**Implementation of a Neural Collaborative Filtering System**

**Executive Summary**

Every business is attempting to remain sustainable and grow in an ecosystem prone to competition prompting the question as to how would any given business gain a competitive edge. Technological advances brought about by innovation have proven to be useful to those who use them effectively and disastrous otherwise i.e., business that fail to evolve with current trends fall behind the competition. To this end, several tools ranging from customer management, decision support, marketing to prediction of future events have been put forward.

Some of these tools include recommender systems of which there are various categories such as content-based, knowledge-based, collaborative filtering, etcetera. The current study proposes an implementation of a neural collaborative filtering-based recommender systems which is evaluated against cosine similarity model and a benchmark matrix factorization model (NMF). Each of the three models were fitted and evaluated using three sets of data including *100K MovieLens, 10M MovieLens, and Learn from sets 2019* datasets as was evaluated using Mean Percentile Rank (MPR) and a Recall score. It was observed that on average, the NMF model had on average lower MPR compared to both cosine similarity and neural collaborative models respectively, for the three datasets i.e., *100K MovieLens* (40.487%, 42.323%, 49.279%), *10M MovieLens* (37.234%, 40.586%, 49.25%), and *learn from sets 2019 data* (46.404%, 47.469%, 46.404%) respectively. It was noted that the model-based recommender systems were negatively affected by sparsity compared to the model-based neural network model.

Table of Contents

[Chapter 1: Introduction 4](#_Toc80137703)

[1.1 Background 4](#_Toc80137704)

[1.1.1 Recommender Systems. What are they? 4](#_Toc80137705)

[1.1.1 Web-Based analytical tools 5](#_Toc80137706)

[1.2 Statement of Problem 6](#_Toc80137707)

[1.3 Purpose of the Study 7](#_Toc80137708)

[1.4 Significance of the Study 7](#_Toc80137709)

[1.5 Research Hypothesis 8](#_Toc80137710)

[1.6 Research Questions 8](#_Toc80137711)

[1.7 Disclaimer (tentative) 8](#_Toc80137712)

[1.8 Structure of the Report 9](#_Toc80137713)

[Chapter 2: Literature Review 11](#_Toc80137714)

[2.1 Introduction 11](#_Toc80137715)

[2.1.1 Big Data 11](#_Toc80137716)

[2.1.2 Recommender Systems 12](#_Toc80137717)

[2.2 Collaborative recommender system 13](#_Toc80137718)

[2.2.1 Classical Types Of Collaborative Filtering Algorithms 13](#_Toc80137719)

[2.2.2 Evaluation Metrics of Recommender Systems 19](#_Toc80137720)

[2.3 Other Aspects of Filtering systems 20](#_Toc80137721)

[2.3.1 Content-Based Filtering 20](#_Toc80137722)

[2.3.2 Data Sparsity 21](#_Toc80137723)

[2.4 Deep Learning 21](#_Toc80137724)

[2.5 Web-Based UI for Machine Learning Models 24](#_Toc80137725)

[Chapter 3: Research Methodology 24](#_Toc80137726)

[3.1 Introduction 24](#_Toc80137727)

[3.2 Data 25](#_Toc80137728)

[3.2.1 Data Collection 26](#_Toc80137729)

[3.3 Preliminaries 26](#_Toc80137730)

[3.3.1 Network Architecture 26](#_Toc80137731)

[3.3.2 Simple Cosine Similarity Collaborative Filtering 28](#_Toc80137732)

[3.3.3 Matrix Factorization 29](#_Toc80137733)

[3.4 Evaluation Metrics 29](#_Toc80137734)

[3.5 Experiments 30](#_Toc80137735)

[Research Limitations 33](#_Toc80137736)

[Chapter 4: Findings and Analysis 34](#_Toc80137737)

[4.1 Introduction 34](#_Toc80137738)

[4.1.1 Hypothesis 34](#_Toc80137739)

[4.2 Demographic Information 35](#_Toc80137740)

[4.3 Descriptive Statistics 37](#_Toc80137741)

[Recommender Experiments 39](#_Toc80137742)

[Chapter 4: Conclusion 43](#_Toc80137743)

[5.1 Performance of Neural Collaborative Filtering Compared to Memory-based models 43](#_Toc80137744)

[5.1.1 Matrix Factorization 43](#_Toc80137745)

[5.1.2 Model Performance 44](#_Toc80137746)

[5.2 Recommendations 45](#_Toc80137747)

[References 47](#_Toc80137748)

[Appendix 56](#_Toc80137749)

[Correlation statistics 56](#_Toc80137750)

[Neural Model Performance per Epoch 57](#_Toc80137751)

**List of Tables**

[Table 1: an example of a simple rating matrix 16](#_Toc80138167)

[Table 2: Common performance evaluation metrics 22](#_Toc80138168)

[Table 3: Model evaluation metrics 32](#_Toc80138169)

[Table 4: Dataset information 34](#_Toc80138170)

[Table 5: Summary statistics for 100K movies 40](#_Toc80138171)

[Table 6: Summary statistics for 10M movies 41](#_Toc80138172)

[Table 7: Test for trends in ratings (10M dataset) 41](#_Toc80138173)

[Table 8: Test for trends in ratings (100K dataset) 42](#_Toc80138174)

[Table 9: Cosine Similarity model performance vs non-negative Matrix factorization 43](#_Toc80138175)

[Table 10: Neural collaborative filtering model performance vs non-negative Matrix factorization 43](#_Toc80138176)

