this strategy is the idea that it doesn't rely on any data outside of ratings. The outcomes demonstrate that when the noise is eliminated, accuracy increases. Toledo et al. [132] describe an alternative method based on the same idea that detects and corrects natural noise using current ratings. It classifies noisy data using item and user profiles, then makes necessary corrections.

An alternative method [133] models the noise as outliers using the Gaussian Distribution and computes the recovered ratings using the Expectation Maximization (EM) algorithm. Following several iterations of this comparison, the authors compare the original and recovered ratings to identify outliers; ratings that are suspected of being outliers are then handled as such.

3.5.1.5 Global Information

Global information has been used in some natural noise management techniques. A proposal for this method is found in [134], wherein global information is derived from user and item preferences. If the user's and the item's preferences are not aligned, this information is used to classify ratings as noisy. The noise that is detected is then corrected.

3.5.1.6 Summary of Crisp Methods

Table 3.5 enumerates all of the crisp techniques covered above, outlining the various datasets used to assess these techniques and the evaluation metrics to evaluate the proposed approaches.

Table 3.5: Summary of all the discussed crisp approaches

S.No.	Publications	Noise filtering methods	Datasets	Accuracy measurement tools
1.	[116]	Re-Rating	Netflix & Movies Dataset	RMSE
2.	[117]	Re-Rating	Netflix Dataset	RMSE
3.	[118]	Re-Rating	MovieLens & Netflix Dataset	RMSE
4.	[119]	Zero error in user and item training dataset	Netflix Prize, MovieLens, Jester & Jester2 Datasets	RMSE
5.	[120]	Using Current Ratings	IncarMusic & LDOS-CoMoDa Datasets	RMSE, Response Time, F1-measure
6.	[121]	Ratings	MovieLens & Netflix Datasets	RMSE
7.	[122]	Rating Levels	Netflix Tiny & MovieLens datasets	MAE
8.	[123]	Ranking	MovieLens and Jester Datasets	MAE, RMSE, F1 measure
9.	[124]	Classification	MovieLens Dataset	RMSE
10.	[125]	Clustering	MovieLens Dataset	RMSE, Precision

11.	[126]	Classification	MovieLens Dataset	MAE, RMSE, Precision, Recall, F1 measure
12.	[127]	Clustering	Facebook Dataset	Accuracy
13.	[128]	Magic Barrier	Moviepilot Dataset	RMSE
14.	[129]	Magic Barrier	MovieLens, Moviepilot & Yelp Datasets	RMSE
15.	[130]	Magic Barrier	Yahoo! Webscope & MovieLens Datasets	RMSE, Average Difference
16.	[131]	Using Current Ratings	EachMovie & MovieLens Datasets	Precision
17.	[132]	Using Current Ratings	MovieLens & MovieTweeting Datasets	MAE, F1 measure
18.	[133]	Using Recovered Ratings	MovieLens, Amazon, Yelp & FilmTrust Datasets	Mean Average Precision, RMSE
19.	[134]	Global Information	MovieLens Dataset	MAE

3.5.2 Fuzzy Tools

Fuzzy tools were first proposed by Yera et al. [135] to control natural noise in recommender systems. This study discusses the Crisp management's shortcomings and how they are eliminated by the fuzzy method. This indicates that when recommending products to users, the fuzzy method is strong and adaptable enough to handle rating ambiguity and uncertainty as well as natural noise management.

Another study [136] uses fuzzy tools in four steps to detect and correct natural noise. First, it detects the irregularities in the ratings database, then, secondly, it filters them. Then in the third step, it calculates the degree of noise and finally in the fourth step, it detects and corrects that noise, making the ratings database noise-free.

In [137], a multiphase fuzzy linguistic approach is proposed. It first determines the users' gender before calculating the rating values and item tendency scores. The next step is to define fuzzy sets, which are used to translate rating values and item tendency scores into linguistic values. After that, conflict ratings are determined by comparing the linguistic values of the two. Next, the linguistic sets are used to categorize the noise. Ultimately, noise is identified and user attributes are calculated.

In [138], a different fuzzy approach that divides ratings into light, medium, and heavy classes is presented. Then this method generates fuzzy user and item profiles. The generated user and item fuzzy profiles are then used to identify the noise. Using threshold rating values in accordance with the Maximum membership principle, detected noise is finally corrected.

Group recommendations are another application for fuzzy tools. In the age of social media, it's possible that we begin to follow certain individuals. Thus, group recommendation is required in these situations. A method for making recommendations to a group of users that takes into account everyone's likes and dislikes is called group recommendation. When recommending a group, one must take into account the interests of each member and then suggest a course of action that will satisfy everyone.

J. Castro et al. [139] suggest another method called Natural Noise Management in Group Recommendation using Fuzzy Tools (NNMG-FT). It eliminates the ratings that users have left for a variety of items from the ratings database. There are three steps to this approach: (i) Fuzzy Profiling generates features for users, items, and ratings; (ii) Global Noise Management takes the features produced by fuzzy profiling and manages the rating database globally; (iii) Local Noise Management eliminates localized noise from the ratings database. A noise-free ratings database is produced throughout the entire process.

All of the fuzzy approaches covered above are compiled in Table 3.6, which also includes the evaluation metrics, various datasets used to test these methods, and whether these recommendations are applied on individual or group.

Table 3.6: Summary of all the discussed fuzzy approaches

S.No.	Publications	Recommendation Type - Individual / Group	Datasets	Accuracy measurement tools
1.	[135]	Individual	MovieLens, MovieTweeting & Netflix Datasets	MAE, F1 measure
2.	[136]	Individual	MovieLens, MovieTweeting & Netflix Datasets	MAE
3.	[137]	Individual	MovieLens Dataset	Detection Percentage, Precision, Recall, F1-measure
4.	[138]	Individual	MovieLens & Yahoo music Datasets	MAE, RMSE, Precision, Recall, F1-measure
5.	[139]	Group	Netflix Tiny & MovieLens datasets	MAE

3.6 Structural and Contextual Noise

Apart from malicious and natural noise, there exists other forms of noise as well:

- Structural Noise: This type of noise refers to the inconsistencies and irregularities in the data structure or format.

- Contextual Noise: This noise refers to the dynamic needs and behavior of the user due to the change in the context like time, location and moods.

To handle such types of noise, advanced techniques are required. Optimization frameworks have emerged as a powerful tool to take care of such noises. Optimization is a mathematical and computational technique which picks up the best solution among any other possible solutions for a given problem. It has been applied to various fields including engineering, marketing, economics and finance where it helps in decision making to make more profits. Some of the applications of optimization are fake news detection [140], healthcare [141], supply chain management [142], Natural Language Processing [143] and many more [144,145].

Optimization improves the quality of the recommendation systems. It focusses on refining the performance of RS by reducing the impact of noise through mathematical and computational methods. One such work [50] includes the concept of Federated Learning in POI based RS. In this work, Contrastive Learning (CL) which is a multi-task framework is used for this purpose. This study suggests a novel sequential information-based (FedSR) framework for POI-RS. A multi-task framework in the FedSR is constructed using Contrastive Learning and uses a data augmentation method based on spatial relationships among POIs. For the recommendation task, Bayesian Personalized Ranking (BPR) optimization is applied. BPR optimization rationalizes the negative examples in federated CL, which improves the accuracy of the recommendations made.

The study [146] explores the application of Deep Q-Networks (DQN) in news recommendation systems for recommending personalized news to users. DQN emphasizes the combination of gradient descent optimization methods and loss functions. By combining these two methods, the accuracy of Q-value estimation is improved and thus provides users with more precise and customized article recommendations.

Another work [59] uses Gradient Descent Optimization to optimize the loss function in order to improve the accuracy of recommendations made for social friends and foes. The authors propose a deep learning technique to compute the

nonlinear correlations between user preferences and the social knowledge of links between distrust and trust at the deep representations by optimizing the ranking loss function with various ranking criteria. Using backpropagation, they determine the parameters of their ranking model. To achieve the numerous ranking criteria of their ranking loss function, they employ a social negative sampling technique in each backpropagation phase.

In another work [80], authors propose utilizing sequential decision optimization to enhance user engagement over the long term for recommender systems. To be more precise, authors directly model users' click and return behaviors for online optimization. In online learning, three competing factors are taken into account: exploring unknowns for model estimation, exploiting current clicks and exploiting clicks in the future. Authors devise a bandit-based solution to achieve this balance. They rigorously show that in optimizing accumulated interactions from a sample of consumers in an interval of time, their proposed strategy most likely accomplishes an upper remorse limit. However, a linear remorse is unavoidable if the recommendations are made without taking into account the user's temporal return behavior.

The authors of [109] suggest a three-stage method for detecting abnormal ratings in collaborative filtering recommender systems. First, an undirected user-user graph is built using user profiles as a guide. In the second step, user similarities are discovered using graph mining techniques to optimize the graph. The optimized graph is also subjected to an equivalent analysis to separate a subset of legitimate users from dubious ones. The remaining genuine users are then further filtered out in the third stage by looking at target items in order to identify the fake profiles. Experiments are conducted using the MovieLens dataset, and results show how effective this strategy is compared to other methods.

All of the optimization approaches covered above are compiled in Table 3.7, which also includes the type of recommendation, various datasets used to test these methods and accuracy measurement tools.

Table 3.7: Summary of all the discussed optimization approaches

S.No.	Publications	Optimization Type	Recommendation Type	Datasets	Accuracy measurement tools
1.	[50]	Bayesian Personalized Ranking (BPR) Optimization	POI based RS	Brightkite, Gowalla	Hit Rate, NDCG
2.	[146]	Gradient Descent Optimization	News Recommendation	Historical user interaction data	Click-through rate (CTR)
3.	[59]	Gradient Descent Optimization	Social friends and foes recommendation	Epinions dataset	Recall, NDCG
4.	[80]	Sequential Decision Optimization	News Recommendation	Yahoo frontpage news recommendation module	CTR, Return Rate, User ratio
5.	[109]	Graph Optimization	Movies Recommendation	MovieLens dataset	Detection Rate, False Alarm Rate

3.7 Conclusion

We present a summary of sampling techniques that include everything from movie or product recommendations to interactive and conversational recommendations. We have discussed six types of sampling methods used for recommender systems, namely, Bayesian Hierarchical Sampling, Negative Sampling, Thompson Sampling,

Bernoulli Sampling, Gibbs Sampling and Bootstrap Sampling. We conclude that for disease diagnosis recommendation systems, bootstrap sampling is used. When ranking or rating is involved, such as in online user-item recommendations, negative sampling is deployed. While Gibbs sampling and Bernoulli distribution are better suited for suggesting movies and products, Bayesian Hierarchical sampling is employed for suggesting picturesque locations. While Thompson sampling is useful for determining how offline parameters affect online performance, it is limited in that it cannot replicate real-world human behavior, which makes it difficult to improve the performance of interactive and conversational recommender systems.

Next, we present a summary of noise filtering techniques. We distinguish between two major types of noise: malicious and natural. Furthermore, we discover that all of the suggested methods for filtration of noise are based on Collaborative Filtering Recommender Systems (CFRS) rather than Content-based RS. In this context, "noise" refers to false ratings that can be generated by users' inconsistent behavior or deliberately entered by users. The CFRS, which builds a user-item matrix based on user ratings for items they like, is intentionally impacted by false ratings. Adolescents with similar interests are recommended inappropriate items based on this fake matrix. As a result, it's critical to eliminate noise from CFRS.

Apart from malicious and natural noise, there exists other forms of noise as well such as structural noise and contextual noise. In order to handle these types of noise, advanced techniques such as optimization algorithms are applied to the recommendation systems. Optimization improves the quality of the recommendation systems. It focusses on refining the performance of RS by reducing the impact of noise through mathematical and computational methods. Thus, we provide an overview of a few studies that use optimization techniques to reduce/eliminate the impact of noise while measuring the performance of the recommender systems.

CHAPTER 4

NPO BASED MOVIE RECOMMENDATION MODEL

After the exploration of various sampling and noise filtering methods in the previous chapter, this chapter proposes a Movie recommendation model using a suitable sampling method and an optimization technique for noise filtering. With the exponential growth of social media platforms, user-generated content has become a valuable resource for various applications, including recommendation systems. However, the presence of noise in user-centric tweets often hampers the performance of these systems by introducing irrelevant or misleading information. This chapter presents a novel approach utilizing the Nuclear Physics Optimization Algorithm (NPO) for noise filtering among user-centric tweets to enhance the accuracy and reliability of recommendation systems.

The proposed methodology involves preprocessing the raw tweet data and extracting relevant features that capture user preferences and interests. Subsequently, the NPO is employed to identify and filter out noisy tweets based on their semantic similarity and relevance to the target domain. The NPO dynamically adjusts its parameters to optimize the filtering process and adapt to the characteristics of the tweet dataset. Experimental evaluations conducted on real-world tweet dataset of Movie recommendation demonstrate the effectiveness of the proposed approach in improving the performance of recommendation systems. Comparative analysis against baseline methods reveal significant enhancements in prediction accuracy and recommendation quality. Moreover, the proposed method exhibits robustness and scalability across diverse tweet datasets and recommendation scenarios.

4.1 Introduction

The proliferation of social media platforms has introduced new opportunities and challenges for recommendation systems. User-generated content, social interactions, and social graphs contain valuable information that can augment traditional recommendation techniques. Integrating social signals into recommendation

algorithms enables systems to leverage social influence, user trust, and community preferences to generate more effective recommendations [158]. However, the noisy and unstructured nature of social media data poses challenges in extracting relevant information and mitigating the impact of misinformation or biased signals. Addressing these challenges requires innovative approaches that combine data mining, natural language processing, and social network analysis to harness the power of social media for personalized recommendation generation.

Beyond e-commerce, recommender systems have found extensive utility in content streaming platforms such as Netflix [2], Spotify [159], and YouTube [1], where they curate personalized playlists, and recommend movies, music tracks, and videos based on user interactions and feedback. By accurately predicting user preferences and anticipating their content consumption patterns, these systems optimize user engagement, retention, and subscription revenues. Moreover, in the domain of social networking, recommender systems facilitate meaningful connections and interactions by suggesting friends, groups, and content of interest, thereby enhancing user experience and fostering community engagement [158].

In the realm of NLP, recommender systems leverage advanced linguistic analysis techniques to understand the semantic meaning and context of textual data, enabling them to generate personalized recommendations [160]. These systems employ a variety of algorithms to model user preferences based on their interaction history and the characteristics of the items being recommended. By leveraging linguistic features such as semantic similarity, sentiment analysis and topic modeling, NLP-powered recommender systems can provide more accurate and contextually relevant recommendations to users.

On the other hand, user-centric tweets represent a unique and valuable source of data for NLP-based recommender systems, offering real-time insights into user preferences, opinions, and interests. Unlike traditional textual sources, such as product reviews or news articles, user-centric tweets are characterized by their brevity, informality, and rapid dissemination. Despite these challenges, NLP techniques enable recommender systems to extract valuable signals from user-centric

tweets, such as sentiment, topics, and user interactions, to better understand user preferences and generate personalized recommendations [161]. However, the noisy and unstructured nature of tweet data poses significant challenges for recommendation systems, necessitating the development of innovative approaches for noise filtering and information extraction to enhance recommendation quality.

Integrating user-centric tweets into NLP-powered recommender systems opens up new opportunities for enhancing recommendation accuracy and relevance by incorporating real-time user feedback and social context. By harnessing the rich linguistic information embedded in tweets, such as hashtags, mentions, and conversational patterns, recommender systems can capture nuanced user preferences and emerging trends in a dynamic social environment [38]. Moreover, the inherent diversity and timeliness of tweet data enable recommender systems to adapt quickly to changing user interests and preferences, providing more personalized and up-to-date recommendations.

In recommendation systems, the presence of noise in tweets can distort user preferences and interests, resulting in erroneous recommendations. By filtering out noise, recommendation algorithms can focus on meaningful content that reflects genuine user preferences, leading to more personalized and relevant recommendations. Moreover, noise filtering contributes to the overall trustworthiness and credibility of recommendation systems, enhancing user satisfaction and engagement.

Filtering noise in tweets is crucial due to the vast amount of user-generated content on social media platforms. With millions of tweets posted daily, noise, such as spam, irrelevant information, and misinformation, can significantly degrade the quality of data used for various applications, including recommendation systems. Noise filtering helps to ensure that only relevant and reliable information is considered, leading to more accurate predictions and improved user experiences.

The main contributions of this chapter are:

Firstly, the utilization of the Nuclear Physics Algorithm (NPO) introduces a novel and effective method for filtering noise in tweet datasets, thus enhancing the accuracy and reliability of recommendation systems. Unlike traditional approaches, NPO dynamically adjusts its parameters to optimize noise filtering, thereby adapting to the unique characteristics of the tweet data and improving its efficiency and effectiveness.

Secondly, the proposed methodology contributes to enhancing the accuracy and reliability of recommendation systems by effectively removing noise from user-centric tweets. This ensures that only relevant and reliable information is considered in the recommendation process. This leads to more accurate predictions and personalized recommendations, ultimately enhancing user satisfaction and engagement with the recommendation system.

Lastly, the proposed approach contributes to advancing the utilization of social media data for recommendation systems by providing a scalable and robust solution for noise filtering in diverse tweet datasets. The effectiveness of the NPO-based filtering method is demonstrated through comprehensive experimental evaluations on real-world tweet datasets, showcasing its superiority over baseline methods.

4.2 Proposed Methodology

The proposed methodology involves preprocessing the raw tweet data and extracting relevant features that capture user preferences and interests. Subsequently, the NPO is utilized to optimize the noise-filtering process by iteratively adjusting its parameters to maximize the relevance of retained tweets while minimizing the impact of noise.

The key contributions of this chapter lie in the innovative application of the NPO for noise filtering in user-centric tweets and its integration into recommendation systems. By enhancing the quality of tweet data used for predictions, the proposed approach leads to more personalized and relevant recommendations, ultimately improving user

satisfaction and engagement. Moreover, the scalability and adaptability of the NPO make it well-suited for handling diverse tweet datasets and recommendation scenarios, highlighting its potential for practical applications in real-world systems. The proposed framework for the recommendation system of the user-centric data is shown in Fig.4.1.

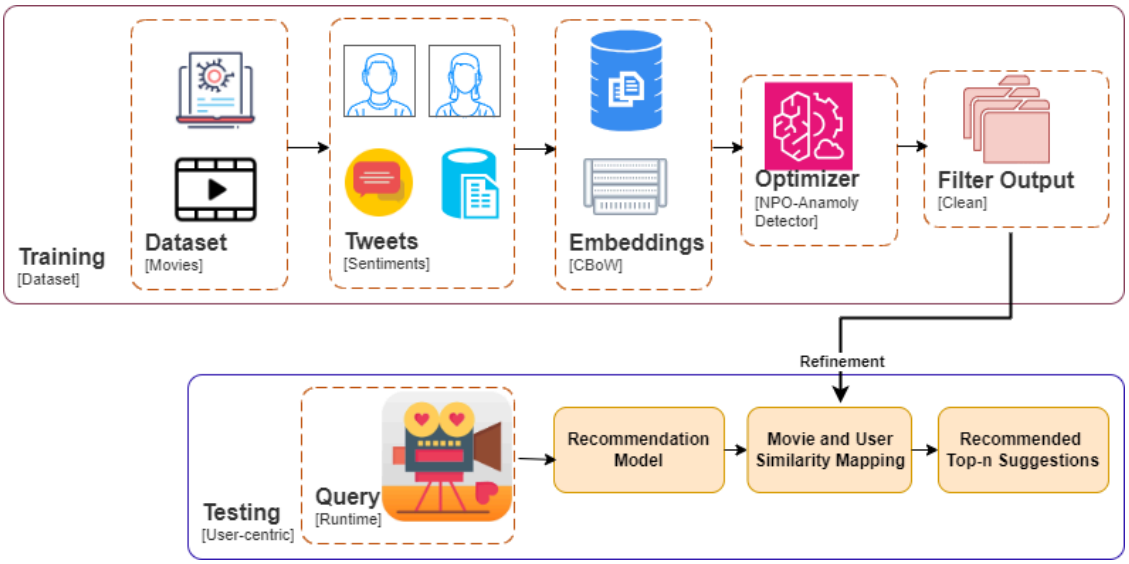

Fig.4.1: Flowchart of the proposed framework for movie recommendation

4.2.1 Dataset Preparation

For our study, we utilize two distinct datasets: one comprising movie metadata taken from the MovieLens Dataset from the URL: https://grouplens.org/datasets/movielens/latest/ and the other capturing Twitter sentiments related to these movies. The movie dataset encompasses comprehensive information on over 45,000 movies included in the Full MovieLens Dataset. A wide range of information is included in this metadata, such as cast and crew bios, posters, plot keywords, revenue and budget estimates, countries of origin, production firms and release dates. Additionally, it provides insights into the popularity and reception of each movie, with TMDB vote counts and average ratings facilitating a quantitative assessment of user opinions.

Moreover, the movie dataset incorporates a substantial volume of user ratings, constituting a significant aspect of the dataset's richness. With a repository of 26 million ratings contributed by 270,000 users, this dataset offers a comprehensive view of audience preferences and opinions across the entire spectrum of movies. Ratings are recorded on a scale of 1 to 5 and are sourced from the official GroupLens website, ensuring credibility and reliability. This extensive collection of user ratings serves as a valuable resource for understanding viewer sentiments and preferences towards different movies.

In conjunction with the movie metadata, we augment our analysis with data derived from Twitter sentiments pertaining to these movies. By leveraging Twitter's vast platform for real-time user-generated content, we gain insights into the public discourse and sentiment surrounding each movie. This Twitter sentiment dataset provides a complementary perspective to the quantitative movie metadata, offering qualitative insights into audience reactions, opinions, and discussions in the social media sphere. Integrating these two datasets enables us to perform a comprehensive analysis that combines quantitative metrics with qualitative insights, enriching our understanding of the factors influencing movie recommendations and user preferences.

The sentiments incorporated in our analysis are derived from users who have assigned sentiment labels to their expressions, categorized into five distinct categories: 0 for negative sentiment, 1 for somewhat negative, 2 for neutral, 3 for somewhat positive, and 4 for positive sentiment. These sentiment labels provide a nuanced understanding of user opinions and attitudes towards the movies under consideration. By categorizing sentiments into discrete levels, ranging from highly negative to highly positive, we capture the full spectrum of user sentiment variations, enabling a comprehensive analysis of audience perceptions.

The sentiment labels assigned by users serve as valuable indicators of their subjective reactions and experiences with the movies. Negative sentiments (labels 0 and 1) reflect dissatisfaction or disappointment, while positive sentiments (labels 3 and 4) signify enjoyment or appreciation. The neutral sentiment label (label 2) indicates a

lack of strong emotional inclination, suggesting a more balanced or indifferent stance towards the movie. By considering these diverse sentiment labels, we can discern patterns and trends in user perceptions, identifying key factors that influence audience satisfaction and engagement.

4.2.2 Movie Review Sentence Embedding

To make the user-centric comments on the movies ready for the model training, here we have used the Continuous Bag-of-Words (CBoW) algorithm [162] with negative sampling for sentence embedding, facilitating the creation of dense vector representations for sentences. This approach is particularly effective for sentiment analysis and recommendation systems where understanding the semantic meaning of sentences is crucial. The framework of sentence embedding is shown in Fig.4.2.

The CBoW algorithm operates by predicting a target word based on the context of surrounding words within a fixed-size window. Unlike skip-gram models [162], which focus on predicting surrounding words given a target word, CBoW aims to predict the target word given its context. This makes it well-suited for generating sentence embedding, as it captures the overall meaning and context of the sentence rather than individual word meanings.

In the CBoW algorithm with negative sampling, the training process involves sampling negative examples (words not present in the context window) to contrast with the positive examples (actual context words). This helps the model learn to distinguish between relevant and irrelevant words in the context of generating sentence embedding. By iteratively adjusting the model parameters to minimize the prediction error for positive examples while maximizing it for negative examples, the algorithm learns to produce more accurate and meaningful sentence embedding.

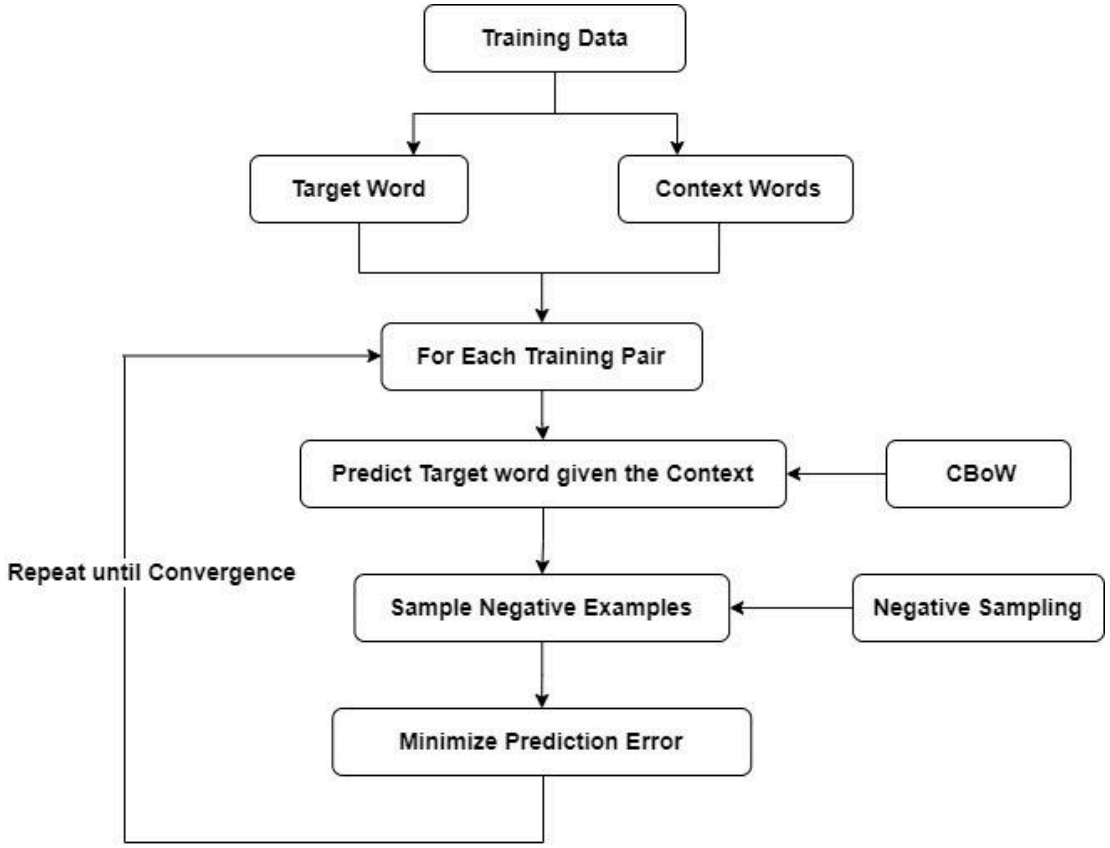

Fig.4.2: Framework for Movie Review Sentence Embedding

During training, the CBoW model processes each sentence by averaging the word embedding of its constituent words, effectively condensing the entire sentence into a single vector representation. This sentence embedding captures the semantic meaning and context of the sentence in the high-dimensional vector space. By leveraging negative sampling, the model learns to focus on informative words while filtering out noise and irrelevant information, resulting in more robust and informative sentence embedding.

Here is an example to showcase the negative samples. Suppose, there is a tweet: "The acting in the movie was great, but the plot was confusing."

- Target Word: "great"
- Context Words: "acting", "movie"
- Sentiment Context: Positive

The positive examples consist of pairs of the target word and its actual context words. In this case, positive examples will be: (great, acting), (great, movie). Negative examples are words that are not present in the context window of the target word but could potentially be associated with it incorrectly. In this case, negative examples will be: (great, confusing), (great, plot). During training, the model is presented with both positive and negative examples. It learns to predict the positive examples accurately while predicting the negative examples as unlikely or irrelevant. The model's parameters are adjusted to minimize the prediction error for positive examples and maximize it for negative examples. Through this process, the model learns to focus on words that are relevant to the context of the movie review while filtering out noise and irrelevant words.

4.2.3 Nuclear Physics Optimization

Utilizing Nuclear Physics Optimization (NPO) for filtering noisy tweets represents a novel approach to enhance the quality and relevance of information extracted from social media data. Inspired by the principles governing atomic nuclei interactions, NPO offers a unique framework for optimizing the process of noise filtering in tweets, thereby improving the reliability of the extracted content.

In the context of noisy tweet filtering, NPO operates by treating each tweet as a particle within a system, with the objective of optimizing the selection of informative tweets while minimizing the impact of noise. The algorithm simulates the interactions between these ”particles,” representing tweets, by applying principles analogous to nuclear forces and electrostatic interactions. Through iterative optimization cycles, NPO dynamically adjusts the positions of tweets within the solution space, with attractive forces guiding the selection of relevant tweets and repulsive forces aiding in the expulsion of noisy or irrelevant content.

The effectiveness of NPO for filtering noisy tweets lies in its ability to balance exploration and exploitation, allowing for the efficient identification and retention of informative tweets while simultaneously mitigating the influence of noise. By leveraging the inherent parallelism and adaptability of NPO, the algorithm can

effectively navigate the complex landscape of social media data, identifying relevant signals amidst the noise. Moreover, NPO offers the flexibility to incorporate additional constraints or domain-specific knowledge, enabling tailored solutions for different noise filtering tasks.

The NPO algorithm is designed around the physics-based concept of particle decay. There are three types of decay processes prevalent in physics: alpha, beta, and gamma [164]. The term ”alpha, beta, and gamma particle decay” refers to the use of nuclear physics principles to drive particle decay simulation and comprehension. In our context, particles represent tweets and the decay simulation aims to filter out noisy or irrelevant tweets. In nuclear physics, alpha, beta, and gamma decay are the three most frequent kinds of radioactive decay that change unstable atomic nuclei into more stable structures. This decay process is described in Table 4.1.

Table 4.1 Various decays and their reference to noise filtration

Types of Decay	**Noisy Tweets Filtration**
The unsteady nucleus releases an alpha particle, which is made up of two neutrons and protons, during **alpha decay**.	Tweets containing spammy hashtags or promotional links are removed.
A neutron can change into a proton (beta-plus decay) or a proton can change into a neutron (beta-minus decay) in **Beta decay.**	Tweets having grammatical errors or slang language are corrected so as to transform the tweets to be able to be processed.
When an energized nucleus emits extra energy in the form of a gamma-ray photon, it is known as **gamma decay.**	Excessive emoticons or punctuation marks are removed from the tweets so as to reduce noise, just like the excess energy is released by an energized nucleus as a gamma-ray.

In the physics concepts, the particle is also evaluated on the measure of the Neutron Enrichment Level (NEL). It is a measure of the relative abundance of neutrons compared to protons in a given nucleus or particle. It is often used in the context of

nuclear physics and nuclear engineering to describe the stability and characteristics of atomic nuclei. Thus to find the stability of the particle A_i, the NEL of the particle plays a crucial role. For any particle, NEL is calculated using Equation 4.1.

$$NEL_{Ai} = (N_{Ai} - P_{Ai}) / M \tag{4.1}$$

Where N_{Ai} and P_{Ai} are the number of Neutrons and protons of particle A_i , and M is the atomic mass of the particle and can be evaluated as the sum of the Protons and Neutrons.

The value computed by Equation 4.1 has importance in decision making. The neutron surplus or deficit in the nucleus about the amount of protons is indicated by the NEL value. A positive NEL value denotes a neutron excess, which is the nucleus having more neutrons than protons. There may be a propensity toward neutron-richness if the NEL value is positive. Conversely, a negative NEL score denotes a neutron deficiency, which is the absence of more neutrons in the nucleus than protons. One may expect a propensity toward proton-richness if the NEL value is negative.

Another use of the NEL value in particle physics is to provide insights into the stability and behavior of atomic nuclei. Nuclei with higher NEL values (positive values) have an excess of neutrons compared to protons and may be more likely to undergo decay to achieve a more stable neutron-to-proton ratio. Conversely, nuclei with lower NEL values (negative values) may be more stable or prone to other modes that release energy only, like gamma decay, depending on their specific nuclear properties.

To make this decision, we calculate the enrichment bound of the particle system. It refers to the maximum or minimum value allowed for a certain property or characteristic of the particle. In the context of nuclear physics or particle interactions, the enrichment bound may pertain to parameters such as the neutron-to-proton ratio, energy levels, or other physical quantities. The mathematical calculation to compute the enrichment bound is given by Equation 4.2.

$$EB_u = \frac{\sum_{i=1}^{N} NEL(Ai)}{N} \quad (4.2)$$

where N is the total number of particles in the system.

Thus, if the NEL of a particle is discovered to be more than the Enrichment bound (EB_u), i.e. $NEL(Ai) > EB_u$, then the associated particle is assumed to have the higher Neutron to Proton ratio, and the particle is expected to decay. Finally, the new particle formed is given by Equation 4.3.

$$A_i^{new} = A_i^m + \text{Best Solution} \quad (4.3)$$

Next is the condition when the NEL(Ai) is found to be lower than the Enrichment bound EB_u i.e. $NEL(Ai) < EB_u$ then the condition is considered that the system has the lower Neutron to proton ratio. In such a case the particle will not go under any decay process where the mass decreases. There is only the release of energy. Thus, the new particle position in such a case is calculated by using Equation 4.4.

$$A_i^{new} = A_i^m + \text{rand}(0, 1) \quad (4.4)$$

4.2.4 Evaluating Fitness of Each Tweet for Filtering

To evaluate the fitness function of each tweet for filtering noisy tweets, we can formulate it mathematically using the provided variables. Let's denote the fitness function as F(t) where t represents a tweet. The fitness function can be defined as the ratio of properly categorized samples to the total number of samples, adjusted by the ratio of selected features to the total number of features in the dataset. Mathematically, this can be expressed by Equation 4.5.

$$F(t) = A_i^{new} \times (\alpha(1 - TC/TD) + \beta(Nf/NF)) \quad (4.5)$$

where α and β are the constant parameters with the values of 0.1 and 0.55 respectively which are figured out experimentally. TC is the number of samples that were properly categorized, TD is the total number of samples in the data set, Nf is the number of features selected by the optimization algorithm, and NF is the total number of features in the data set.

This fitness function captures the performance of each tweet in terms of its relevance to the classification task (properly categorized samples) and the importance of the selected features in distinguishing relevant tweets from noisy ones. By incorporating these factors, the fitness function provides a quantitative measure of the suitability of each tweet for filtering noisy tweets, guiding the optimization algorithm towards selecting tweets that contribute most effectively to the noise filtering process.

4.2.5 Models for Learning Recommendation System

Various machine learning and deep learning models are used in the learning phase of movie recommendation using tweets to evaluate twitter data and extract valuable insights for recommendation purposes. Using supervised machine learning methods, including Logistic Regression, Support Vector Machines (SVMs), or Naive Bayes classifiers [165], is one often employed method for categorizing tweets according to their sentiment or applicability to particular films. The sentiment (positive, negative, or neutral) or relevance (relevant, irrelevant) labels attached to each tweet are used to train these models on labeled tweet data. These models can efficiently categorize fresh tweets and determine which ones are most pertinent for movie recommendation by learning from the labeled data.

Moreover, movie recommendation systems are increasingly using deep learning models—in particular, Recurrent Neural Networks (RNN) [166]—to analyze Twitter data. Sentiment analysis tasks require a comprehension of language flow and context, and RNNs are well suited for these kinds of tasks since they can capture sequential dependencies in tweet text. More precise and sophisticated movie suggestions are made possible by this deep learning model, which is trained on vast amounts of twitter data to create representations that capture the sentiment and semantic meaning of tweets.

4.2.6 Relevancy Selection with Cosine Similarity

In the realm of movie recommendation systems utilizing tweets, cosine similarity [163] emerges as a fundamental metric for retrieving relevant movie suggestions

based on the textual content of tweets. Cosine similarity quantifies the similarity between two vectors in a multi-dimensional space, making it an ideal measure for comparing the semantic similarity of tweet representations and movie attributes. Mathematically, cosine similarity between two vectors A and B is calculated as the cosine of the angle between them and is defined by using Equation 4.6

$$Cos_{sim}(A,B) = (A.B) / ||A||.||B|| \quad (4.6)$$

where A.B denotes the dot product of vectors A and B, and ||A|| and ||B|| represent their respective Euclidean norms.

To apply cosine similarity in the context of movie recommendation using tweets, each tweet and movie are represented as vectors in a common feature space. For tweets, vector representations can be derived from tweet embeddings obtained through CBoW capturing the semantic meaning of tweet content. Similarly, movie vectors encapsulate relevant attributes such as genre, cast, and plot keywords, extracted from movie metadata.