[Table 11:Cosine Similarity model performance vs non-negative Matrix factorization (10M dataset) 43](#_Toc80138177)

[Table 12: Neural collaborative filtering model performance vs non-negative Matrix factorization (10M dataset) 44](#_Toc80138178)

[Table 13: Cosine Similarity model performance vs non-negative Matrix factorization (learn from sets data) 44](#_Toc80138179)

[Table 14: Neural collaborative filtering model performance vs non-negative Matrix factorization (learn from sets data) 44](#_Toc80138180)

[Table 15 59](#_Toc80138181)

**List of Figures**

[Figure 1:Overview of the training phase of deep learning recommender systems, source: developer blogs 25](#_Toc80138226)

[Figure 2: Recommendation inference using deep learning, source: developer blogs 26](#_Toc80138227)

[Figure 3: Movie recommendation using deep learning 26](#_Toc80138228)

[Figure 4; Neural Collaborative filtering model architecture. 29](#_Toc80138229)

[Figure 5: Structure of the initial neural collaborative filtering model 35](#_Toc80138230)

[Figure 6: Distribution of the number of movies by genre (100K data) 38](#_Toc80138231)

[Figure 7: Distribution of the number of movies by genre (10M sampled data) 38](#_Toc80138232)

[Figure 8: Distribution of movie ratings (100K dataset) 39](#_Toc80138233)

[Figure 9: Distribution of movie ratings (10M dataset) 39](#_Toc80138234)

[Figure 10: Distribution of movie ratings (learn from sets data) 40](#_Toc80138235)

[Figure 11: Graphical comparison of how each model performs under various levels of data sparsity. 45](#_Toc80138236)

[Figure 12 60](#_Toc80138237)

[Figure 13 60](#_Toc80138238)

[Figure 14 60](#_Toc80138239)

**List of Equations**

[Equation 1 18](#_Toc80138277)

[Equation 2 19](#_Toc80138278)

[Equation 3 19](#_Toc80138279)

[Equation 4 19](#_Toc80138280)

[Equation 5 20](#_Toc80138281)

[Equation 6 20](#_Toc80138282)

[Equation 7 20](#_Toc80138283)

[Equation 8 20](#_Toc80138284)

[Equation 9 21](#_Toc80138285)

[Equation 10 22](#_Toc80138286)

[Equation 11 25](#_Toc80138287)

[Equation 12 32](#_Toc80138288)

[Equation 13 33](#_Toc80138289)

# Chapter 1: Introduction

## Background

After finishing your favorite movie on Netflix or Prime or watching that song on YouTube with nice visuals, one is often required to make an important decision regarding what to watch next? Which song to listen to after that lovely song you have been humming to on Spotify? What to buy next to compliment that shoe or even what combination of snacks to purchase. Often, in a sea of seemingly infinite choices to make, there is a need for a decision support system to enable an individual to make decisions regarding the next best option given their prior preferences. Whether a business is concerned with improving their customer experience or developing an online strategy, mobile strategy, or marketing, or any other customer-impacting part of an organization, there is a need for a decision support system [1].

### Recommender Systems. What are they?

At the very basic, the challenge to businesses operating in a highly competitive market relates to the provision of goods and services that are directly appealing to their customers [2]. The question however is which system should be adopted to inform the decision-making process. Of course, one could listen to all the songs or watch the trailers of each movie before settling for the best movie to watch. But that would not be the most efficient way, or would it? Alternatively, one could follow the advice of their friends regarding the best next cloth to purchase, next interesting movie or the best latest song by their favorite artist or the underlying businesses could adopt cutting-edge systems that predict what their customers would be interested in following their previous interactions i.e., recommender systems [3, 4]. Besides, according to [5], the need for recommender systems is driven by the continual growth of network technology as well as the ever-expanding scale of e-commerce, the available number of goods available to users that will often translate to spending a lot of time trying to find goods they want to buy.

According to [6], the extensive amount of resources that is available on the internet necessitates an efficient system that can be used for filtration, prioritization, as well as efficient delivery to aid in handling the problem of information overload a problem that [3] observe as having caused problems to internet users and can be solved by the integration of recommender systems in user applications but what exactly is a recommender system? [7] define a recommender system by examining its objective that is, “…recommender systems aim to predict users’ interests and recommend product items that quite likely are interesting for them.” [8] posit that recommender systems are essentially information filtration systems that help with the problem of information overload through applying filtering mechanisms to obtain vital information out of a possibly large amount of dynamically generated information based on a user’s preferences, interest, or observed interaction behavior about an underlying item” [9].

Ideally, items such as but not limited to movies, music, products, books, and articles can be identified as data or raw data to be precise in generic internet terms making the problem of what to buy, what to listen to, or watch solvable by recommender systems assuming the proposed recommender systems predict the accurate need of the user. Ideally, the process of making sense of customer-product interaction and being able to use the learned interactions for decision-making processes can then be considered as information generation.

### Web-Based analytical tools

One persistent problem in the field of data analytics is the problem of how to deploy the underlying systems [10]. Often, the deployment of analytical tools regardless of whether the tool is intended for descriptive or predictive analytics is reliant on factors such as the amount of data being used, how scalable the architecture of the deployment platforms is, latency time, among other factors [11]. Both [12] on the deployment of machine learning models as web applications and [13] on technologies on which to deploy machine learning models propose the use of web-based applications for the integration of machine learning models into production.