Once tweet and movie vectors are obtained, cosine similarity is computed between each tweet vector and the vectors representing all movies in the dataset. This yields a similarity score for each movie, quantifying its relevance to the tweet. The movies with the highest cosine similarity scores are then recommended to the user, as they are deemed most similar in content and context to the tweet.

4.3 Experimentation Results

4.3.1 Pre-build Dataset

To evaluate the results of the proposed approach, the dataset was built using two strategies. The first strategy uses the movie lens dataset which has numerous movies with their genre, titles, and ratings. For these movies, the user-centric reviews of the respective movies were fetched, and the sentiment score was generated for each movie item in the dataset. The sample collection of the dataset is shown in Table 4.2. Also, Fig.4.3 provides a summary of the distribution of ratings in the Full MovieLens Dataset, showing how many samples fall within each rating category. It helps

visualize the frequency of different rating values given by users, which is crucial for understanding user preferences and evaluating recommendation algorithms.

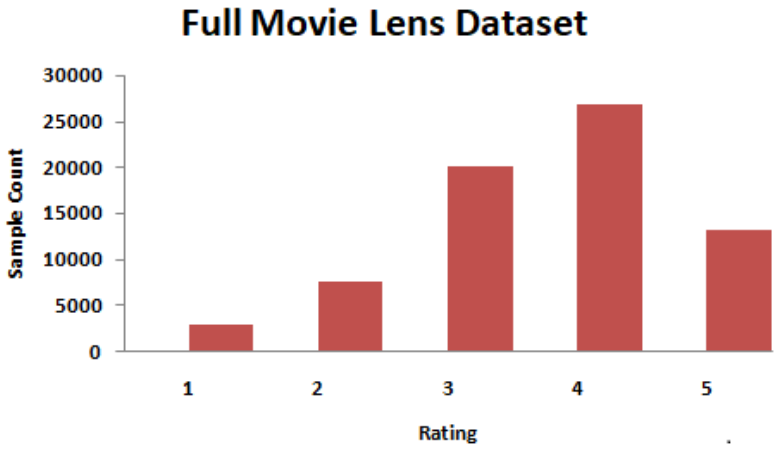

Fig.4.3: Distribution of samples in Movie Lens dataset as per rating

Table 4.2: Sample Dataset having movies title, genre, fetched user-centric review and sentiment score

S.No.	Original Title	Genres	User-centric Comment	Sentiment
1	Jumanji	Adventure, Fantasy, Family	Jumanji, with plenty of laughs, action-packed excitement, great music, spectacular sets, and inspirational themes, this film is an absolutely winning adventure.	4
2	Ace Ventura	Romance, Comedy	Ace Ventura. Neither terrible, boring nor soporific, just not very funny.	1
3	Die Hard	Drama	Die Hard. There are good performances from everyone in this long, often funny, very violent but exciting melodrama.	2

4	Meet Joe Black	Drama	Meet Joe Black. I've never encountered such dramatic flatulence, never heard so many pregnant silences that don't deliver, never watched so many close-ups that graze on actors' faces until every last trace of expression has been devoured.	3
5	Toy Story	Comedy	Toy Story is a Pixar classic, one of the best kids' movies of all time.	4

Using the formed dataset, the model is trained to items that are most closely associated with the query item. Based on the constraint, the model predicts the top k items among the dataset that are most likely to be the query item.

4.3.2 Results with NPO Algorithm

The experimentation results from model training on movie recommendation using tweets filtered by the proposed NPO algorithm offer insightful information on the performance of different classifiers, including Logistic regression, Naive Bayes classifiers, Support Vector Machines (SVMs) and Recurrent Neural Networks (RNN). The evaluation measures, which comprise Mean Average Precision (MAP), Precision, Recall, F1-Score and Accuracy, were applied to the diverse group of classifiers. They offer a comprehensive assessment of their effectiveness in generating movie recommendations tailored to user preferences based on tweet data. The experimentation results of the proposed framework are shown in Table 4.3.

Table 4.3: Experimentation results of the model training with proposed NPO algorithm-based tweet filtering

Classifier	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**MAP**
Support Vector Machine (SVM)	0.91	0.90	0.88	0.89	0.88

Naive Bayes (NB)	0.82	0.80	0.85	0.82	0.70
Logistic Regression (LR)	0.87	0.88	0.86	0.87	0.78
Recurrent Neural Networks (RNN)	0.94	0.93	0.91	0.92	0.92

The results in Table 4.3 indicate that the machine learning classifiers achieved varying degrees of success in accurately predicting relevant movie recommendations. Support Vector Machines demonstrated strong overall performance, achieving high accuracy, precision, recall, and F1-score, suggesting robustness in distinguishing relevant tweets for movie recommendation. Naive Bayes classifiers exhibited competitive performance, particularly in recall and F1-score, although slightly lower accuracy and precision were observed compared to SVMs. Logistic regression models also yielded promising results, with high accuracy and precision, indicating their efficacy in classifying relevant tweets for recommendation purposes.

Furthermore, the experimentation results highlight the effectiveness of deep learning-based approaches, particularly Recurrent Neural Networks (RNNs), in capturing the complex relationships and patterns in tweet data for movie recommendations. RNNs demonstrated superior performance across multiple metrics, achieving the highest accuracy, precision, recall, and F1-score among the classifiers evaluated. Additionally, RNNs outperformed traditional classifiers in terms of Mean Average Precision (MAP), indicating their ability to provide high-quality ranked recommendations that align closely with user preferences.

Also, the experimentation was performed to present the outcomes of our movie recommendation system, focusing on the top-k (k=3) movie titles recommended to users based on their queries. We begin by outlining the methodology employed to generate these recommendations, detailing the algorithms, models, and evaluation metrics utilized in the process. Subsequently, we provide a comprehensive analysis of

the recommended movie titles, highlighting their relevance, diversity, and alignment with user preferences. The experimentation results of the recommendation system are shown in Table 4.4. The top-k movie recommendations are presented in a tabular format, showcasing the movie titles along with additional information such as sentiment score (Positive, Negative, or Neutral) with the score in a range of 0 to 1.

Table 4.4: Results of the Top K (k=3) recommended movies based on user query along with sentiment score

User Query	**Top k Recommended Movies**	**Sentiment Score**
”Love romantic movies”	1. Titanic 2. The Notebook 3. La La Land	Positive (0.8)
”Action-packed films”	1. Avengers: Endgame 2. The Dark Knight 3. Mad Max: Fury Road	Positive (0.7)
”Best comedy movies”	1. Superbad 2. The Hangover 3. Dumb and Dumber	Positive (0.6)
”Horror movies”	1. The Conjuring 2. A Nightmare on Elm Street 3. Hereditary	Negative (0.3)
”Classic films”	1. Casablanca 2. Gone with the Wind 3. Citizen Kane	Positive (0.8)
”Sci-fi movies”	1. Star Wars: The Empire Strikes Back 2. Blade Runner 2049 3. Inception	Positive (0.7)

”Animated films”	1. Toy Story 2. Finding Nemo 3. Shrek	Positive (0.8)
”Family-friendly movies”	1. The Lion King 2. Up 3. Frozen	Positive (0.9)
”Indie movies”	1. Juno 2. Little Miss Sunshine 3. Moonlight	Neutral (0.5)
”Documentaries”	1. Fahrenheit 9/11 2. Blackfish 3. March of the Penguins	Positive (0.6)

The experimentation results of the recommended movies by using the context of the given movie as a query vector are shown in Table 4.5. The results show the top k (k=3) recommended movies with their similarity scores. The proposed model is very effective in recommending movies based on the fetched features and the learned RNN model. The results in Table 4.5 show the randomly selected 4 movies as the query with their information of Genres and the index position in the dataset.

Table 4.5: Results of the recommended movies using the query movie along with genres and similarity scores

Query 1	**Movie: Jumanji — Genres: Adventure Fantasy Family — Index: 1**
Recommended Results (k=3)	Return To Oz — Genres: Adventure Family Fantasy — Similarity: 1.0
	Peter Pan — Genres: Adventure Fantasy Family — Similarity: 1.0

	Harry Potter And The Prisoner Of Azkaban — Genres: Adventure Fantasy Family — Similarity: 0.9
Query 2	**Movie: Die Hard — Genres: Action Thriller — Index: 1007**
Recommended Results (k=3)	Iron Eagle Iii — Genres: Action Thriller — Similarity: 1.0
	The Peacemaker — Genres: Action Thriller — Similarity: 1.0
	D-Tox — Genres: Action Thriller — Similarity: 0.9
Query 3	**Movie: Faces — Genres: Drama — Index: 688**
Recommended Results (k=3)	The Hours — Genres: Drama — Similarity: 1.0
	The Graduate — Genres: Drama — Similarity: 0.9
	Coming Apart — Genres: Drama — Similarity: 0.9
Query 4	**Movie: Toy Story — Genres: Animation Comedy Family — Index: 0**
Recommended Results (k=3)	The Wrong Trousers — Genres: Animation Comedy Family — Similarity: 1.0
	A Close Shave — Genres: Family Animation Comedy — Similarity: 1.0
	Creature Comforts — Genres: Animation Comedy Family — Similarity: 0.9

4.3.3 Comparative Analysis

In the comparative analysis, the performance of the proposed model is evaluated on several checkpoints. The initial comparison in the experimentation was in between the performance metrics and their evaluations on the dataset using the proposed NPO

and without the NPO algorithm. The results of the proposed NPO algorithm with performance metrics are presented in Table 4.3 and the performance on the same dataset using the same metrics but without using NPO as a filtering approach is shown in Table 4.6.

Table 4.6: Experimentation results of the model training without NPO algorithm for tweet filtering

Classifier	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**MAP**
Support Vector Machine (SVM)	0.85	0.86	0.84	0.85	0.75
Naive Bayes (NB)	0.82	0.80	0.85	0.82	0.70
Logistic Regression (LR)	0.87	0.88	0.86	0.87	0.78
Recurrent Neural Networks (RNN)	0.90	0.92	0.88	0.90	0.82

By comparing the performance metrics across different optimization and feature selection algorithms, we can assess the effectiveness of the proposed optimization algorithm in filtering data in the MovieLens dataset compared to alternative approaches. Higher values of accuracy, precision, recall, F1-score, and MAP indicate better performance of the algorithm in improving the quality of filtered data for movie recommendation purposes. The experimentation results of the comparison with other benchmark algorithms are shown in Table 4.7.

Table 4.7: Comparative results of the proposed optimization algorithm with other benchmark algorithms on MovieLens dataset using RNN model

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**MAP**
Proposed Optimization	0.94	0.93	0.91	0.92	0.92
Genetic Algorithm [167]	0.87	0.86	0.86	0.86	0.82
Particle Swarm Optimization [167]	0.90	0.88	0.86	0.87	0.89

4.4 Conclusion

In this work, we proposed a framework incorporating the new optimization algorithm inspired by nuclear physics for the filtering of noise in user-centric tweets. The proposed framework is used for the recommendation of movies based on the user query and filtered output from the optimization algorithm in the MovieLens dataset.

The experimentation on optimization-based noise filtering among user-centric tweets to improve predictions in movie recommendation systems, employing classifiers such as Support Vector Machines (SVMs), Naive Bayes classifiers, logistic regression, and Recurrent Neural Networks (RNN), has yielded valuable insights and promising outcomes. Through the proposed optimization based noise filtering approach, we aimed to enhance the quality and relevance of movie recommendations generated from user-centric tweets, thereby enriching the overall movie recommendation experience.

Our findings demonstrate that the optimization-based noise filtering method significantly improves the performance of movie recommendation systems across various classifiers and evaluation metrics. Specifically, we observed notable enhancements in accuracy, precision, recall, F1-score, and Mean Average Precision (MAP) when compared to traditional recommendation approaches without noise filtering. This underscores the effectiveness of our approach in mitigating the impact of noisy data and extracting more accurate and relevant information from user-centric tweets for movie recommendations.

Furthermore, the comparative analysis revealed the superiority of certain classifiers over others in leveraging the filtered tweet data for movie recommendation purposes. While Support Vector Machines (SVMs), logistic regression, and Naive Bayes classifiers exhibited commendable performance across multiple metrics, Recurrent Neural Networks (RNN) demonstrated competitive results in an overall assessment, showcasing the adaptability of our noise filtering approach to different classification algorithms.

The future scope of this work encompasses several avenues for further exploration and enhancement. Firstly, we aim to evaluate and optimize the time complexity of the proposed model to make it more efficient. Secondly, there is a need for continued research into refining the optimization based noise filtering method to accommodate diverse data sources and recommendation contexts beyond movie recommendations. Additionally, incorporating additional features and contextual information from user-centric tweets, such as user engagement metrics, could further enrich the recommendation process.

CHAPTER 5

CONTEXT-AWARE HASHTAG RECOMMENDATION MODEL

In the previous chapter, we proposed a model for movie recommendation using negative samples and Nuclear Physics Optimization technique for filtering noisy tweets. Now, we shift our focus from movie recommendations to hashtag recommendations. Unlike movie recommendation, which is mainly based on user ratings and preferences, hashtag recommendation requires a thorough comprehension of the contextual and dynamic nature of social media content. This chapter puts forth a model intended to capture the meanings associated with words in tweets for recommending relevant hashtags. Our proposed method comprehends the context and meaning behind user-generated content, thus improving the user engagement and interaction on social media platforms.

Using hashtags to summarize thoughts, feelings, emotions, mood swings, food tastes, and much more has become popular. It also symbolizes a variety of things, including locations, families, and friendships. It's a tool for searching and sorting different content on social media platforms. The term "Hashtag Recommendation" originated from the necessity to automate hashtagging due to its increasing prevalence. Furthermore, a large number of posts on social media platforms go untagged. These untagged posts are removed during the search process by applying a label to the data. Such posts are abandoned and add nothing useful to the conversation. However, if the user sees labels that are relevant to his post, he may select one or more of them, labeling the posts in the process. Hashtag recommendation enters the picture in these situations. We have presented a model for hashtag recommendation in this chapter called BELHASH, an LSTM based on Bert Embedding. Because the hashtags are one-hot encoded using MultiLabelBinarizer into multiple binary vectors of zeros and ones, this task is classified as a multilabel classification task. Covid 19 tweets have been used to assess this model.

5.1 Introduction

Deep learning has emerged as a trending approach to be applied on language related domains. DL algorithms can detect trends, sentiments and emerging issues in real-time, providing a deeper understanding of online conversations. Furthermore, they contribute to the development of recommendation systems and content curation, shaping our social media interactions and experiences. Social media has become an important part of our daily life. Plenty of data is available on social media sites which can be used for mining patterns indicating various implications related to health, understanding and assessing the disasters and thus helping in disaster management, analyzing customer feedback, developing marketing strategies, brand communication and monitoring, analyzing social media reviews for various purposes such as election candidate approval ratings, hotel and restaurant ratings.

As the number of Internet users is increasing day by day, the number of posts on social media sites has also increased. Users keep posting their day-to-day life on social media. Among many social media sites, Twitter has become a popular microblogging social site where users interact by posting tweets. Tweets not only describe one's thoughts or life, but also represent the various trends going around in the world at that moment. Twitter data has been used for different purposes such as sentiment analysis, healthcare field, educational sector, predicting elections, and many more.

With the increase in the number of posts on Twitter, there is a need to label these posts. Many users are not aware or have little knowledge of the concept of hashtagging. For such users, recommendation is provided so that they can select the hashtags according to their needs from among the recommended ones. This hashtag recommendation makes the posts labelled from being unlabelled.

A lot of research has been done on healthcare [147,148], but a few years back, Covid-19 pandemic shook the entire world. Almost all the countries were affected economically as well as medically by this infectious virus. Twitter was filled with tweets related to coronavirus and its impact on health. Although many of these tweets are accompanied by hashtags, still many tweets remain untagged. Such untagged

tweets get filtered out while searching and categorizing tweets using a label. A hashtag is a label which summarizes the entire text. As the hashtags are being posted along with the tweets, there is a need to automate this system. Automating means recommending the hashtags based on the tweets as posted by the user. Such recommendation will help the untagged tweets to get a tag from the recommended hashtags. Thus, a need is felt to make a model that can capture the context of the words of tweets and then recommend hashtags.

Recommendation systems are mainly of two types: Content based RS and Collaborative Filtering RS [168]. Content based RS mainly focus on the users' content and then recommending the tags according to that content. Whereas, Collaborative Filtering RS makes use of other users' tagging history with the similar interests of posts as the current user is having. In this work, we have focussed on Content Based RS. This task is considered as a Multilabel Classification task. In the context of multilabel classification, the hashtag recommendation task can be seen as predicting a set of relevant hashtags for a given text input. Each hashtag acts as a distinct label, and the goal is to determine which hashtags are most suitable for describing or categorizing the content of the text. Hashtags are one-hot encoded into multiple binary vectors [169], where multiple tags can be set to 1 for a given tweet.

We have presented a model for hashtag recommendation in this chapter called BELHASH, an LSTM based on Bert Embedding. 100K Covid-19 tweets were used to evaluate this model. The process flow diagram is shown in Fig.5.1. Combining LSTM with BERT in the hashtag recommendation task serves a unique purpose. Coupling these two potent models provides complimentary benefits, improving the model's capacity to capture global semantics and local context—that is, BERT's comprehensive learning of the text and LSTM's capacity to record temporal information. The proposed model can successfully represent both short- and long-term interdependence because of this fusion.

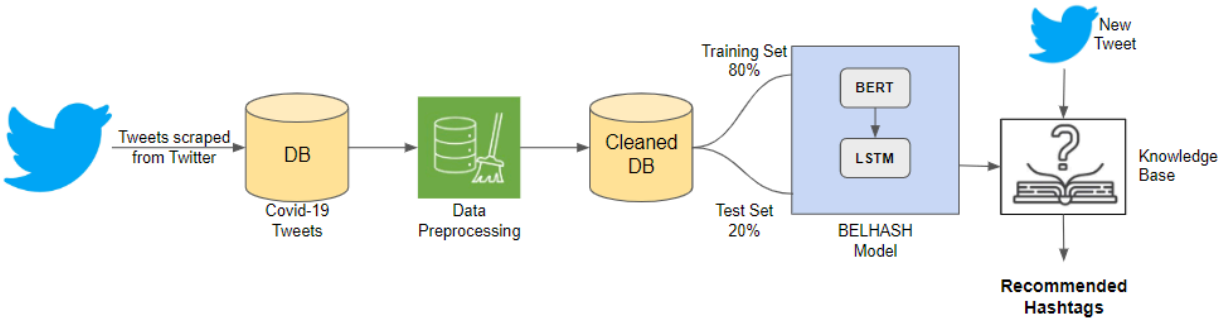

Fig.5.1: Flow diagram of data preprocessing leading to recommended hashtags

The main contributions of this chapter are:

- We have proposed a model named BELHASH to recommend Hashtags.
- Proposed model is experimented with 100K Covid-19 related tweets.
- Proposed model is composed of a cascading of BERT-LSTM for relevant text feature extraction. This model achieves state of the arts results.