## Statement of Problem

Currently, there is limited research regarding the use of deep learning approaches in the development of recommender systems, and given the growing popularity of deep learning systems in fields such as computer vision, robotics, artificial intelligence, medical research, and medical diagnosis, the learning ability of deep learning models ought to be adopted for business practices including as decision making support tools, predictive analytics, and marketing tools with recommender systems as a subset of marketing methodologies given the effect of machine learning tools in effecting marketing campaigns and helping marketers to reach to quick decisions from a large amount of data big data available.

Most of the research conducted regarding the application of deep learning in the development of recommender systems dates back to 2016 an indication of how young the field of deep learning recommender systems is leaving room for contribution. Moreover, there are few studies if any that propose the use of deep learning collaborative filtering recommender system algorithms for movie recommendation using implicit information. Most of the existing recommender systems rely on explicit information where actions such as ratings by a customer are considered as the customer’s preference or lack of preference by the customer regarding a given product [14].

In practice as observed in the preceding sections, recommender systems rely on feedback i.e., explicit feedback which is considered the most convenient since it provides direct information regarding a user’s preference [15] but it is not always that customers provide such explicit feedback regarding products leaving businesses with the second feedback type, implicit information making the adoption of implicit-based recommender systems necessary in such cases. Few studies explore the adoption of both explicit-based and implicit-based recommender systems.

## Purpose of the Study

The purpose of the current study is to propose a deep learning-based recommender system for the recommendation of movies and deploy the optimal model on a web-based platform for use in production. The existence of implicit information regarding user interaction with movies makes it necessary for the development of systems that accept implicit information as input and make recommendations. In this study, both explicit and implicit-based deep learning models are proposed following the need to evaluate how well each of the models performs when making recommendations on movies to watch given a user’s ID. Moreover, the study proposes a non-negative Matrix Factorization implicit-based recommender system as a benchmark model for the implicit deep learning model. Also, the current work focus on deep learning in explicitly Collaborative Filtering-based approaches.

## Significance of the Study

As observed, the current study seeks to explore the landscape of deep learning models and their applications in developing recommender systems using both implicit and explicit information collected from user interactions. It is expected that by the completion of this study, the findings of this study will contribute towards the growing literature on deep learning-based recommender systems as alternatives to traditional recommender systems. Also, it is expected that the study will shed light on how different types of recommender systems perform especially from implicit and explicit perspectives as well as help develop an understanding of the integration of deep learning models on web-based platforms i.e., using a web-based user interface.

The study is particularly relevant following the growing interest in the possible applications of deep learning models to address business and engineering problems besides in “…video, image, text and audio recognition, autonomous driving, robotics, healthcare” [16] since it will further contribute and enhance the understanding of deep learning can be used for marketing purposes through recommender systems.

## Research Hypothesis

The study would seek to find evidence to support the variations in the performance of implicit-based and explicit-based deep learning recommender systems and use evidence to propose the validity of implicit-based deep learning recommender systems as recommender tools for movies.

## Research Questions

While exploring the application of explicit and implicit-based recommender systems, the study will attempt to find answers to the following questions:

1. How does the implicit-based simple cosine similarity model perform given various performance evaluation metrics?
2. How does the implicit-based non-negative matrix factorization model perform given various performance evaluation metrics?
3. How does the implicit-based deep learning model perform given various performance evaluation metrics?
4. How does the explicit-based deep learning model perform given various performance evaluation metrics?
5. Compared to the benchmark model (non-negative matrix factorization), how does the implicit-based model perform?
6. Does increasing the size of data improve the performance of the collaborative filtering models?
7. Does sparsity affect the performance of recommender systems?

## Disclaimer (tentative)

All the research conducted herein was accomplished following the required acceptance and authorization from the instructor. Generally, the information explored throughout this dissertation will be properly cited with the issues related to plagiarism and copyright infringement being addressed accordingly. Despite that much care has been taken in the research and preparation, business decisions should not be formulated based solely on the outcome of this report. In addition, the distribution of the research report is subject to the condition that it shall not, in any way of trade or otherwise, be resold, lent, or circulated on a commercial basis without the prior approval of both the instructor and researcher.

## Structure of the Report

Overall, the research is reported based on the following structure:

1. Chapter 1: Introduction

Provides a general introduction to the dissertation by giving context to what the dissertation is about with the help of Background which provides brief background information regarding the scope of the study. The background is split into general sub-sections including what recommender systems are about and what web-based deployment comprises, scope as well as the objectives and significance of the study.

1. Chapter 2: Literature review

The literature review chapter includes a comprehensive exploration of previous research studies that are related to recommender systems, their applications, deep learning approaches, and their applications in addressing business problems and particularly the application of deep learning in business from different perspectives including as a marketing tool, web-based machine learning deployment platforms, the use of web graphical user interfaces for managing deep learning models. This chapter will also introduce the integration of deep learning and big data as well as how deep learning is used for the development of recommender systems.

1. Chapter 3: Research Methodology

The third chapter provides an elaborate research methodology that is adopted by the research study in the endeavor of addressing the research objectives and formulating answers to the research questions. This includes an exploration of the functionality of the different recommender systems and the aspects of the research that the research methodology will cover.

1. Chapter 4: Findings and Analysis

This chapter involves the presentation of the analytical findings of this study following the adopted research methodology and the literature review. It will include a discussion of the findings in light of the literature review and research objective.