5.2 Methodology

In this section, we describe our proposed model, BELHASH (Bert Embedding based LSTM for Hashtag Recommendation), in detail followed by the various terminologies used for our work.

5.2.1 Proposed Work

BELHASH is a Bert Embedding [170] based LSTM model [171] for Hashtag Recommendation. The architecture of the proposed model is shown in Fig.5.2. This model has mainly five components. The first component is hashtag encoding where hashtags are one-hot encoded into a binary vector of 0s and 1s. Here, 0 represents the absence of a hashtag for that tweet whereas 1 represents the presence of a hashtag for that tweet. The second component is word tokenization where tweets are tokenized into multi-dimensional vectors using BERT tokenizer [170]. The third component comprises feature extraction by LSTM. The fourth component is Part-of-speech (POS) tagging where tags are given a probability of being associated with a tweet. The last component is hashtag recommendation where cosine similarity [163] is

applied between all tags and encoded vector representation of a new tweet. Those tags having high similarity are given a ranking according to the probability distribution of cosine similarity function. These hashtags are then sorted in the descending order according to their probabilities, recommending top-k hashtags. Now, each of these components are described below in detail.

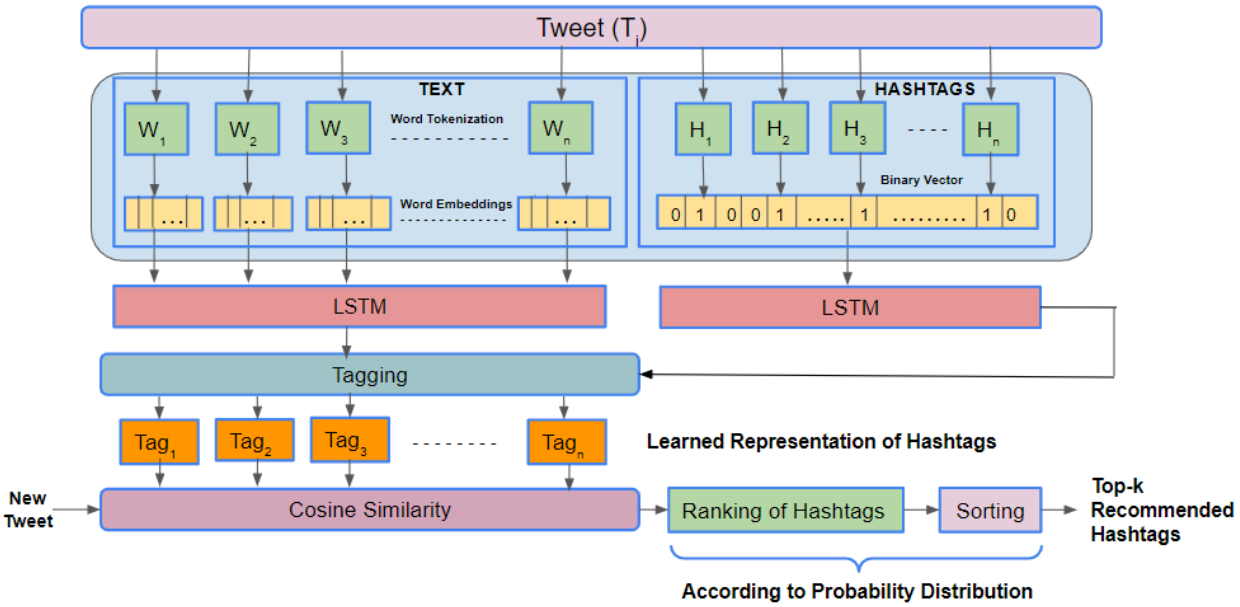

Fig.5.2: Workflow of the proposed BELHASH model

5.2.1.1 Hashtag Encoding

We have scraped 100K Covid-19 tweets using Twitter API. This tweets dataset has 25000 unique hashtags. Out of these, we have filtered out 1350 hashtags having frequency of occurrence greater than or equal to 10. These hashtags are encoded using MultiLabelBinarizer (MLB). MLB is used to encode the presence or absence of each hashtag for a given tweet. It creates a binary vector where each element corresponds to whether a specific hashtag is present in the tweet or not. This vector is often referred to as a "binary label vector." Thus, through MLB, a binary vector is created for a given tweet containing zeros and ones. If a hashtag is present in the tweet, the corresponding element in the binary vector is set to 1; otherwise, it's set to 0. Let’s take an example:

total_hashtags: ['covid19', 'home', 'health', 'staysafe', ...]
tweet_hashtags: ['covid19', 'health']
binary vector: [1, 0, 1, 0, ...]

Here, for the given tweet, the element at index 0 represents the presence of 'covid19', the element at index 1 represents the absence of 'home', the element at index 2 represents the presence of 'health', the element at index 3 represents the absence of 'staysafe' and so on. During training, the binary label vectors are used as target values, and during inference, the predicted binary vectors are decoded to obtain the recommended hashtags for a given tweet.

5.2.1.2 Word Tokenization and Embedding

Word tokenization is the second step of text processing tasks. In this step, each tweet goes through the process of word tokenization. It is a method to split the words in the form of tokens or unbreakable units. Each word is then given a unique number which is represented by a multi dimensional vector. We have used Bert tokenizer for this purpose. Bert tokenizer captures the context of the same words according to the sentences in which they are appearing. If a word is appearing with two different semantics, then that word will be represented by two different vectors. In contrast, other embedding techniques like Word2Vec, Glove, generate the same vector representation for the same word, even if it is appearing in two different contexts. This is because other embedding techniques do not capture the semantics of the words according to their occurrences.

5.2.1.3 Feature Extraction by LSTM

Long Short Term Memory (LSTM) is a variation of Recurrent Neural Networks (RNN). LSTM resolves the vanishing gradient problem of RNN by retaining the previous words in the memory to generate the next outputs in the sequences. In the context of hashtag recommendation, the LSTM neural network plays a crucial role in extracting meaningful features from the input text. These features are essential for capturing the underlying semantics and context of the text, which in turn aids in

making accurate hashtag predictions. Simultaneously, the hashtags associated with each tweet are one-hot encoded and processed through a separate LSTM layer.

The LSTM learns to associate specific patterns, phrases, and contextual cues with relevant hashtags. For instance, consider the sentence "Stay safe and wash your hands regularly. #COVID19 #health." The LSTM can learn that the presence of phrases like "wash your hands" and "stay safe" are indicative of the hashtags "#COVID19" and "#health." These associations are learned over time through the iterative process of training. Thus, LSTM extract the features in a text processing task effectively.

5.2.1.4 POS Tagging

The textual features of input tweets are extracted through LSTM networks, capturing the contextual nuances and sequential dependencies within the text. Simultaneously, the hashtags associated with each tweet are one-hot encoded and processed through a separate LSTM layer. The outputs from these two distinct LSTM pathways are then fused in a tagging layer, where the model learns to generate relevant part-of-speech (POS) tags based on the intricate interplay between the textual content and the hashtag information. The tagging layer serves as a bridge between the textual and hashtag domains, synthesizing meaningful representations that encapsulate the essence of the input tweet.

5.2.1.5 Hashtag Recommendation

At last, Cosine Similarity is applied between all POS tags and encoded vector representation of a new tweet. Those tags having high similarity are given a ranking according to the probability distribution of cosine similarity function. These hashtags are then sorted in the descending order according to their probabilities. Then top-k recommendations are made.

5.2.2 Terminologies Used

The various terminologies used are given below followed by two algorithms of BELHASH. Algorithms 5.1 and 5.2 describe the hashtag and text processing parts of the tweet respectively.

$T = \{T_1, T_2, \ldots. T_i \ldots T_n\}$: Set of Cleaned Tweets, where T_i is the tweet at the i^{th} location and n is the total number of tweets.

$H = \{H_1, H_2, \ldots. H_i \ldots H_n\}$: Set of hashtags for a Tweet T, where H_i is the hashtag at the i^{th} location and n is the total number of hashtags.

$H_R = \{H_{R1}, H_{R2}, \ldots. H_{Ri} \ldots H_{Rk}\}$: Set of recommended hashtags for a Tweet T, where H_{Ri} is the hashtag at the i^{th} location according to the rank probability and k is the number of recommended hashtags.

$T_i(w)$: Word Tokenization of tweet T_i

$T_i(E)$: Encoded vector representation of tokenized words of tweet T_i

$T_i(F)$: Features extracted for tweet T_i

$P = \{H_{P1}, H_{P2}, \ldots. H_{Pi} \ldots H_{Pn}\}$: H_{Pi} is the Probability distribution (Ranking) of the hashtag H_i for a given tweet T.

$S = \{H_{P1}, H_{P2}, \ldots. H_{Pi} \ldots H_{Pn}\}$: Set of hashtags of tweet T_i sorted according to their probability distributions. H_{P1} is the hashtag with the highest probability. H_{P2} is the hashtag with the second highest probability and so on.

$FH = \{FH_1, FH_2, \ldots. FH_i \ldots FH_n\}$: Set of all 1350 filtered hashtags where FH_i is the hashtag at the i^{th} location.

V: Binary vector of all Hashtags(H) for a tweet T_i

V(F): Features extracted for Binary Vector V of all Hashtags(H) for a tweet T_i

5.3 Evaluation and Results

In this section, we provide the details to the dataset and its preprocessing, implementation details and comparison with baseline models.

Algorithm 5.1: HASHTAG_PROCESSING(FH, H)

Input: Set of hashtags (H) for a tweet T_i and all filtered hashtags (FH)
Output: Extracted features of Binary Vector (V(F))
1. **Procedure** HASHTAG_PROCESSING(FH, H)
2. Transformed_labels (TL) ← MLB(FH)
3. **for** each FH_i in TL do
4. **for** each $H_i \in H$ do
5. **if** $FH_i == H_i$, then
6. Set $V[FH_i] = 1$
7. **else**
8. Set $V[FH_i] = 0$
9. **end if**
10. **end**
11. **end**
12. Extracted_features (V(F)) ← LSTM (V)
13. Return V(F)

Algorithm 5.2: TEXT_PROCESSING(T, H)

Input: Set of Tweets (T) and its hashtags (H)
Output: Top-k recommended hashtags (H_R)
1. **Procedure** TEXT_PROCESSING(T, H)
2. **for** each tweet T_i do
3. $T_i(w)$ ← bert-based-uncased-tokenizer (T_i)
4. $T_i(E)$ ← bert-based-uncased-model-encoder ($T_i(w)$)
5. Extracted_features ($T_i(F)$) ← LSTM ($T_i(E)$)
6. Part-of-speech Tags (POST) ← Tagging ($T_i(F)$ + V(F))
7. Probability distribution (P) ← Cosine_Similarity (POST)
8. Sorted probabilities (S) ← Sorting (P)
9. H_R ← Pick Top-k recommendations from S
10. **end**
11. return H_R

5.3.1 Dataset

We have evaluated our proposed model on the Covid-19 dataset, which is scraped using Twitter API. A sample of collected tweets is shown in Fig.5.3.

text	hashtags
If I smelled the scent of hand sanitizers today on someone in the past, I would think they were so intoxicated thatâ€¦ https://t.co/QZvYbrOgb0	
Hey @Yankees @YankeesPR and @MLB - wouldn't it have made more sense to have the players pay their respects to the Aâ€¦ https://t.co/1QvW0zgyPu	
@diane3443 @wdunlap @realDonaldTrump Trump never once claimed #COVID19 was a hoax. We all claim that this effort toâ€¦ https://t.co/Jkk8vHWHb3	['COVID19']
@brookbanktv The one gift #COVID19 has give me is an appreciation for the simple things that were always around meâ€¦ https://t.co/Z0pOAIFXcW	['COVID19']
25 July : Media Bulletin on Novel #CoronaVirusUpdates #COVID19 @kansalrohit69 @DrSyedSehrish @airnewsalerts @ANIâ€¦ https://t.co/MN0EEcsJHh	['CoronaVirusUpdates', 'COVID19']
#coronavirus #covid19 deaths continue to rise. It's almost as bad as it ever was. Politicians and businesses wantâ€¦ https://t.co/hXMHooXX2C	['coronavirus', 'covid19']
How #COVID19 Will Change Work in General (and recruiting, specifically) via/ @ProactiveTalent #Recruitingâ€¦ https://t.co/bjZxzGPMbK	['COVID19', 'Recruiting']

Fig.5.3: A sample of raw/ unprocessed Covid-19 tweets

We performed data preprocessing so as to remove noisy data. Several steps of data preprocessing are described below:

1. Non-Hashtags Tweets Removal - We removed the tweets containing no hashtags and kept the tweets containing at least one hashtag for a given tweet.
2. Non-English Tweets and Unicode Characters Removal - We removed non-english characters from tweets and non-english hashtags. We also removed tweets and hashtags containing unicode characters.
3. URL, HTML and Punctuations Removal - In this step we removed tweets containing URL patterns starting from http(s), email IDs and punctuation marks.
4. Emojis Removal - This step is concentrated on removing emoticons, symbols & pictographs, transport & map symbols, and flags.
5. Conversion to Lowercase - In this step, we convert all the tweets and their associated hashtags to lowercase so that the same hashtags but in uppercase can be treated as one.
6. Hashtag Removal - In this step, we removed all the hashtags appearing at the end of the tweets but retained the ones appearing in between the tweets. Generally, hashtags are present at the end, describing the overall theme of the tweet. So, we have removed them so as to predict hashtags from the context of the tweet.
7. Lemmatization - We lemmatized all tweets and hashtags so that the variant forms of one word can be treated as a single word.

8. Filter Candidate Tweets - In this step, tweets with very short length compared to its numbers of hashtags, are filtered out based on a threshold.

   If length(tweets) / Number of Hashtags < Threshold => drop that tweet

9. Drop Duplicate Tweets - For duplicate tweets, we have kept only the first tweet and removed the duplicates.
10. Hashtag Filtration - Hashtags having frequency of occurrence greater than or equal to 10 in the entire dataset are kept. So, out of 25K tags, 1350 tags are kept.

After preprocessing, we split our dataset into 80% and 20% for the training and testing respectively. A sample of the cleaned dataset is shown in Fig.5.4. Table 5.1 shows the dataset details.

	text	hashtags
0	diane3443 wdunlap realdonaldtrump trump never once claim covid19 be a hoax . we all claim that this effort to	['covid19']
1	brookbanktv the one gift covid19 have give me be an appreciation for the simple thing that be always around me	['covid19']
2	25 july : medium bulletin on novel coronavirusupdates covid19 kansalrohit69 drsyedsehrish airnewsalerts ani	['coronavirusupdates', 'covid19']
3	coronavirus covid19 death continue to rise . it s almost as bad a it ever be . politician and business want	['coronavirus', 'covid19']
4	how covid19 will change work in general and recruiting , specifically via proactivetalent	['covid19', 'recruiting']
5	pray for good health and recovery of chouhanshivraj .	['covid19', 'covidpositive']
6	pope a god prophet sadhu sundar selvaraj . watch here at	['hurricanehanna', 'covid19']

Fig.5.4: A sample of pre-processed Covid-19 tweets

Table 5.1: Dataset Values before and after pre-processing the tweets

Dataset Parameters	**Values**
Total fetched Tweets	1.65 Lakhs
Pre-processed Tweets	1 Lakh
Unique hashtags before preprocessing	25000
Unique hashtags after preprocessing	1350
Training Dataset	80K Tweets
Test Dataset	20K Tweets

5.3.2 Implementation Details

Pytorch library of Python is used to perform the experiments. The dataset is divided into 80% training set and 20% test set. All the tweets are padded to the maximum length of tweets i.e. 280 characters, by padding extra tokens (zeros), to ensure uniformity in the input data. Both the training set and test set are divided into batches each of 32 size. For the training set, we shuffle the tweets but not for the test set. We have used 0.05 as the Learning Rate and Adam as the optimizer. The hyperparameters and their values are given in Table 5.2.

Table 5.2: Hyperparameters names and their values used in BELHASH

Parameter Name	**Parameter Value**
Tokenizer	'bert-base-uncased'
Max_Padding	280
Batch Size	32
Epochs	50
Dropout	0.3
Learning Rate	0.05
Optimizer	Adam
Loss function	BCEWithLogitsLoss

5.3.2.1 Model Training

The Bert-based-uncased Pretrained model is used to tokenize the tweets. Then this tokenized form is encoded into a vector representation capturing the semantics of the words. LSTM layer is applied over this vector representation of tweets to extract the features. After this, a Dropout of value 0.3 is used to regularize the weights in the

backpropagation process. Then the last layer of Softmax is applied to get the learned representation of hashtags of a given tweet.

5.3.2.2 Model Testing

For a new tweet, Bert embedding and LSTM layers are applied to have features extracted from encoded vector representation for that tweet. Then, Cosine Similarity is applied for that encoded representation of the new tweet and the learned representation of hashtags obtained in the training step. Cosine similarity gives a probability distribution (rank) of hashtags to be recommended for the new tweet. Then, sorting is applied over the rank of the hashtags, arranging them in a descending sequence in order to recommend top-k hashtags.

5.3.3 Baseline models

To compare our model with other models, following baseline models are considered:

- Latent Dirichlet Allocation: This method extracts latent topics from social media content and associates hashtags with these topics [149].
- SVM: The method proposed in [150] converts the tag recommendation problem into classification task and uses SVM to model it.
- EmHash: Bert Embedding based model forming clusters of hashtags and then recommending hashtags by predicting the desired cluster [151].

5.3.4 Evaluation Results

The results obtained after training and evaluating the BELHASH model over the scraped tweets are given in Table 5.3. It achieved an accuracy of 72.31%. The graphs of various benchmark models compared with our model are given in fig.5.5, fig.5.6 and fig.5.7. Precision, Recall and F1-Scores for top-k (k=6) recommendations are given in table 5.4 and its graph is shown in Fig.5.8. The results show that BELHASH performs better than other state-of-the-art models.

Table 5.3: Comparative analysis of evaluation results of different methods for hashtag recommendation

Models	Precision	Recall	F1-Score
Latent Dirichlet Allocation	9.8%	7.8%	8.7%
SVM	23.8%	20.3%	21.9%
EmHash	15%	46%	22%
BELHASH	**70%**	**66%**	**67%**

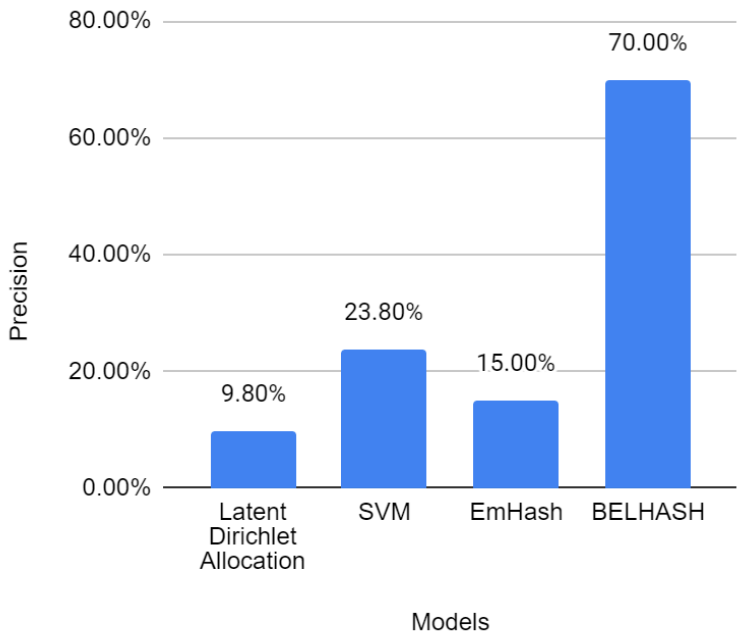

Fig.5.5: Comparison of BELHASH with other models with respect to Precision

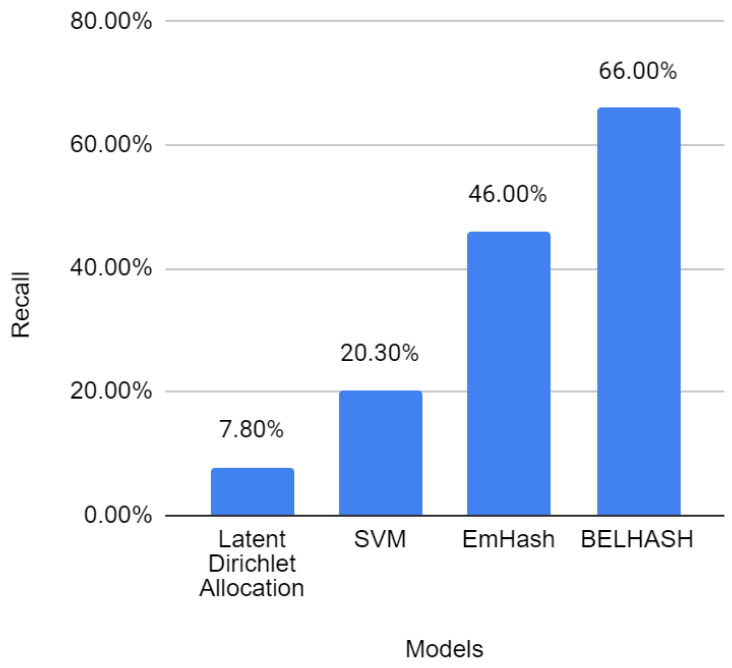

Fig.5.6: Comparison of BELHASH with other models with respect to Recall

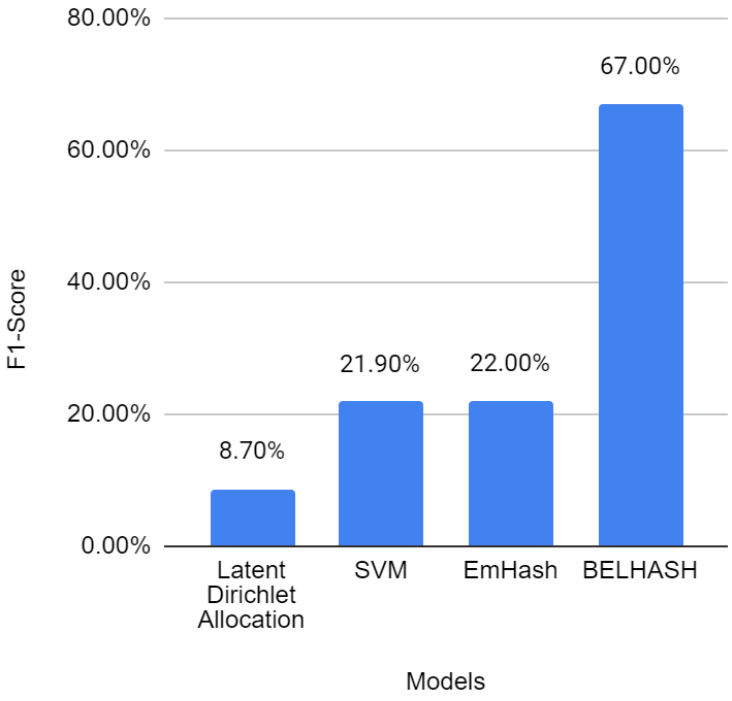

Fig.5.7: Comparison of BELHASH with other models with respect to F1-Score

Table 5.4: Precision, Recall and F1-Scores for top-k (k=6) recommendations

	k=1	**k=2**	**k=3**	**k=4**	**k=5**	**k=6**
Precision	0.66	0.4	0.5	0.71	1	0.92
Recall	0.7	0.4	0.5	0.92	0.63	0.8
F1-Score	0.69	0.4	0.5	0.8	0.77	0.86

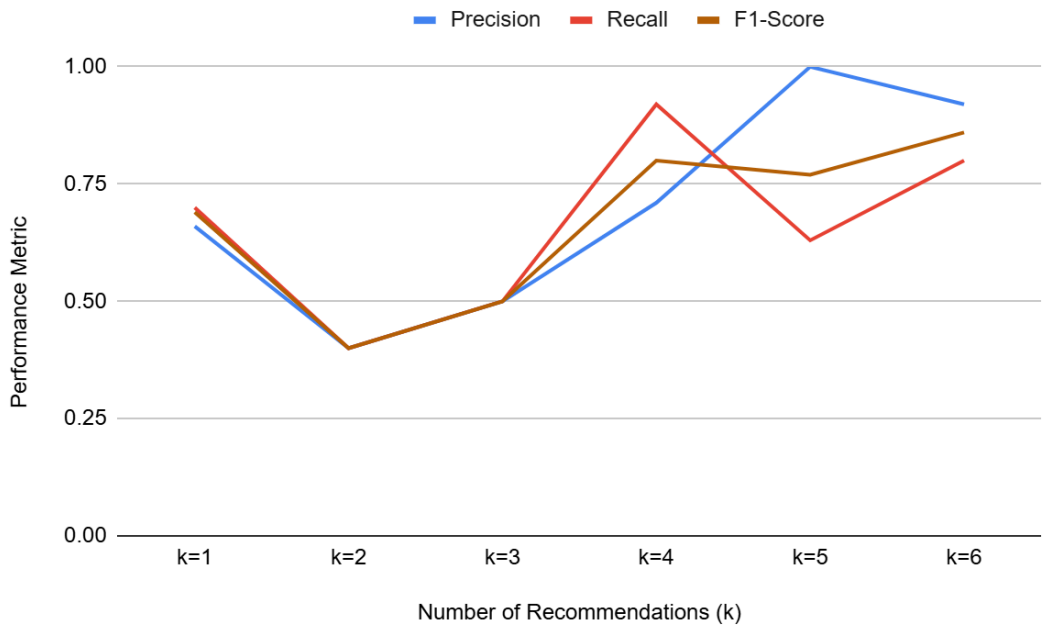

Fig.5.8: Values of evaluation Metrics for top-k (k=6) recommendations

5.4 Conclusion

In this paper, we have proposed a novel model named BELHASH, which stands for Bert Embedding based LSTM for Hashtag Recommendation. This model is evaluated on 100K covid-19 tweets, scraped using Twitter API. Existing works related to Hashtag recommendation comprises other embeddings like Word2Vec, GLOVE etc. But these embeddings do not capture the semantics or the context of the words and thus generate the same embeddings for different appearances of a word even if the word is used for different contexts in different appearances. Bert Embedding, on the

other hand, captures the context of the same word in different perspectives and thus generates different embeddings. Bert combined with LSTM works effectively and generates good results. In experimentation, we achieved the best performance metrics as precision 100% on top-5, recall as 92% on top-4 and F1-score as 86% on top-6 recommendations. The average of these values are Precision-70%, Recall-66% and F1-Score-67%. The overall accuracy of the proposed methodology is 72%. The results show that BELHASH performs better than other state-of-the-art models. In the future, we aim to evaluate and optimize the time complexity of the proposed model to make it more efficient. We also plan to extend this work for capturing the trending and dynamic nature of tweets and thus recommending the hashtags according to the on-going trends.

CHAPTER 6

EVALUATION METRICS

After the implementation of deep learning models for movie and hashtag recommendations in the previous chapters, we now proceed towards exploring the various performance evaluation metrics. Recommender systems have flooded the internet with their capability to attract more users. Social media sites like Facebook recommend posts and friends based on one's interests. E-commerce sites like Amazon, Flipkart recommend items to the users based on their history. Entertaining sites like YouTube, Netflix recommend content based on users' likings. Thus, implementing the appropriate model for the problem of recommendation is the first challenging task. Many companies invest majorly to improve the performance of their recommendation system so as to increase sales. So, now when a lot of research is being done to improve RS from time to time, there is a question that arises, how well the RS is working. That means, we need to measure its quality using some suitable evaluation metrics. So, choosing the appropriate evaluation metrics is another challenging task. In this chapter, we provide a glimpse of various evaluation metrics so that one can have an idea what metrics to use for evaluating the performance of their model.

6.1 Introduction

Almost all online websites are working through recommendations. The more they recommend quality products, the more their revenue is going to increase. RS is applied in daily life in various applications such as e-commerce, e-learning and education, tourism, healthcare, software engineering and many more [172]. Implementing the appropriate model for your problem of recommendation is the first challenging task. But, choosing the right metric to evaluate that recommender model is another challenging task. Sometimes, it may happen that selecting the wrong metric to evaluate the model may give worse results even if the model is correctly designed. So, it is important to know which metric is to be used while measuring the

performance of the system. In this chapter, we have explained almost all the evaluation metrics that are so far used for evaluating the quality of the recommendation models. According to the Fig. 6.1, we have distributed the various evaluation metrics in four divisions: Based on Confusion Matrix, Based on Error, Based on Ranking and Based on Linguistics.

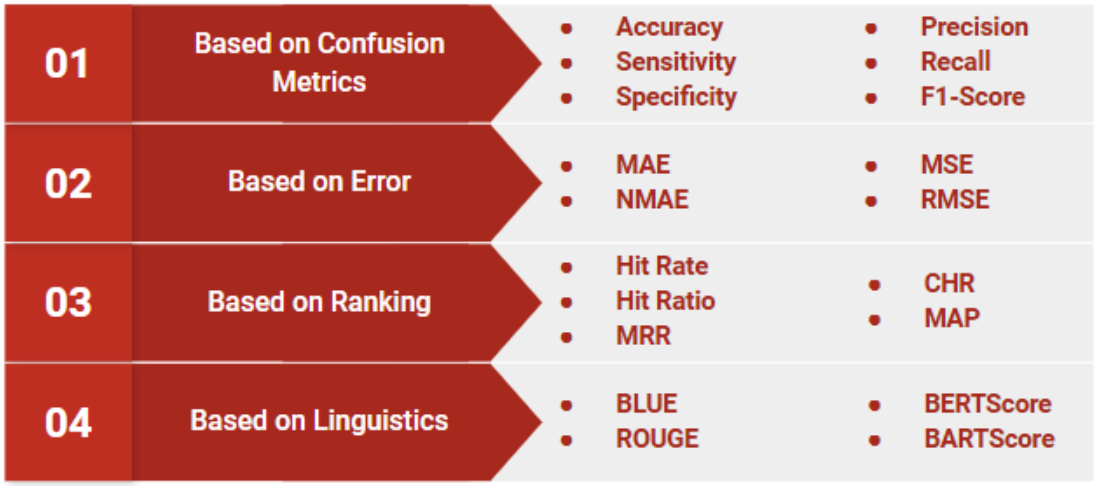

Fig.6.1: Various Evaluation Metrics

The main contributions of the chapter are:

- This chapter describes almost all the evaluation metrics that have been used and can be used in the future for the recommender systems.
- This chapter describes the various categories of the evaluation metrics, letting one know which metric to be used accordingly.
- We propose a novel evaluation metric named Semantic Recommendation Score (SRS), specifically designed for recommender systems, which takes the advantages of both BERTScore and BARTScore.

6.2 Metrics based on Confusion Matrix

Confusion Matrix [173] summarizes the performance of the recommendation model. It is well explained through Table 6.1.

Table 6.1: Confusion Matrix

		Predicted Items	
		Recommended	Not Recommended
Actual Items	Liked Items	True Positive (TP)	False Negative (FN)
	Not Liked Items	False Positive (FP)	True Negative (TN)

TP - It is recommended and liked by the user.

TN - It is not recommended and it is not liked by the user.

FP - It is recommended but not liked by the user.

FN - It is not recommended but liked by the user.

Consider an example:

Let's assume that there are the following 10 fruits in the dataset: Apple, Banana, Orange, Pineapple, Strawberry, Mango, Grapes, Guava, Kiwi, and Papaya.

Items liked by the user: Banana, Grapes, Apple and Orange

Recommended Items: Apple, Papaya, Banana, Orange and Mango

Here, TP = 3, FP = 2, TN = 4, FN = 1

The evaluation metrics, based on the confusion matrix, are explained below, considering the above example.

6.2.1 Accuracy - It is a very basic metric, defined as the proportion of correct predictions made by the model to all the predictions. It is computed using the equation 6.1.

$$\text{Accuracy} = (TP + TN) / (TP + FP + TN + FN) \quad (6.1)$$

Accuracy = (3 + 4) / (3 + 2 + 4 + 1) = 7/10 = 0.7 ~ 70%

6.2.2 Sensitivity - It is defined as the correctly recommended items that are liked or purchased by the users. It is computed using the equation 6.2.

$$\text{Sensitivity} = TP / (TP + FN) \quad (6.2)$$

Sensitivity = 3 / (3 + 1) = 3/4 = 0.75 ~ 75%

6.2.3 Specificity - It is mainly used to remove noise from recommendation systems. It is used to identify the non-recommended items that should not be liked by the user. It is computed using the equation 6.3.

Specificity = TN / (TN + FP) (6.3)

Specificity = 4 / (4 + 2) = 4/6 = 0.67 ~ 67%

6.2.4 Precision - It is defined as the fraction of relevant recommended items over all recommended items. Here, the reference is taken as recommended (predicted) items. It is computed using the equation 6.4.

Precision = TP / (TP + FP) (6.4)

Precision = 3 / (3+2) = 3/5 = 0.6 ~ 60%

6.2.5 Recall - It is defined as the fraction of relevant recommended items over all items. Here, the reference is taken as the items liked by the user. It is computed using the equation 6.5.

Recall = TP / (TP + FN) (6.5)

Recall = 3 / (3+1) = 3/4 = 0.75 ~ 75%

6.2.6 F1 Score - It is defined as the Harmonic Mean of Precision and Recall. It is computed using the equation 6.6.

F1 = (2 * Precision * Recall) / (Precision + Recall) (6.6)

F1 = (2 * 0.6 * 0.75) / (0.6 + 0.75) = 0.9 / 1.35 = 0.66 ~ 66%

6.2.7 Precision and Recall are Equal

Let's consider another example from the same database.

Items liked by the user: Apple, Orange, Banana and Mango

Recommended Items: Banana, Grapes, Apple and Orange

Here, TP = 3, TN = 5, FP = 1, FN = 1

Precision = TP / (TP + FP) = 3 / (3 + 1) = 3/4 = 0.75 ~ 75%

Recall = TP / (TP + FN) = 3 / (3 + 1) = 3/4 = 0.75 ~ 75%

F1 = (2 * 0.75 * 0.75) / (0.75 + 0.75) = 1.125 / 1.5 = 0.75 ~ 75%

We see that when Precision and Recall are equal, then F1-Score also becomes the same.

6.2.8 Top-k recommendations

There is a case when recommendation systems provide a filter to top-k recommendations, where k is the number of items in the ranked order. Consider the below example:

Items liked by the user: Banana, Grapes, Apple and Orange

Top-3 Recommended Items: Apple, Papaya and Banana

Here, TP = 2, FP = 1, TN = 5, FN = 2

There is another version of precision, recall and F1 score as precision@k, recall@k and F1@k, k is the number of items in the top-k recommender list.

Precision@3 = TP / (TP + FP) = 2 / (2+1) = 2/3 = 0.67 ~ 67%

Recall@3 = TP / (TP + FN) = 2 / (2+2) = 2/4 = 0.5 ~ 50%

F1@3 = (2 * 0.67 * 0.5) / (0.67 + 0.5) = 0.67 / 1.17 = 0.57 ~ 57%

6.3 Error based Metrics

Sometimes, it may happen that users leave no rating for the products they have liked. Thus, the ratings database is almost empty or filled with NaNs (NaN is used for missing data). Now to measure the performance of the recommender systems, one has to predict the ratings. In such cases, where the database is sparse and the ratings have to be predicted, error based metrics [173] are used.

6.3.1 Mean Absolute Error (MAE) - It is used when the database doesn't have many outliers. When the outliers don't affect the rating prediction, then MAE can be used, as it gives equal weight to the outliers as to the relevant data. It doesn't penalize the errors during prediction. It is basically the average magnitude of difference between the actual rating and the predicted rating. It measures the average of the residuals in the dataset. It is computed using the equation 6.7.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |Actual\ Rating_i - \ Predicted\ Rating_i| \tag{6.7}$$

where, n is the number of items in the database.

6.3.2 Normalized Mean Absolute Error (NMAE) - It is a variation of MAE in which the average of mean error is normalized over the average of all actual ratings.