1. Chapter 5: Conclusion

The conclusions and recommendations are reported in this section and are drawn from a critical exploration of the findings presented in this study. They help outline the outcome of the developed recommender system as well as the implications of the current study.

# Chapter 2: Literature Review

## Introduction

### Big Data

For business firms to remain competitive while ensuring sustainability and growth in today’s market, they ought to adopt both market-oriented as well as customer-centric strategies which to a large extent apart from organizational management, include technological adoptions and appropriate decision-making strategies. [17] define a customer-centric approach as “… a strategy and a culture of doing business that focuses on creating the best experience for the customer, and by doing so builds brand loyalty.” The objective of improving customer experience can primarily be addressed through integrating suitable methodologies into business practices but first, it is imperative to understand customers but how? According to [18], customer experience (CX) which is conceptually a customer’s response to interactions with the business in question, is considered as a competitive differentiation strategy and with the underlying developments in big data analytics, there lie numerous potentials for generating insights useful for customer experience management (CXM).

In an article by [19], it is observed that big data is central in the development of personalized experiences that customers have come to expect mainly at scale, with minimal if any friction and with some element of human still intact. Today, big data in business practices can be found in smarter customer experience tools that are powered by AI, machine learning, as well as advanced data analytics that aid businesses in understanding their customers, what factors would lead to churn, and how to improve customer experience.

#### Applications of Big Data

Overall, big data has numerous applications including insight generation and predictive analytics or as [20] notes, it is useful for enabling “data scientists, predictive modelers, and other analytics professionals to analyze large volumes of transactional data.” Predictive models for analyzing big data include but are not limited to machine learning, deep learning, and data mining. Ideally, deep learning models facilitate the extraction of high-level abstractions as data representations through a hierarchical learning process with one prominent advantage of deep learning being that it can be used to extract insights from large amounts of unsupervised data as well as supervised data [21] this allows the application of implicit and explicit based analytical processes i.e., deep learning can be used to develop analytical models following implicit or explicit input.

### Recommender Systems

When you purchase a book from Amazon or watch a movie on YouTube, listen to a piece of music from Deezer or Spotify, the website proposes a list of items that you are likely to show interest in that is, they are designed to make recommendations. Recommender systems have become essential tools for a variety of applications such as online shopping, video rental, news, among other personalized websites The whole functionality of recommendation as demonstrated by such websites is based on the principle of online decision-making fuelled by recommender systems. [22] observes that the construction of systems that support users during their online decision-making processes in the selection of items of interest is the central purpose of the field of recommender systems. [22] further argues that recommender systems are helpful for customers besides also being essential for e-commerce activities and businesses.

In practice, there are up to six types of recommender systems that generally come in two varieties i.e., user and item-based [23]. The six types of recommender systems include:

1. Collaborative Recommender system
2. Content-based recommender system
3. Demographic-based recommender system
4. Utility-based recommender system
5. Knowledge-based recommender system
6. Hybrid recommender system

## Collaborative recommender system

In collaborative filtering, the recommender system is built to leverage user ratings that are, the model is built to learn whether an individual will prefer or dislike a movie based on the previous preferences of the user [23]. Regarding collaborative filtering (CF), [24] observe that it is a technology that focuses on learning on an individual’s past preference and making recommendations based on the learned behavior and pooled data. Ordinarily, a user is interpreted as an M-dimensional vector of items i.e. rating matrix in existed collaborative filtering system [24], and through conducting a comparison of the resulting vectors, the system then conducts an aggregation of the similar users as neighbors while eliminating users who have varying behavior and makes recommendations of items to the user in question-based on the behavior of similar users (neighbors) [24].

Based on the definition of CF by [23, 24, 25] in a classic CF problem, there is a list of *m* users {*u*1, *u*2, *u*3, …*u*m} as well as a list of *n* items {*i*1, *i*2, *i*3, …*i*n}, where each of the users, *ui* , has a list of items, *Iu*i , “…which the user has rated, or about which their preferences have been inferred through their behavior” [25]. Preferably, the ratings can either be explicit indications of following a specific scale such as 1-5 or implicit implications such as follow-through purchases, clicks, page views, or addition of the items to carts. The following table provides an example of a rating matrix.

Table 1: an example of a simple rating matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User/Item | Iris | Fast and Furious | Altered carbon | Angry Birds |
| Alice | 3 | 5 | 4 | 5 |
| Abigail | 5 | 4 | 5 | 5 |
| David | 4 | 5 | 5 | 3 |
| Antoine | 5 | 3 | 5 | 5 |

### Classical Types Of Collaborative Filtering Algorithms

There are two main classes of collaborative filtering algorithm systems that is, memory and model-based techniques [25, 26]. While in memory-based collaborative filtering (also identified as Neighborhood-based CF) the rating matrix is directly saved to the system for direct utilization to predict the ratings of the target items, model-based CF utilizes the items in the rating matrix to develop a model that is then exploited to identify the applicability of probable items for the target users [27].

#### Memory-based CF

Memory-based CF as observed earlier capitalizes on the underlying similarities between users or items for inferring the user’s possible preference in items that the user has not evaluated or interacted with previously [27]. Primarily, memory-based CF can be categorized into user-based CF where the system attempts to predict the ratings of a user on select items following ratings of similar users on the items in question [28]. The user-based CF conducts prediction of the user-item *u* to the item *i* is computed as shown in equation 1:

Equation 1



Where u is the average user rating of the user *u*, *sim* (*u, v*) is the similarity following a predetermined similarity matrix such as cosine similarity, Manhattan distance, Euclidean distance, Minkowski distance, and Jaccard similarity [29], of the users *u* and *v* while *Nu* denotes the group of users that are similar to user *u* i.e., the neighbors of user *u* who had rated item *i*.