6.3.3 Mean Squared Error (MSE) - Unlike MAE, MSE doesn't give equal weight to the outliers, i.e. it penalizes the errors during prediction. It is used when there are many outliers in the database. It is defined as the average of the square of difference between actual rating and the predicted rating. It basically measures the variance of the residuals in the dataset. It is computed using the equation 6.8.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (Actual\ Rating_i - \ Predicted\ Rating_i)^2 \quad (6.8)$$

6.3.4 Root Mean Squared Error (RMSE) - It is the square root of MSE. It measures the standard deviation of the residuals in the dataset. RMSE is preferred over MSE as it is measured in the same units as the predicted values, whereas, MSE is measured in the squared units of the predicted values. It is computed using the equation 6.9.

$$\text{RMSE} = \sqrt{MSE} = (1/n) * \sqrt{\sum_{i=1}^{n} (Actual\ Rating_i - \ Predicted\ Rating_i)^2} \quad (6.9)$$

6.4 Metrics based on Ranking

Sometimes, users are in a hurry while browsing through the items. In such cases, recommending a lot of items will not make much profit, but recommending top-k items will make a difference. This top-k recommendation is made through ranking the items according to their ratings. Thus, users can like any item among these top-k recommendations as they are better aligned with the interests of the user. The metrics based on ranking [174] are explained below:

6.4.1 Hit Rate - If the recommended item in top-k recommendation is relevant to the user or liked by the user, then it is a hit, otherwise, miss. Hit Rate is defined as the ratio of number of hits per user. It is used when evaluating the proportion of relevant items that are successfully recommended. It is computed using the equation 6.10.

Hit Rate = Number of hits / Number of users (6.10)

6.4.2 Hit Ratio - It is defined as the fraction of items in the top-k recommendation that are relevant to the users. It is used when assessing the likelihood of at least one relevant item being recommended to users. It is computed using the equation 6.11.

Hit Ratio = (# relevant items in top-k recommendations / # all relevant items) (6.11)

where, # represents the term "number of"

6.4.3 Cumulative Hit Rate (CHR) - In this, those hits are removed if they have a predicted rating below than some threshold value. It is used when evaluating recommendation systems based on the relevance of items at different positions in the recommendation list.

6.4.4 Mean Reciprocal Rank (MRR) - It is also known as Average Reciprocal Hit Rank (ARHR). It is used when prioritizing the rank of the first relevant item in the recommendation list. It is defined as the sum of the reciprocal of rank of each hit over the users. It is computed using the equation 6.12. It is used in 2 cases:

- If there is only one relevant item in the top-k recommendations
- If first item is relevant in the top-k recommendations

Let's say, if the model has recommended top-10 items, and if the item ranked 4th is the relevant item, then MRR does not care about the items ranked through 5 to 10.

$$\text{MRR} = \frac{1}{\#Users} \sum_{i=1}^{\#\,Hits} \left(\frac{1}{Rank_i}\right) \quad (6.12)$$

6.4.5 Mean Average Precision (MAP) - It is defined as the mean of average precision. Average Precision (AP) is the average of precision of one user. So, MAP is the average of AP of all users. It considers the recommendations in the ranked order. It is used when considering the precision of recommendations at multiple points in the recommendation list. It is computed using the equation 6.13.

$$\text{MAP} = \frac{1}{U} \sum_{i=1}^{U} (AP_i) \quad (6.13)$$

Here, AP_i represents the Average Precision of i^{th} user and U represents the number of users in the database. Consider fig.6.2 and fig.6.3 for this purpose. Fig.6.3 takes top-3 recommendations into account. There are 7 items in the dataset: Apple, Banana, Orange, Strawberry, Mango, Grapes and Pineapple.

	User 1		User 2		User 3	
Rank	**Liked Items**	**Recommended Items**	**Liked Items**	**Recommended Items**	**Liked Items**	**Recommended Items**
1	Apple	Banana	Strawberry	Banana	Pineapple	Banana
2	Banana	Apple	Mango	Apple	Apple	Apple
3	Orange	Mango	Grapes	Strawberry	Strawberry	Strawberry
4	Pineapple	Grapes	Pineapple	Pineapple	Banana	Orange
5	Mango	Strawberry	Orange	Grapes	Orange	Mango

Fig.6.2: Items liked by the various users and the items recommended by the model

	User 1		User 2		User 3	
Rank	**Liked Items**	**Recommended Items**	**Liked Items**	**Recommended Items**	**Liked Items**	**Recommended Items**
1	Apple	Banana	Strawberry	Banana	Pineapple	Banana
2	Banana	Apple	Mango	Apple	Apple	Apple
3	Orange	Mango	Grapes	Strawberry	Strawberry	Strawberry
4	Pineapple	Grapes	Pineapple	Pineapple	Banana	Orange
5	Mango	Strawberry	Orange	Grapes	Orange	Mango

Fig.6.3: Items liked by the various users and the Top-3 items recommended by the model

Precision@k for the users of fig.6.2 and fig.6.3 are given in Table 6.2.

Table 6.2: Precision@k for users of fig.6.2 and fig.6.3

User 1		**User 2**		**User 3**	
Fig.6.2	Fig.6.3	Fig.6.2	Fig.6.3	Fig.6.2	Fig.6.3
TP = 3	TP = 2	TP = 3	TP = 1	TP = 4	TP = 2
FP = 2	FP = 1	FP = 2	FP = 2	FP = 1	FP = 1
Precision@5 = 3/5 = 0.6	Precision@3 = 2/3 = 0.67	Precision@5 = 3/5 = 0.6	Precision@3 = 1/3 = 0.33	Precision@5 = 4/5 = 0.8	Precision@3 = 2/3 = 0.67
AP_1 = (0.6+0.67)/2 = 0.635		AP_2 = (0.6+0.33)/2 = 0.465		AP_3 = (0.8+0.67)/2 = 0.735	

$MAP = (AP_1 + AP_2 + AP_3) / 3 = (0.635 + 0.465 + 0.735) / 3 = 0.61 \sim 61\%$

6.5 Metrics based on Linguistics

Recently, there is a lot of research done on language processing. Sequence-to-sequence tasks like summarization of text, text optimization, responding to questions, chatbots to communicate and AI for translation are all part of this processing. To evaluate such models, 2 metrics popularly being used are BLEU[152] and ROUGE[153]. These metrics work on the concept of n-grams. n-grams are the sequence of words of length n. Unigram (1-gram) is the sequence of words of length 1, bigram (2-gram) is the sequence of words of length 2. Similarly, n-grams are sequences of words of length n. Both BLEU and ROUGE only consider the syntactic structures of the generated/recommended text and have the disadvantage that they don't consider semantics of the generated text. Another metric named BARTScore [154] assesses the linguistic fluency and coherence of the generated text. But, it also has the same disadvantage as BLEU and ROUGE. To overcome the above-mentioned drawback, BERTScore [155] is used which considers both syntactic and semantics of the generated text.

6.5.1 Bilingual Evaluation Understudy (BLEU) - It is measured using the following equation 6.14.

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n log_e p_n\right) \quad (6.14)$$

exp → exponential

w_n → weight between 0 and 1 for $log_e p_n$

p_n → precision of n-grams

BP → brevity penalty is defined by equation 6.15.

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ \exp(1 - (r/c)) & \text{if } c <= r \end{cases} \quad (6.15)$$

c → the number of unigrams in the recommended text

r → best match length between the reference text / ground truth and the recommended text

6.5.2 Recall-Oriented Understudy for Gisting Evaluation (ROUGE) - It is a variation of BLEU which focuses on recall as well rather than only on precision. It measures how many words from the reference text / ground truth are appearing in the recommended text. It is just not a single metric but a set of metrics including: Precision, Recall, F1 Score, ROUGE-N, ROUGE-L and ROUGE-S. Here, Precision, Recall and F1 Score are the same as explained in section 6.2. We will explain the remaining three metrics.

ROUGE-N - It represents the number of matching N-grams between model generated text and reference text. It is represented using the equation 6.16.

$$\text{ROUGE-N} = \frac{\sum_{n-grams\ in\ reference} min\left(Count\left(reference\ (n-gram)\right), Count\left(generated\ (n-gram)\right)\right)}{\sum_{n-grams\ in\ reference} Count\left(reference\ n-gram\right)} \quad (6.16)$$

where,

Count (reference (n-gram)) → Number of times an n-gram appears in reference text.

Count (generated (n-gram)) → Number of times an n-gram appears in generated text.

Min → Takes the minimum count of an n-gram between the reference and generated, ensuring only matched parts are considered.

ROUGE-L - It represents the Longest Common Subsequence between model generated text and reference text i.e. longest sequence of words matching between the two. It is represented using the equation 6.17.

$$\text{ROUGE-L} = \frac{LCS\ (Reference\ text, Generated\ text)}{Length\ of\ Reference\ text} \tag{6.17}$$

where,
LCS → Length of Longest Common Subsequence between model generated text and reference text.

ROUGE-S - Here, S stands for Skip-gram. It provides a leniency over ROUGE-N which searches for the exact match in the recommended text. ROUGE-S searches for the sequence of words in the recommended text matching with the reference text but having other words in between them. It is represented using the equation 6.18.

$$\text{ROUGE-S} = \frac{\sum_{skip-bigram\ in\ reference} min\ (Count\ (reference\ (skip-bigram)), Count\ (generated\ (skip\ -bigram)))}{\sum_{skip-bigram\ in\ reference} Count\ (reference\ skip-bigram)} \tag{6.18}$$

where,
Skip-bigrams → pairs of words appearing in the same order but not necessarily consecutively.

6.5.3 BERTScore - BERTScore is defined as the cosine similarity between the BERT embeddings of recommended and ground truth items. It can be denoted by the equation 6.19.

$$BERTScore = \frac{\sum_{i=1}^{n} cosine\_sim(BERT\_embedding(ri), BERT\_embedding(gi))}{n} \tag{6.19}$$

Here, ri is the i-th recommended item and gi is the i-th ground truth item.

6.5.4 BARTScore - BARTScore is defined as the cosine similarity between the BART embeddings of recommended and ground truth items. It can be denoted by the equation 6.20.

$$BARTScore = \frac{\sum_{i=1}^{n} cosine\_sim(BART\_embedding(ri), BART\_embedding(gi))}{n} \quad (6.20)$$

Here, ri is the i-th recommended item and gi is the i-th ground truth item.

6.6 Proposed Evaluation Metric

After having a discussion of various evaluation metrics in the above sections, we now proceed towards proposing a novel metric based on linguistics.

6.6.1 Semantic Recommendation Score (SRS)

Sometimes, there are situations where both semantics along with linguistic fluency needs to be considered. For such systems, we propose a novel metric, named Semantic Recommendation Score (SRS), which is a hybrid of BERTScore and BARTScore. SRS can be defined as the combination of both BERTScore and BARTScore by assigning appropriate weights to each of them. It can be defined by the equation 6.21. Framework of the proposed metric is shown in fig.6.4. The description of the procedure of the calculation of the SRS value is given by Algorithm 6.1.

$$SRS = W_a * BERTScore + W_b * BARTScore \quad (6.21)$$

where, W_a and W_b are the appropriate weights assigned to BERTScore and BARTScore respectively. BERTScore is calculated using equation 6.19 and BARTScore is calculated using equation 6.20.

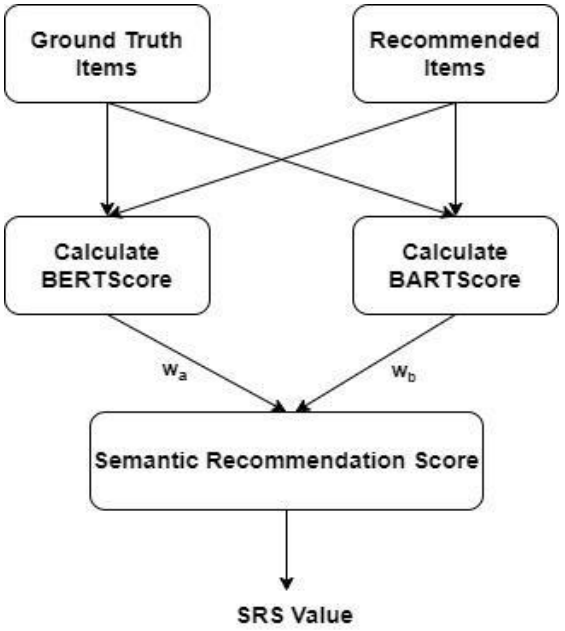

Fig.6.4: Flowchart of the calculation of SRS value

Algorithm 1: SRS_CALCULATION (G, R)

Input: Set of ground truth items (G) and recommended items (R)

Output: SRS_Value

1. Procedure SRS_CALCULATION (G, R)
2. for each gi and ri in G and R respectively do
3. gi_emb_bert ← BERT_embedding(gi)
4. ri_emb_bert ← BERT_embedding(ri)
5. Similarity_score_bert←cosine_similarity(gi_emb_bert, ri_emb_bert)
6. BERTScore ← average(similarity_score_bert)
7. gi_emb__bart ← BART_embedding(gi)
8. ri_emb_bart ← BART_embedding(ri)
9. similarity_score_bart←cosine_similarity (gi_emb_bart, ri_emb_bart)
10. BARTScore ← average (similarity_score_bart)
11. SRS_Value ← W_a * BERTScore + W_b * BARTScore
12. end
13. Return SRS_Value

6.6.2 SRS Analysis

In the following section, we describe the values of weights W_a and W_b to be assigned in different scenarios.

6.6.2.1 Emphasis on Semantics

In situations where semantics is relevant in comparison to linguistic quality then W_a is given a higher weightage to W_b. For example, in News Recommendation systems, where semantics is a crucial factor, W_a can be given the value of 0.7 and W_b can be given the value of 0.3.

6.6.2.2 Emphasis on Linguistic Quality

In situations where linguistic quality is relevant in comparison to semantics then W_b is given a higher weightage to W_a. For example, in an application for Language Learning, recommendations are made for the interactive lessons, well explained narrations and engaging exercises. In such cases, linguistic quality is important, so W_a can be given the value of 0.3 and W_b can be given the value of 0.7.

6.6.2.3 Equal Emphasis on both Semantics and Linguistic Quality

In situations like Movie Recommendations where both linguistic quality (movie descriptions, reviews) and semantics (genre, plot) are important, then both W_a and W_b can be given the value of 0.5 each.

6.7 Conclusion

In this chapter, we have explained various accuracy measurement metrics broadly categorized in four divisions - confusion matrix, error based, ranking based and language translations. We have seen that when precision and recall are equal, then F1 score is also equal. Also, precision does not take rank into account, so for this, MAP is used for top-k recommendations. Hit Rate, Hit Ratio can also be used in case of ranking. When the rating database is sparse, and the model has to predict the ratings of the items to fill the database, then error based metrics like MAE, MSE, and RMSE

are used. For recommendation based on language translations and summarization, BLEU, ROUGE, BERTScore and BARTScore are used. ROUGE is a special form of BLEU considering recall as well. BLEU, ROUGE and BARTScore don't take semantics (meaning) of the text into consideration i.e. they give different scores to the sentences having the same meaning but different words. This drawback is taken care of by BERTScore which has bright future scope in the recommendation systems.

We also propose a novel evaluation metric, Semantic Recommendation Score, which is a hybrid of BERTScore and BARTScore. As BERTScore considers the semantics along with the syntactic structure of the generated text and BARTScore mainly focuses on the linguistic quality and the coherence between the generated and reference text, sometimes there are situations where both the semantics and linguistic quality needs to be considered. So, the proposed metric, SRS, helps to consider both the semantics and linguistic fluency of the generated/recommended text. This metric also has a limitation as it involves the computation of BERT and BART models. These heavy models lead to resource intensive computational costs for large datasets. SRS involves the computation of weights W_a and W_b manually, we plan to use some optimization algorithm to compute these weights as the future work.

CHAPTER 7

CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

7.1 Conclusion

Sampling and noise filtering are the important steps before making the data suitable for processing. Thus, a comprehensive survey of both sampling and noise filtering methods is discussed in chapter 3. We have discussed six types of sampling methods used for recommender systems, namely, Bayesian Hierarchical Sampling, Negative Sampling, Thompson Sampling, Bernoulli Sampling, Gibbs Sampling and Bootstrap Sampling. Negative sampling works best for user-item recommendations that involve ratings of the items. Thompson sampling has been found to be effective in enhancing the performance of conversational and interactive recommender systems. For recommending movies and products, Gibbs sampling and Bernoulli distribution work best.

Next, in chapter 3, we describe various types of noise - malicious, natural, structural and contextual and various methods / models to eradicate these noise from RS. Because of the open nature of recommender systems, malicious noise can find its way in. As a result, we have exposed some research articles that focus on techniques for identifying this kind of noise in the database. Additionally, customers are occasionally reluctant to provide accurate reviews or ratings for the products they have bought. Consequently, this introduces false ratings, which adds to the system's natural noise. Prior to processing the data and making additional recommendations, natural noise must also be eliminated. As a result, we also offer a review of studies that address natural noise in recommender systems.

Structural and contextual noise are some of the other types of noise that are present in addition to malicious and natural noise. The recommendation systems are equipped with sophisticated methods, like optimization algorithms, to deal with these kinds of noise. The quality of the recommendation systems is enhanced by optimization. Its main goal is to improve RS performance by reducing the effect of noise by using

computational and mathematical techniques. As a result, we give a summary of a few studies that employ optimization strategies to reduce or eliminate the influence of noise while gauging recommender system performance.

After the critical analysis of noise filtering methods, in chapter 4, we proposed a model for movie recommendation in which tweets are trained with negative examples along with positive examples. We also introduced a novel optimization technique called Nuclear Physics Optimization (NPO). It is applied to filter out the irrelevant tweets. Through the proposed optimization-based noise filtering approach, we aimed to enhance the quality and relevance of movie recommendations generated from user-centric tweets, thereby enriching the overall movie recommendation experience. Our findings demonstrate that the optimization-based noise filtering method significantly improves the performance of movie recommendation systems.

Further, in chapter 5, we also proposed a model named BELHASH, which stands for ‘Bert Embedding based LSTM for Hashtag recommendation’. This model is assessed using 100K COVID-19 tweets that were scraped via the Twitter API. Embeddings like Word2Vec, GLOVE do not take into account the context or semantics of the words, which results in the same embeddings for words that appear in different contexts or at different times. Conversely, Bert Embedding which is used in this work, captures the context of the same word in different perspectives and thus generates different embeddings. BERT combined with LSTM works effectively and generates good results in recommending hashtags. The outcomes demonstrate that BELHASH outperforms other cutting-edge models.

We also made an analysis of various performance evaluation parameters in chapter 6. We categorize accuracy measurement metrics broadly into four divisions - confusion matrix, error based, ranking based and linguistic based. As the technology is advancing and many language processing systems are arriving in the market, we focussed that there is a scope to apply lingual based metrics such as BLEU, ROUGE, BARTScore and BERTScore for recommendations based on language translations and summarization. Since BARTScore primarily focuses on

linguistic quality and coherence between the generated and reference text, and BERTScore takes into account both the syntactic structure and semantics of the generated text, there are circumstances in which both linguistic quality and semantics must be taken into account. Thus, we propose a novel evaluation metric, Semantic Recommendation Score, which is a hybrid of BERTScore and BARTScore. The suggested metric, SRS, aids in taking into account both the linguistic fluency and semantics of the generated/recommended text.

7.2 Future Work

We found that numerous noise filtration studies, as discussed in chapter 3, focus solely on numerical ratings, but in the future, diverse user feedback may be taken into account. Another area where noise handling is not well addressed is context-aware recommendation scenarios. Therefore, adding more features for noise detection and correction can be achieved by working on the context information, such as user comments. Furthermore, a lot of research was conducted using static datasets, but dynamic data sets are required for practical applications. Thus, models and processing techniques can be created in the future and applied to work with dynamic data streams.

Future research directions for the proposed model in chapter 4 of Movie recommendation, include refining the optimization-based noise filtering method for diverse data sources, as well as incorporating additional features and contextual information from user-centric tweets to further enrich the recommendation process. The model proposed in chapter 5 for Hashtag recommendation works only for Covid based tweets. We plan to extend this work capturing the trending and dynamic nature of tweets and thus recommending the hashtags according to the on-going trends. Also, we plan to extend our work for recommending the hashtags for multimodal data including images and videos. We also plan to resolve the cold start and sparsity problems in the proposed models for movies and hashtags. We also plan to extend the proposed models to handle cross-domain recommendation scenarios, where users' preferences and item characteristics may vary across different domains (e.g., movies, music, books) by investigating transfer learning techniques.

7.3 Impact on Society

Recommendation systems provide personalized content making someone relaxed and feel healthy. This automatic suggestion of personalized content saves a lot of time as it reduces labor to search anything manually. It also increases user satisfaction and engagement with the platform. Various movie recommendation platforms like Netflix, Amazon Prime lead to increased subscriptions and advertising revenues. These platforms generate original content, based on audience preferences. Data-driven insights from user interactions encourage innovation and creativity in the entertainment sector. Consequently, movie recommendation systems are not only improving user experiences but also influencing how media will be consumed in the future, promoting economic expansion, and opening up new business opportunities for content creators across the globe.

Recommendation systems on social media sites increase social connectivity by connecting similar users and thus boosting social relations. It also leads to better spread of information within and across societies. These systems produce communities centered around common interests and passions by recommending friends, groups, and content that match users' preferences. Users are able to grow their social networks in unthinkable ways due to this improved connectivity, which not only makes new connections easier but also reinforces the ones that already exist.

REFERENCES

[1] Abbas, M., Riaz, M. U., Rauf, A., Khan, M. T., & Khalid, S. (2017, December). Context-aware Youtube recommender system. In *2017 IEEE international conference on information and communication technologies (ICICT),* (pp. 161-164).

[2] Gürmeriç, C. (2019). *Behavioral changes of the audience by the algorithmic recommendation systems inside video-on-demand platforms considering the example of Netflix* (Master's thesis, Bilkent Universitesi (Turkey)).

[3] Hannon, J., Bennett, M., & Smyth, B. (2010, September). Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems,* (pp. 199-206).

[4] Dwivedi, R., Anand, A., Johri, P., Banerji, A., & Gaur, N. (2020). Product based recommendation system on amazon data. *Int J Creat Res Thoughts–IJCRT,* (pp. 1-8).

[5] Liang, Z., Zhang, G., Huang, J. X., & Hu, Q. V. (2014, November). Deep learning for healthcare decision making with EMRs. In *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM),* (pp. 556-559).

[6] Wu, Q., Liu, Y., Li, Q., Jin, S., & Li, F. (2017, October). The application of deep learning in computer vision. In *2017 IEEE Chinese Automation Congress (CAC),* (pp. 6522-6527).

[7] Strubell, E., Ganesh, A., & McCallum, A. (2020, April). Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34, No. 09, (pp. 13693-13696).

[8] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of big data*, *2*, (pp. 1-21).

[9] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, *147*, (pp. 70-90).

[10] Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, *187*, (pp. 27-48).

[11] Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*. Berlin, Heidelberg: Springer Berlin Heidelberg, (pp. 325-341).

[12] Sen, S., Vig, J., & Riedl, J. (2009, April). Tagommenders: connecting users to items through tags. In *Proceedings of the 18th international conference on World wide web,* (pp. 671-680).

[13] Shi, B., Ifrim, G., & Hurley, N. (2016, April). Learning-to-rank for real-time high-precision hashtag recommendation for streaming news. *In Proceedings of the 25th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, (pp. 1191-1202).

[14] Shi, B., Poghosyan, G., Ifrim, G., & Hurley, N. (2017). Hashtagger+: Efficient high-coverage social tagging of streaming news. *IEEE Transactions on Knowledge and Data Engineering*, 30(1), (pp. 43-58).

[15] Li, J., & Xu, H. (2016). Suggest what to tag: Recommending more precise hashtags based on users' dynamic interests and streaming tweet content. *Knowledge-Based Systems*, 106, (pp. 196-205).

[16] Mooney, R. J., & Roy, L. (2000, June). Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries,* (pp. 195-204).

[17] Ma, H., King, I., & Lyu, M. R. (2009, July). Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval,* (pp. 203-210).

[18] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web,* (pp. 285-295).

[19] Hu, Y., Koren, Y., & Volinsky, C. (2008, December). Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining,* (pp. 263-272).

[20] Popescul, A., Ungar, L. H., Pennock, D. M., & Lawrence, S. (2013). Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. *arXiv preprint arXiv:1301.2303,* (pp. 437-444).

[21] Wang, H., Wang, N., & Yeung, D. Y. (2015, August). Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining,* (pp. 1235-1244).

[22] Li, S., Kawale, J., & Fu, Y. (2015, October). Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management,* (pp. 811-820).

[23] Covington, P., Adams, J., & Sargin, E. (2016, September). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems,* (pp. 191-198).

[24] Yang, C., Bai, L., Zhang, C., Yuan, Q., & Han, J. (2017, August). Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* (pp. 1245-1254).

[25] Wu, H., Zhang, Z., Yue, K., Zhang, B., He, J., & Sun, L. (2018). Dual-regularized matrix factorization with deep neural networks for recommender systems. *Knowledge-Based Systems*, *145*, (pp. 46-58).

[26] Shen, X., Yi, B., Zhang, Z., Shu, J., & Liu, H. (2016, July). Automatic recommendation technology for learning resources with convolutional neural network. In *2016 IEEE International Symposium on Educational Technology (ISET),* (pp. 30-34).

[27] Gong, Y., & Zhang, Q. (2016, July). Hashtag Recommendation Using Attention-Based Convolutional Neural Network. In *IJCAI,* (pp. 2782-2788).

[28] Wang, J., Sun, J., Lin, H., Dong, H., & Zhang, S. (2017). Convolutional neural networks for expert recommendation in community question answering. *Science China Information Sciences*, *60*(110102), (pp. 1-9).

[29] Zheng, L., Noroozi, V., & Yu, P. S. (2017, February). Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining,* (pp. 425-434).

[30] Barbieri, J., Alvim, L. G., Braida, F., & Zimbrão, G. (2017). Autoencoders and recommender systems: COFILS approach. *Expert Systems with Applications*, *89*, (pp. 81-90).

[31] Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015, May). Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th ACM International Conference on World Wide Web,* (pp. 111-112).

[32] Deng, S., Huang, L., Xu, G., Wu, X., & Wu, Z. (2016). On deep learning for trust-aware recommendations in social networks. *IEEE transactions on neural networks and learning systems*, *28*(5), (pp. 1164-1177).

[33] Zuo, Y., Zeng, J., Gong, M., & Jiao, L. (2016). Tag-aware recommender systems based on deep neural networks. *Neurocomputing*, *204*, (pp. 51-60).

[34] Wang, H., Shi, X., & Yeung, D. Y. (2015, February). Relational stacked denoising autoencoder for tag recommendation. In *Twenty-ninth AAAI conference on artificial intelligence,* (pp. 3052-3058).

[35] Gao, J., Zhang, T., & Xu, C. (2017, October). A unified personalized video recommendation via dynamic recurrent neural networks. In *Proceedings of the 25th ACM international conference on Multimedia,* (pp. 127-135).

[36] Jannach, D., & Ludewig, M. (2017, August). When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems,* (pp. 306-310).

[37] Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939,* (pp. 1-10).

[38] Li, J., Xu, H., He, X., Deng, J., & Sun, X. (2016, July). Tweet modeling with LSTM recurrent neural networks for hashtag recommendation. In *2016 IEEE International Joint Conference on Neural Networks (IJCNN)*, (pp. 1570-1577).

[39] Cui, Q., Wu, S., Liu, Q., Zhong, W., & Wang, L. (2018). MV-RNN: A multi-view recurrent neural network for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *32*(2), (pp. 317-331).

[40] Zhang, Q., Wang, J., Huang, H., Huang, X., & Gong, Y. (2017, August). Hashtag Recommendation for Multimodal Microblog Using Co-Attention Network. In *IJCAI,* (pp. 3420-3426).

[41] Zhao, Z., Sun, J., Yao, L., Wang, X., Chu, J., Liu, H., & Yu, G. (2017). Modeling Chinese microblogs with five Ws for topic hashtags extraction. *Tsinghua Science and Technology*, 22(2), (pp. 135-148).

[42] Ma, J., Feng, C., Shi, G., Shi, X., & Huang, H. (2018). Temporal enhanced sentence-level attention model for hashtag recommendation. *CAAI Transactions on Intelligence Technology,* 3(2), (pp. 95-100).

[43] Zhang, S., Yao, Y., Xu, F., Tong, H., Yan, X., & Lu, J. (2019, July). Hashtag recommendation for photo sharing services. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33, No. 01, (pp. 5805-5812).

[44] Maity, S. K., Panigrahi, A., Ghosh, S., Banerjee, A., Goyal, P., & Mukherjee, A. (2019). DeepTagRec: A content-cum-user based tag recommendation framework for stack overflow. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part II 41,* Springer International Publishing, (pp. 125-131).

[45] Wei, Y., Cheng, Z., Yu, X., Zhao, Z., Zhu, L., & Nie, L. (2019, October). Personalized hashtag recommendation for micro-videos. In *Proceedings of the 27th ACM International Conference on Multimedia,* (pp. 1446-1454).

[46] Yang, C., Wang, X., & Jiang, B. (2020). Sentiment Enhanced Multi-Modal Hashtag Recommendation for Micro-Videos. *IEEE Access*, 8, (pp. 78252-78264).

[47] Liu, S., Xie, J., Zou, C., & Chen, Z. (2020, July). User Conditional Hashtag Recommendation for Micro-Videos. *In 2020 IEEE International Conference on Multimedia and Expo (ICME),* (pp. 1-6).

[48] Liu, Y., Yang, S., Xu, Y., Miao, C., Wu, M., & Zhang, J. (2021). Contextualized graph attention network for recommendation with item knowledge graph. *IEEE Transactions on knowledge and data engineering*, *35*(1), (pp. 181-195).

[49] Chen, L., Cao, J., Wang, Y., Liang, W., & Zhu, G. (2022). Multi-view graph attention network for travel recommendation. *Expert Systems with Applications*, *191*, 116234, (pp. 1-13).

[50] Dong, Q., Liu, B., Zhang, X., Qin, J., & Wang, B. (2023). Sequential POI Recommend Based on Personalized Federated Learning. *Neural Processing Letters*, *55*(6), (pp. 7351-7368).

[51] Liu, Z., Yang, L., Fan, Z., Peng, H., & Yu, P. S. (2022). Federated social recommendation with graph neural network. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *13*(4), (pp. 1-24).

[52] Kumar, A., Jain, D. K., Mallik, A., & Kumar, S. (2024). Modified node2vec and attention based fusion framework for next POI recommendation. *Information Fusion*, *101*, 101998, (pp. 1-13).

[53] O'Mahony, M. P., Hurley, N. J., & Silvestre, G. C. (2006, January). Detecting noise in recommender system databases. In *Proceedings of the 11th international conference on Intelligent user interfaces,* (pp. 109-115).

[54] S. Lavanya and S. Palaniswami, 2016. Hierarchical Sampling Techniques for Imbalanced Datasets. *Asian Journal of Information Technology, 15,* (pp. 2887-2896).

[55] Li, G., Zhu, T., Hua, J., Yuan, T., Niu, Z., Li, T., & Zhang, H. (2019). Asking images: Hybrid recommendation system for tourist spots by hierarchical sampling statistics and multimodal visual Bayesian personalized ranking. *IEEE Access*, *7*, (pp. 126539-126560).

[56] Li, G., Hua, J., Yuan, T., Wu, J., Jiang, Z., Zhang, H., & Li, T. (2019). Novel recommendation system for tourist spots based on hierarchical sampling statistics and SVD++. *Mathematical Problems in Engineering*, *2019*(1), 2072375, (pp. 1-15).

[57] Zhang, Y., & Koren, J. (2007, July). Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval,* (pp. 47-54).

[58] John, "*Overview Negative Sampling on Recommendation Systems*", MLearning.ai. https://medium.com/mlearning-ai/overview-negative-sampling-on-recommendation-systems-230a051c6cd7 [accessed Jul 11, 2021].

[59] Rafailidis, D. (2019, October). Bayesian deep learning with trust and distrust in recommendation systems. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI),* (pp. 18-25).

[60] Liu, Z., Ma, Y., Ouyang, Y., & Xiong, Z. (2021). Contrastive learning for recommender system. *arXiv preprint arXiv:2101.01317,* (pp. 1-10).

[61] Xie, R., Liu, Q., Wang, L., Liu, S., Zhang, B., & Lin, L. (2022, August). Contrastive cross-domain recommendation in matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining,* (pp. 4226-4236).

[62] Liao, W., Zhang, Q., Yuan, B., Zhang, G., & Lu, J. (2022). Heterogeneous multidomain recommender system through adversarial learning. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(11), (pp. 8965-8977).

[63] Yang, J., Yi, X., Zhiyuan Cheng, D., Hong, L., Li, Y., Xiaoming Wang, S., ... & Chi, E. H. (2020, April). Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion Proceedings of the Web Conference 2020,* (pp. 441-447).

[64] Zhou, K., Zhao, W. X., Wang, H., Wang, S., Zhang, F., Wang, Z., & Wen, J. R. (2020, October). Leveraging historical interaction data for improving conversational recommender system. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management,* (pp. 2349-2352).

[65] Huang, T., Zhang, D., & Bi, L. (2020). Neural embedding collaborative filtering for recommender systems. *Neural Computing and Applications*, *32*(22), (pp. 17043-17057).

[66] Katehakis, M. N., & Veinott Jr, A. F. (1987). The multi-armed bandit problem: decomposition and computation. *Mathematics of Operations Research*, *12*(2), (pp. 262-268).

[67] Kuleshov, V., & Precup, D. (2014). Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028,* (pp. 1-32).

[68] Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., & Wen, Z. (2018). A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, *11*(1), (pp. 1-96).

[69] Chapelle, O., & Li, L. (2011). An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, *24,* (pp. 1-9).

[70] Hariri, N., Mobasher, B., & Burke, R. (2014, October). Context adaptation in interactive recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems,* (pp. 41-48).

[71] Song, Y., Wang, L., Dang, H., Zhou, W., Guan, J., Zhao, X., ... & Shao, J. (2021, July). Underestimation Refinement: A General Enhancement Strategy for Exploration in Recommendation Systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval,* (pp. 1818-1822).

[72] Lei, W., He, X., de Rijke, M., & Chua, T. S. (2020, July). Conversational recommendation: Formulation, methods, and evaluation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval,* (pp. 2425-2428).

[73] Krauth, K., Dean, S., Zhao, A., Guo, W., Curmei, M., Recht, B., & Jordan, M. I. (2020). Do Offline Metrics Predict Online Performance in Recommender Systems?. *arXiv preprint arXiv:2011.07931,* (pp.1-21).

[74] Gauthier, C. S., Gaudel, R., & Fromont, E. (2020). Position-based multiple-play bandits with thompson sampling. *arXiv preprint arXiv:2009.13181,* (pp. 1-7).

[75] Vargas, S., Baltrunas, L., Karatzoglou, A., & Castells, P. (2014, October). Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender systems,* (pp. 209-216).

[76] Liu, H., Wen, J., Jing, L., & Yu, J. (2019, September). Deep generative ranking for personalized recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems,* (pp. 34-42).

[77] Schmit, S., & Riquelme, C. (2018, March). Human interaction with recommendation systems. In *International Conference on Artificial Intelligence and Statistics,* PMLR, (pp. 862-870).

[78] Ortega, F., Lara-Cabrera, R., González-Prieto, Á., & Bobadilla, J. (2021). Providing reliability in recommender systems through Bernoulli Matrix Factorization. *Information Sciences*, *553*, (pp. 110-128).

[79] Ginart, A. A., Naumov, M., Mudigere, D., Yang, J., & Zou, J. (2021, July). Mixed dimension embeddings with application to memory-efficient recommendation systems. In *2021 IEEE International Symposium on Information Theory (ISIT),* (pp. 2786-2791).

[80] Wu, Q., Wang, H., Hong, L., & Shi, Y. (2017, November). Returning is believing: Optimizing long-term user engagement in recommender systems. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management,* (pp. 1927-1936).

[81] Liu, J., Wu, C., & Liu, W. (2013). Bayesian probabilistic matrix factorization with social relations and item contents for recommendation. *Decision Support Systems*, *55*(3), (pp. 838-850).

[82] Chakrabarty, N., Rana, S., Chowdhury, S., & Maitra, R. (2019, December). RBM based joke recommendation system and joke reader segmentation. In *International Conference on Pattern Recognition and Machine Intelligence,* Springer, Cham, (pp. 229-239).

[83] Liu, C., Jin, T., Hoi, S. C., Zhao, P., & Sun, J. (2017). Collaborative topic regression for online recommender systems: an online and Bayesian approach. *Machine Learning*, *106*(5), (pp. 651-670).

[84] Narayan, S., & Sathiyamoorthy, E. (2019). A novel recommender system based on FFT with machine learning for predicting and identifying heart diseases. *Neural Computing and Applications*, *31*(1), (pp. 93-102).

[85] Hussein, A. S., Omar, W. M., Li, X., & Ati, M. (2012). Accurate and reliable recommender system for chronic disease diagnosis. *Global Health*, (pp. 113-118).

[86] Hussein, A. S., Omar, W. M., Li, X., & Ati, M. (2012, December). Efficient chronic disease diagnosis prediction and recommendation system. In *2012 IEEE-EMBS Conference on Biomedical Engineering and Sciences,* (pp. 209-214).

[87] Tang, L., Jiang, Y., Li, L., Zeng, C., & Li, T. (2015, August). Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval,* (pp. 323-332).