On the other hand, the item-based CF depends on the similarities between the user items [27]. That is, it makes predictions of a user for an item based on the underlying user’s ratings of items that are similar to the item in question [25]. In the item-based technique, two items are considered similar if several users have evaluated these items similarly [30]. In this method, the rating of an individual is computed as shown in equation 2.

Equation 2

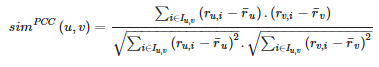


Where N*i* denotes a group of similar items to item j, *sim* (*u, v*) is the similarity score between items *j* and *k*. In practice, the computation of the similarity score is the main aspect of memory-based collaborative filtering systems since the similarity score will determine how accurate the system will be as well as the overall performance of the system [31]. From the mentioned similarity matrix measures, the standard similarity measures such as the cosine measure (COS) [32], the Pearson correlation coefficient (PCC) [34] as well as the Jaccard coefficient [33] are popularly used to find the most similar users or items.

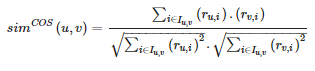
According to [27], the Pearson correlation coefficient determines the similarity of items or users depending on the linear correlation between two rating vectors of users or items. On the other hand, cosine measure computes the similarity of items or users by using the angle’s cosine value between rating vectors while the Jaccard similarity measures take into consideration the number of common ratings between users or items while ignoring the rating values.

The three standard metrics compute the similarity between two users as shown in equations 3, 4, and 5 below.

Equation 3



Equation 4



Equation 5

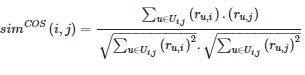


From the preceding three equations, *I*u,v represents the items that were rated by users u and v, u is the average user rating of the user *u* and *r*u, i denotes the ratings of user u for item i. Moreover, *I*u and *I*v denote the two sets of items that are rated by users *u* and *v* respectively. Similarly, the model-based CF system computes the similarity between two items say *i* and *j* by using the user’s ratings assigned two the two items using the formulae given in equations 6, 7, and 8.

Equation 6



Equation 7



Equation 8



From equations 6 to 8 above, *U*i,j denotes the group of users that evaluated items *i* and *j,* and i represents the average ratings for item i while *U*i and *U*j denote the sets of users who rated items *i* and *j* respectively.

Regardless, the main limitation of memory-based CF models is that they might incur prohibitive computational costs and by computation, it is in reference to the time used to compute the similarities between users and items alike. The computational costs have the tendency of augmenting as the number of users or items in the system grows [34].

#### Model-based CF

Compared to memory-based models which are simple to be implemented and effective for the inference of unknown ratings of users, model-based CF is comparatively complex to implement but tends to generate more precise predictions [26]. According to [27] follow the idea of utilizing data mining and machine learning modeling approaches for developing offline predictions. Based on the resulting models, the recommender systems predict missing ratings in the user-item matrix [35]. Model-based CF models have attracted a considerable amount of research in recent years with studies such as [36, 37] proposing Bayesian networks [36]and neural networks [38], support vector machines [39], fuzzy-based systems [37] and deep learning methods [38].

Regardless, Matrix factorization (MF) models are normally considered as state-of-the-art in recommendation problems given their optimal performance in terms of accuracy and scalability [40]. The functionality of MF is based on their use of high-level correlation among the respective rows and columns i.e., users and items respectively of the target user-item rating matrix during the process of learning the underlying patterns of users’ as well as item’s latent representations also identified as latent factors [41]. Specifically, every user *u* and each item *i* are represented by k-dimensional latent factors, i.e., qi ∈ Rk which denotes k-attributes of the item, and pu ∈ Rk that refer to the preference of a given user for the observed attributes. Conventionally, the rating score of any user *u* on a corresponding item *i* can be computed as shown below:

Equation 9



To allow the optimization of the generated latent factors for better estimation of u, i, the loss function ought to be minimized in such a way that:

Equation 10



Where T denotes the user-item (u, i) pairs for which real rating ru, i are observed in the training set and β is the defined regularization parameter that is adopted to handle any possibility of overfitting of the model. Inclusively, the minimization of the loss function as given in equation 6. In matrix factorization models such as the Non-Negative Matrix Factorization model, the number of latent factors might vary [42]. Generally, an increase in the number of chosen latent factors corresponds to an increase in the hidden factors that have been extracted from the user-item data and ultimately leads to an increase in the model’s quality of recommendations. However, to some extent, increasing the number of latent factors might lead to over-fitting over the data thus leading to a decrease in the model’s overall performance [42].

In comparison to the memory-based recommender systems, model-based systems tend to make more accurate predictions [27]. Besides, the fact that model-based systems consume fewer resources in terms of storage mainly due to the fact that in memory-based CF systems, all ratings need to be loaded onto the memory before generating recommendations, whereas, in model-based CF systems, predictions involve the learned model, which ideally is far smaller compared to the original rating matrix [43]. The most notable downside of the model-based recommender systems is the need for more training time as well as training data from which the models can learn about the patterns in addition to the need to retrain the model in the event that new users or items have been introduced to the system so as to maintain the model’s predictive accuracy.