[88] Vinagre, J., Jorge, A. M., & Gama, J. (2018). Online bagging for recommender systems. *Expert Systems*, *35*(4), e12303, (pp. 1-13).

[89] Zhang, F., & Chen, H. (2016). An ensemble method for detecting shilling attacks based on ordered item sequences. *Security and Communication Networks*, *9*(7), (pp. 680-696).

[90] Burke, R., Mobasher, B., Williams, C., & Bhaumik, R. (2006, August). Classification features for attack detection in collaborative recommender systems. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining,* (pp. 542-547).

[91] Morid, M. A., Shajari, M., & Hashemi, A. R. (2014). Defending recommender systems by influence analysis. *Information Retrieval*, *17*(2), (pp. 137-152).

[92] Chirita, P. A., Nejdl, W., & Zamfir, C. (2005, November). Preventing shilling attacks in online recommender systems. In *Proceedings of the 7th annual ACM international workshop on Web information and data management,* (pp. 67-74).

[93] Zhou, Q. (2016). Supervised approach for detecting average over popular items attack in collaborative recommender systems. *IET Information Security*, *10*(3), (pp. 134-141).

[94] Kapoor, S., Gupta, V., & Kumar, R. (2018). An obfuscated attack detection approach for collaborative recommender systems. *Journal of computing and information technology*, *26*(1), (pp. 45-56).

[95] Williams, C. A., Mobasher, B., & Burke, R. (2007). Defending recommender systems: detection of profile injection attacks. *Service Oriented Computing and Applications*, *1*(3), (pp. 157-170).

[96] Yang, F., Gao, M., Yu, J., Song, Y., & Wang, X. (2018, November). Detection of shilling attack based on bayesian model and user embedding. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI),* (pp. 639-646).

[97] Zhang, F., & Zhou, Q. (2014). HHT–SVM: An online method for detecting profile injection attacks in collaborative recommender systems. *Knowledge-Based Systems*, *65*, (pp. 96-105).

[98] Zhou, W., Wen, J., Gao, M., Liu, L., Cai, H., & Wang, X. (2015, October). A shilling attack detection method based on SVM and target item analysis in collaborative filtering recommender systems. In *International Conference on Knowledge Science, Engineering and Management,* Springer, Cham, (pp. 751-763).

[99] Yang, Z., Xu, L., Cai, Z., & Xu, Z. (2016). Re-scale AdaBoost for attack detection in collaborative filtering recommender systems. *Knowledge-Based Systems*, *100*, (pp. 74-88).

[100] Cao, J., Wu, Z., Mao, B., & Zhang, Y. (2013). Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system. *World Wide Web*, *16*(5), (pp. 729-748).

[101] Zhang, L., Yuan, Y., Wu, Z., & Cao, J. (2017, August). Semi-SGD: Semi-supervised learning based spammer group detection in product reviews. In *2017 IEEE Fifth International Conference on Advanced Cloud and Big Data (CBD),* (pp. 368-373).

[102] Zhou, Q., & Duan, L. (2021). Semi-supervised recommendation attack detection based on Co-Forest. *Computers & Security*, *109*, 102390, (pp. 1-18).

[103] Li, M., & Zhou, Z. H. (2007). Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(6), (pp. 1088-1098).

[104] Alostad, J. M. (2019). Improving the shilling attack detection in recommender systems using an SVM gaussian mixture model. *Journal of Information & Knowledge Management*, *18*(01), 1950011, (pp. 1-18).

[105] Zhang, F., Zhang, Z., Zhang, P., & Wang, S. (2018). UD-HMM: An unsupervised method for shilling attack detection based on hidden Markov model and hierarchical clustering. *Knowledge-Based Systems*, *148*, (pp. 146-166).

[106] Yang, L., Huang, W., & Niu, X. (2017). Defending shilling attacks in recommender systems using soft co‐clustering. *IET Information Security*, *11*(6), (pp. 319-325).

[107] Cai, H., & Zhang, F. (2019). An unsupervised method for detecting shilling attacks in recommender systems by mining item relationship and identifying target items. *The Computer Journal*, *62*(4), (pp. 579-597).

[108] Davoudi, A., & Chatterjee, M. (2017, December). Detection of profile injection attacks in social recommender systems using outlier analysis. In *2017 IEEE International Conference on Big Data (Big Data),* (pp. 2714-2719).

[109] Yang, Z., Cai, Z., & Guan, X. (2016). Estimating user behavior toward detecting anomalous ratings in rating systems. *Knowledge-Based Systems*, *111*, (pp. 144-158).

[110] Yang, Z., Sun, Q., Zhang, Y., & Zhang, B. (2018). Uncovering anomalous rating behaviors for rating systems. *Neurocomputing*, *308*, (pp. 205-226).

[111] Chakraborty, P., & Karforma, S. (2015). Effectiveness of proximity-based outlier analysis in detecting profile-injection attacks in E-Commerce Recommender Systems. In *Information Systems Design and Intelligent Applications,* Springer, New Delhi, (pp. 255-263).

[112] Zhou, W., Koh, Y. S., Wen, J., Alam, S., & Dobbie, G. (2014, July). Detection of abnormal profiles on group attacks in recommender systems. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval,* (pp. 955-958).

[113] Panagiotakis, C., Papadakis, H., & Fragopoulou, P. (2018, September). Detection of hurriedly created abnormal profiles in recommender systems. In *2018 IEEE International Conference on Intelligent Systems (IS),* (pp. 499-506).

[114] Papadakis, H., Michalakis, N., Fragopoulou, P., Panagiotakis, C., & Malamos, A. (2017, September). Movie score: Personalized movie recommendation on mobile devices. In *Proceedings of the 21st Pan-Hellenic Conference on Informatics,* (pp. 1-6).

[115] Panagiotakis, C., Papadakis, H., & Fragopoulou, P. (2020). Unsupervised and supervised methods for the detection of hurriedly created profiles in recommender systems. *International Journal of Machine Learning and Cybernetics*, *11*(9), (pp. 2165-2179).

[116] Amatriain, X., Pujol, J. M., & Oliver, N. (2009, June). I like it... i like it not: Evaluating user ratings noise in recommender systems. In *International Conference on User Modeling, Adaptation, and Personalization,* Springer, Berlin, Heidelberg, (pp. 247-258).

[117] Amatriain, X., Pujol, J. M., Tintarev, N., & Oliver, N. (2009, October). Rate it again: increasing recommendation accuracy by user re-rating. In *Proceedings of the third ACM conference on Recommender systems,* (pp. 173-180).

[118] Pham, H. X., & Jung, J. J. (2013). Preference-based user rating correction process for interactive recommendation systems. Multimedia tools and applications, 65(1), (pp. 119-132).

[119] Panagiotakis, C., Papadakis, H., Papagrigoriou, A., & Fragopoulou, P. (2021). Improving recommender systems via a dual training error based correction approach. *Expert Systems with Applications*, *183*, 115386, (pp. 1-17).

[120] Dixit, V. S., Jain, P., & Gupta, S. (2019). Proposed rcfs-cars framework with noise detection and correction. *Applied Artificial Intelligence*, *33*(4), (pp. 361-377).

[121] Li, D., Chen, C., Gong, Z., Lu, T., Chu, S. M., & Gu, N. (2019, May). Collaborative filtering with noisy ratings. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, Society for Industrial and Applied Mathematics, (pp. 747-755).

[122] Castro, J., Yera, R., & Martínez, L. (2017). An empirical study of natural noise management in group recommendation systems. *Decision Support Systems*, *94*, (pp. 1-11).

[123] Latha, R., & Nadarajan, R. (2015, November). Ranking based approach for noise handling in recommender systems. In *International Conference on Multimedia Communications, Services and Security*, Springer, Cham, (pp. 46-58).

[124] Pham, X. H., Jung, J. J., & Nguyen, N. T. (2012, November). Integrating multiple experts for correction process in interactive recommendation systems. In *International Conference on Computational Collective Intelligence*, Springer, Berlin, Heidelberg, (pp. 31-40).

[125] Yu, P., Lin, L., & Yao, Y. (2016, June). A novel framework to process the quantity and quality of user behavior data in recommender systems. In *International Conference on Web-Age Information Management*, Springer, Cham, (pp. 231-243).

[126] Bag, S., Kumar, S., Awasthi, A., & Tiwari, M. K. (2019). A noise correction-based approach to support a recommender system in a highly sparse rating environment. *Decision Support Systems*, *118*, (pp. 46-57).

[127] Baatarjav, E. A., Phithakkitnukoon, S., & Dantu, R. (2008, November). Group recommendation system for facebook. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, Springer, Berlin, Heidelberg, (pp. 211-219).

[128] Said, A., Jain, B. J., Narr, S., & Plumbaum, T. (2012, July). Users and noise: The magic barrier of recommender systems. In *International conference on user modeling, adaptation, and personalization*, Springer, Berlin, Heidelberg, (pp. 237-248).

[129] Said, A., & Bellogín, A. (2018). Coherence and inconsistencies in rating behavior: estimating the magic barrier of recommender systems. *User Modeling and User-Adapted Interaction*, *28*(2), (pp. 97-125).

[130] Saia, R., Boratto, L., & Carta, S. (2016). A semantic approach to remove incoherent items from a user profile and improve the accuracy of a recommender system. *Journal of Intelligent Information Systems*, *47*(1), (pp. 111-134).

[131] Li, B., Chen, L., Zhu, X., & Zhang, C. (2013). Noisy but non-malicious user detection in social recommender systems. *World Wide Web*, *16*(5), (pp. 677-699).

[132] Toledo, R. Y., Mota, Y. C., & Martínez, L. (2015). Correcting noisy ratings in collaborative recommender systems. *Knowledge-Based Systems*, *76*, (pp. 96-108).

[133] Xu, Y. Y., Gu, S. M., & Min, F. (2022). Improving recommendation quality through outlier removal. *International Journal of Machine Learning and Cybernetics*, *13*(7), (pp. 1819-1832).

[134] Toledo, R. Y., López, L. M., & Mota, Y. C. (2013, June). Managing natural noise in collaborative recommender systems. In *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS),* IEEE, (pp. 872-877).

[135] Yera, R., Castro, J., & Martínez, L. (2016). A fuzzy model for managing natural noise in recommender systems. *Applied Soft Computing*, *40*, (pp. 187-198).

[136] Yera, R., Barranco, M. J., Alzahrani, A. A., & Martínez-López, L. (2019). Exploring Fuzzy Rating Regularities for Managing Natural Noise in Collaborative Recommendation. *Int. J. Comput. Intell. Syst.*, *12*(2), (pp. 1382-1392).

[137] Sharon Moses, J., & Dhinesh Babu, L. D. (2018). A fuzzy linguistic approach-based non-malicious noise detection algorithm for recommendation system. *International Journal of Fuzzy Systems*, *20*(8), (pp. 2368-2382).

[138] Wang, P., Wang, Y., Zhang, L. Y., & Zhu, H. (2021). An effective and efficient fuzzy approach for managing natural noise in recommender systems. *Information Sciences*, *570*, (pp. 623-637).

[139] Castro, J., Yera, R., & Martinez, L. (2018). A fuzzy approach for natural noise management in group recommender systems. *Expert Systems with Applications*, *94*, (pp. 237-249).

[140] Kumar, S., Kumar, A., Mallik, A., & Singh, R. R. (2023). Optnet-fake: Fake news detection in socio-cyber platforms using grasshopper optimization and deep neural network. *IEEE Transactions on Computational Social Systems,* (pp. 1-10).

[141] Ala, A., Alsaadi, F. E., Ahmadi, M., & Mirjalili, S. (2021). Optimization of an appointment scheduling problem for healthcare systems based on the quality of fairness service using whale optimization algorithm and NSGA-II. *Scientific Reports*, *11*(1), 19816, (pp. 1-19).

[142] Ijiga, A. C., Peace, A. E., Idoko, I. P., Agbo, D. O., Harry, K. D., Ezebuka, C. I., & Ukatu, I. E. (2024). Ethical considerations in implementing generative AI for healthcare supply chain optimization: A cross-country analysis across India, the United Kingdom, and the United States of America. *International Journal of Biological and Pharmaceutical Sciences Archive*, *7*(01), (pp. 048-063).

[143] Kolasani, S. (2023). Optimizing natural language processing, large language models (LLMs) for efficient customer service, and hyper-personalization to enable sustainable growth and revenue. *Transactions on Latest Trends in Artificial Intelligence*, *4*(4), (pp. 1-31).

[144] Li, T., Kou, G., Peng, Y., & Philip, S. Y. (2021). An integrated cluster detection, optimization, and interpretation approach for financial data. *IEEE transactions on cybernetics*, *52*(12), (pp. 13848-13861).

[145] Affandi, A., Sarwani, A. S., Erlangga, H., Siagian, A. O., Purwanto, A., Effendy, A. A., & Juhaeri, G. (2020). Optimization of MSMEs empowerment in facing competition in the global market during the COVID-19 pandemic time. *Systematic Reviews in Pharmacy*, *11*(11), (pp. 1506-1515).

[146] Li, Z. (2024). Application And Optimization of Deep Reinforcement Learning in News Recommendation. *Highlights in Science, Engineering and Technology*, *85*, (pp. 389-395).

[147] Tortorella, G. L., Fogliatto, F. S., Mac Cawley Vergara, A., Vassolo, R., & Sawhney, R. (2020). Healthcare 4.0: trends, challenges and research directions. *Production Planning & Control*, *31*(15), (pp. 1245-1260).

[148] Kumar, A., Aggarwal, N., & Kumar, S. (2023). SIRA: a model for propagation and rumor control with epidemic spreading and immunization for healthcare 5.0. *Soft Computing*, *27*(7), (pp. 4307-4320).

[149] Krestel, R., Fankhauser, P., & Nejdl, W. (2009, October). Latent dirichlet allocation for tag recommendation. In *Proceedings of the third ACM conference on Recommender systems,* (pp. 61-68).

[150] Chen, H. M., Chang, M. H., Chang, P. C., Tien, M. C., Hsu, W. H., & Wu, J. L. (2008, October). Sheepdog: group and tag recommendation for flickr photos by automatic search-based learning. In *Proceedings of the 16th ACM international conference on Multimedia,* (pp. 737-740).

[151] Kaviani, M., & Rahmani, H. (2020, April). Emhash: Hashtag recommendation using neural network based on bert embedding. In *2020 IEEE 6th International Conference on Web Research (ICWR),* (pp. 113-118).

[152] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics,* (pp. 311-318).

[153] Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out,* (pp. 74-81).

[154] Yuan, W., Neubig, G., & Liu, P. (2021). Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34, (pp. 27263-27277).

[155] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). BERTScore: Evaluating Text Generation with BERT, *International Conference on Learning Representations*, (pp. 1-43).

[156] Amatriain, X., Jaimes*, A., Oliver, N., & Pujol, J. M. (2010). Data mining methods for recommender systems. In *Recommender systems handbook*. Boston, MA: Springer US, (pp. 39-71).

[157] Adomavicius, G., & Tuzhilin, A. (2010). Context-aware recommender systems. In *Recommender systems handbook*. Boston, MA: Springer US, (pp. 217-253).

[158] Ma, H., Zhou, D., Liu, C., Lyu, M. R., & King, I. (2011). Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining,* (pp. 287-296).

[159] Björklund, G., Bohlin, M., Olander, E., Jansson, J., Walter, C. E., & Au-Yong-Oliveira, M. (2022, April). An Exploratory Study on the Spotify Recommender System. In *World Conference on Information Systems and Technologies,* (pp. 366-378).

[160] Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1), (pp. 1-38).

[161] Liu, J., Wang, X., Tan, Y., Huang, L., & Wang, Y. (2022). An attention-based multi-representational fusion method for social-media-based text classification. *Information*, *13*(4), (pp. 171-185).

[162] Mikolov, T. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, *3781*.

[163] Han, J., Kamber, M., & Pei, J. (2012). Data Mining: Concepts and. Techniques, *Waltham: Morgan Kaufmann Publishers*.

[164] Siegbahn, K. (Ed.). (2012). Alpha-, beta-and gamma-ray spectroscopy. *Elsevier*.

[165] Hastie, T. (2009). The elements of statistical learning: data mining, inference, and prediction.

[166] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), (pp. 157-166).

[167] Onet, E. V. (2009). Particle Swarm Optimization and Genetic Algorithms. *Journal of Computer Science & Control Systems*, *2*(2).

[168] Ricci, F., Rokach, L., & Shapira, B. (2021). Recommender systems: Techniques, applications, and challenges. *Recommender systems handbook*, (pp. 1-35).

[169] Alsini, A., Huynh, D. Q., & Datta, A. (2021). Hashtag recommendation methods for twitter and sina weibo: a review. *Future Internet*, *13*(5), (pp. 129-147).

[170] Kenton, J. D. M. W. C., & Toutanova, L. K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT, 1*, (pp. 2-17).

[171] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. In *Neural Computation, 9*(8), (pp. 1735–1780).

[172] Ko, H., Lee, S., Park, Y., & Choi, A. (2022). A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, *11*(1), (pp. 141-188).

[173] Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. *Recommender systems handbook*, (pp. 257-297).

[174] Villegas, N. M., Sánchez, C., Díaz-Cely, J., & Tamura, G. (2018). Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems*, *140*, (pp. 173-200).

CURRICULUM VITAE / BRIEF PROFILE

Kirti Jain is a research scholar in the Computer Science and Engineering Department at Delhi Technological University. She is pursuing her Ph.D under the supervision of Prof. Rajni Jindal, Professor, ex-hod, Computer Science and Engineering Department, Delhi Technological University. Kirti received her M.Tech in Computer Science & Engineering from IIIT-Delhi and B.Tech in Information Technology from GGSIPU.

She is also working as an Assistant Professor at Computer Science & Engineering and Information Technology Department at Jaypee Institute of Information Technology, Noida, UP. She has prior experience in academics of around 7 years at Inderprastha Engineering College, Ghaziabad. She has also worked as a Software Engineer at Sopra India Pvt. Ltd. for 1 year and at Atlogys Technical Consulting for 8 months.

Her major areas of interest are Machine Learning and Deep Learning. She also has a keen interest in DBMS, Operating Systems, Compiler Design and Data Structures. She has qualified UGC-NET for Assistant Professor in the years 2014 and 2018. She has also qualified GATE in the past years. She has authored around 10 research papers for various national and international journals and conferences. She has also attended various Faculty Development Programs and Workshops of her interest. She has also done certification in various courses like “Programming for Everybody - Python”, “Introduction to Relational Database and SQL” and “Database Management Systems”.