## Evaluation Metrics of Recommender Systems

Whereas it is “easy” to generate recommendations, the adoption or rather acceptance of the recommended items largely depends on the performance of the corresponding recommender system making model evaluation an integral part of developing an efficient and deployable recommender system [44]. Several metrics have been put forward regarding the evaluation of how well the underlying collaborative filtering systems and can be categorized as either offline or online [45].

Online metrics which are considered the best evaluation metrics are considered so given their capacity to provide precise feedback of how pertinent the system is through real users [46], involve supplying recommendations to the users, and then inquire from them how they assess the recommended items while on the other hand, making offline model evaluation does not involve real user interaction rather, it uses historical user interactions which are introduced to the recommender system for training while part of it is held out for testing purposes.

Table 2 below provides an overview of the common metrics used to evaluate the performance of CF recommender systems.

Table 2: Common performance evaluation metrics

|  |  |  |  |
| --- | --- | --- | --- |
| Evaluation Metric | Summary | Formula | References |
| Root Mean Squared Error (RMSE) | Also defined as the standard error of the residuals, it focuses on the absolute errors resulting between the predictions and the real values. | Where:  *r*u, i denotes the real rating of a user *u* for a given item *i* and u, i, denotes the predicted rating by the CF recommender system  T={(*u, i*)} represents the set of user-item pairs whose actual user-ratings *r*u, i are provided. | [26] |
| Mean Absolute Error (MAE) | Computes the average of the absolute variation between the actual and predicted ratings |  | [47] |
| Precision | Evaluates the rate at which the recommendations made by the system are relevant. | Where:  *U*u is the number of items that were used by the user *u* while *L*rec is the list of items there were remembered. | [26] |
| Recall | Computes the rate at which the recommender makes positive recommendations |  | [26] |
| Ranking Score | Evaluates the quality of the recommendations of the recommender system based on their ranked position. | Where:  r(ij) denotes the rating of item *i* in the rank *j*  *md* denotes the median of the ratings  α denotes the value of half-life decay. | [48] |
| Mean Percentile Ranking | Computes the  percentile of an item *i* provided in the list of all the items, and then calculates the average percentile. Primarily, this implies that if one  made an allocation of recommendations at random, then the expected value of the MPR would be 50%. | Where:  r(i,j) denotes the rating of item *i* by user *u*  ui denotes the percentile rank of *i* in an ordered list such that ui = 0% implies that *i* is at the top of the list | [49, 50] |

## Other Aspects of Filtering systems

### Content-Based Filtering

Content-based filtering as opposed to collaborative filtering which copies user-to-user recommendations, content-based filtering generates recommendations based on the preference of the users to contents of products [51]. According to [51], “…it predicts users’ preferences as a linear, weighted combination of other user preferences.” That is, conceptually, content-based filtering utilizes the underlying similarities between features to make decisions [52] i.e., the functionality of content-based systems revolves around the process of making comparisons on user interest and product features an observation supported by [53] who observe that Content-based filtering approaches use a series of discrete attributes of an item so as to make recommendations regarding additional items that have similar properties to a user [55]. The common approach for content-based filtering systems is to observe item-based nearest neighbors, where u, i, is obtained from the ratings of user *u* for similar items [54].

Equation 11



Studies such as those [57, 58] propose the fusion of content-based filtering with collaborative filtering methods to develop hybrid recommender systems for usage in various fields including social network and semantic information.

### Data Sparsity

[55] observes that the problem of data sparsity arises from the phenomenon that users, in general, tend to rate only a limited number of items. Since the functionality of CF-based recommenders is to aggregate the ratings of like-minded users, the fact that most user-item matrices are very sparse with a sparsity of up to 99% as a result of lack of knowledge by the users or lack of incentives to rate the items meaning that there is little information from which to learn [60]. As such, CF models will suffer from highly sparse data with models such as matrix factorization being affected by highly sparse data compared to Cosine similarity since primarily, matrix factorization models are essentially linear models [56].

## Deep Learning

Like in many fields, deep learning is increasingly becoming popular in item recommendation systems [54]. [54] further observes that deep learning approaches are particularly important in the extraction of latent factors from items such as music from audio signals i.e., metadata besides being used to learn sequential patterns of items. Deep learning models generate recommendations by building on existing methods such as factorization to learn about the interactions among attributes and embeddings to handle categorical variables. An embedding is defined as a learned vector of numbers that entity features so that similar entities (users or items) have similar distances in the vector space [57]. According to [57], deep learning recommender systems can be split into two main phases including training and inference such that in the training phase, the model is prepared to make predictions on the probabilities about user-item interaction by using prior user-item interaction information. Figure 1 below provides an overview of the training phase of deep learning recommender systems in predicting whether a user interacted with an item or not.

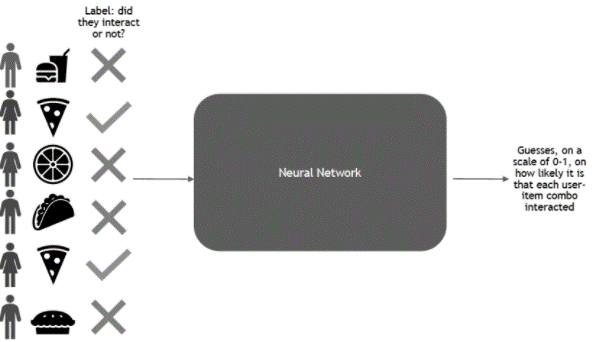


Figure 1:Overview of the training phase of deep learning recommender systems, source: [developer blogs](https://developer-blogs.nvidia.com/wp-content/uploads/2021/04/DL-recommendation_Pic3-625x330.png)

After learning about the underlying patterns, with sufficient accuracy, the resulting model can then be deployed as a service to make recommendations about new interactions as illustrated in figure 2 below.

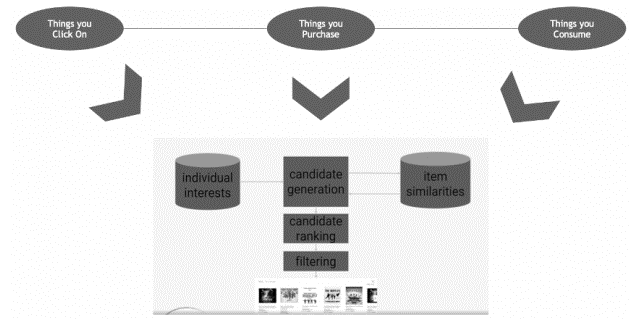


Figure 2: Recommendation inference using deep learning, source: [developer blogs](https://developer-blogs.nvidia.com/wp-content/uploads/2021/04/DL-recommendation_Pic3-625x330.png)

Primarily, for collaborative filtering for movie recommendations, the respective deep learning model learns about the user and item embeddings i.e., latent features based on the interactions between the user and the item using a neural network [57]. The following plot demonstrates how a deep learning model makes predictions for three different users using collaborative filtering.

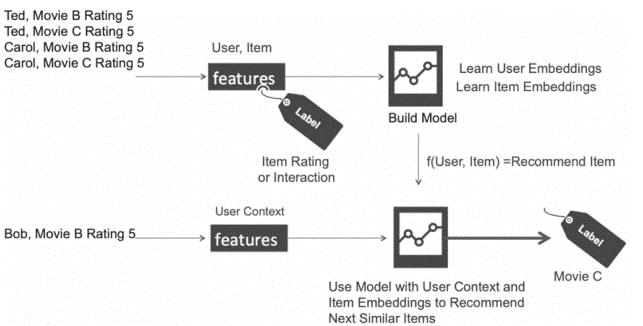


Figure 3: Movie recommendation using deep learning

## Web-Based UI for Machine Learning Models

After developing machine learning models of which deep learning is considered a subset, the subsequent problem lies in how to deploy the models. Often, models are built on standard environments such as in a Jupyter notebook but sharing analyses in such a form is not the most convenient way and as observed earlier, one can consider adopting building web apps for the resulting models [58].

Another popular deployment option is the cloud computing platform some of which include Amazon Web Services, Google’s Vertex AI, Microsoft Azure, Edge AI, etcetera [59]. The popularity of cloud-based computing deployment options generally stems from the observation that it is comparatively faster and easier to design applications through the cloud platform, especially deep learning models [59], but tends to have underlying issues related to the cost of latency during data transfer as well as security issues because cloud computing systems are prone to network attacks.

# Chapter 3: Research Methodology

## Introduction

The current work’s main objective is to propose an implicit and explicit-based movie recommender system using a collaborative filtering deep learning algorithm whose performance is compared with collaborative filtering simple cosine and non-negative matrix factorization. In this chapter, the data, models, performance evaluation, among other aspects related to the proposed research problem are provided.

According to [60], inherently, research in machine learning is empirical since the performance of machine learning (deep learning in this study’s case) algorithms is determined by how well their underlying assumptions align with the structure of the world and because no amount of mathematical analysis can be sufficient to determine whether or not any given machine learning algorithm will perform relatively well. Experimental studies are required. As such, fundamentally, to address the proposed research objectives and answer the research questions the current study will adopt an experimental approach by experimenting with different models and evaluating their performance under various environments (i.e., how the model performs when trained using varying parameters and settings).

## Data

As observed earlier, the objective is to develop a movie recommender system using secondary data sources i.e., historical user-movie interactions for training and a completely new dataset for validation and testing of the model’s performance. To this end, three datasets were collected. The first dataset is the *MovieLens 10M Dataset* which includes 10 million ratings alongside one million tag applications that are applied to 10,000 movies made by approximately 72,000 users. Besides, the data includes tag genome data with about 15 million relevance scores across 100,000 tag applications. The second data is MovieLens 100K Dataset*s* which includes 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users whose last update was in 2018. MovieLens 100K Dataset*s* changes over time and hence are not appropriate for reporting research results. The third dataset is the “Learning from Sets of Items 2019” dataset [61] which is processed to include ratings from 854 users over 29,516 sets containing 12,549 movies. In this study, *small MovieLens Latest Datasets* are used to evaluate the performance of the model that is trained using the *MovieLens 10M Dataset.* Both *MovieLens 100K Dataset* and the *MovieLens 10M Dataset* are used during the process of evaluation such that, the complete small MovieLens dataset is split into training (80% of the data), and testing (20%) of the data while *MovieLens 100K Dataset* is used entirely for testing purposes only. Essentially, this will imply that the model that will be voted for as the best performing model will be deployed using *MovieLens 100K Dataset.*

### Data Collection

Data was collected from <https://grouplens.org/> which is a public repository managed by GroupLens Research. Essentially, GroupLens Research is a human-computer interaction research lab in the Department of Computer Science and Engineering at the University of Minnesota [67]. The *MovieLens 10M Dataset* is designed for use in research problems since it is a stable benchmark dataset compared to *MovieLens 100K Dataset*.

## Preliminaries

### Network Architecture

The following figure provides an overview of the neural collaborative filtering architecture.

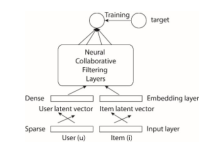


Figure 4; Neural Collaborative filtering model architecture.

As shown in the proposed architecture (*see figure 4 above*), the deep learning architecture focuses on learning about the complex relations that exist among users and the items to generate recommendations.

#### Learning from implicit information

Assume *M* and *N,* are the number of users and items respectively. We define the user-item interaction matrix Y ∈ RM × N from users’ implicit feedback as:

*y*ui =

from the denotation above, *y*ui implies that there was an underlying interaction between user *u* and item *i* k regardless of whether user *u* likes the item. Similarly, 0 denotes the non-interaction of user *u* with item *i* but does not necessarily imply that the user did not like item *i* as it might be due to the user not being aware of the item. This observation presents a challenge for implicit recommender systems as it tends to introduce noisy signals regarding user choices. In practice, while the observed interactions can genuinely reflect a legitimate user interest in such items, the unobserved interactions can as well be missing information and there is just preexisting unavailability of negative feedback.

In implicit feedback recommendation systems, the problem involves estimating the scores of completely unobserved user-item interactions (Y), that are used for ranking items. Ideally, model-based methods make an assumption that the information can be produced or at least be described by the underlying model. Conceptually, the models can be presented as learning:

ui = *f* (*u, i |* Ɵ)

Where:

ui represents the predicted score of the actual interaction *y*ui and both Ɵ and *f* (i.e., the interaction function) denote the model’s parameters and the function that maps the corresponding model’s parameters to the predicted scores. For successful estimation of Ɵ parameters, it follows that deep learning paradigms will be utilized whose objective is to optimize the objective function. The most used objective functions are the point and pairwise loss [62].

Methods on pointwise loss learning act as an extension of explicit-based learning by following a regression framework through minimizing the squared loss between ui and its target value yui [62]. On the other hand, for pairwise loss, the objective is to have the observed entries to be ranked higher relative to unobserved ones. That is, instead of minimizing loss between ui and its target value yui., pairwise loss learning attempts to maximize the margin of the observed value of ui and the unobserved entry of ui. The neural collaborative framework proposed in the current work parametrizes the interaction function *f* using deep learning neural networks to predict the ui observation hence it is assumed to support both pointwise and pairwise loss learning.

### Cosine Similarity Collaborative Filtering

Essentially, cosine similarity is defined as a measure of similarity often between two non-zero vectors [63] and can be denoted mathematically as shown below:

Equation 12



Where x and y are two non-zero vectors x, y ∈ Rn and the dot product x ⋅ y is an operation on the generated vectors that yields a single value [64]. While the two vectors can both be either user or item-based. The vectors in this study will be in the user-item form as presented by unique items and user entries from the data.

Using cosine similarity, the maximum similarity is obtained when the angle between the two vectors is 00 i.e., they are disjointed and they will have zero similarity if the angle between them is 900 i.e., they are orthogonal to each another and have negative similarity if the underlying angle is 1800. During recommendation, for each user *u* for explicit rating, we extract the user’s rating of a movie *i* then multiply it with the cosine similarity of user *k* and user *u*, sum up the weighted ratings, divided by the number of unique users U after which we will have obtained the weighted average rating for movie *j*.

Lastly, the movie recommendations for the users during deployment are sorted by the weighted average rankings which then serve as an estimate for a user’s interaction with a given movie such that movies with higher rankings have a higher likelihood of being favored by the user in question hence, they are proposed for the user during the recommendation phase. For the implicit model, the objective is to generate a user-item matrix then find the closest based on the similarity matrix. By normalizing the matrix of each user, we obtain the percentage importance of movies to a given user, and using this information, the cosine similarity is computed.

### Matrix Factorization

In the current work, matrix factorization associates each user and item to a real-valued vector comprised of latent features. Assume *p*u and *qi* are the latent vectors for both user *u* and item *i* respectively. Matrix vectorization computes the interaction yui as the inner product of pu and qi as shown in equation 13 below.

Equation 13



Where:

K represents the dimension of the latent space. As we have illustrated above (*see equation 13*), matrix factorization models the two-way interaction between the user and the item latent factors by assuming that each dimension of the latent space is primarily independent of each other thus linearly combines both the dimensions with the same weight. Theoretically, this implies that the matrix factorization approach is a linear model of the latent factors [62]. In this study, the similarity between two users for the non-negative matrix factorization is measured by viewing the user matrix as the weight matrix i.e., how much an underlying user is into each latent factor while the item matrix is viewed as the component matrix that is, “…how much the item is consisted of from each latent factor” [65].

## Evaluation Metrics

We will evaluate the performance of the models using the two metrics included in table 3 below.

Table 3: Model evaluation metrics

|  |  |  |  |
| --- | --- | --- | --- |
| Evaluation Metric | Summary | Formula | References |
| Recall | Computes the rate at which the recommender makes positive recommendations |  | [26] |
| Mean Percentile Ranking | Computes the  percentile of an item *i* provided in the list of all the items, and then calculates the average percentile. Primarily, this implies that if one  made an allocation of recommendations at random, then the expected value of the MPR would be 50%. | Where:  r(i,j) denotes the rating of item *i* by user *u*  ui denotes the percentile rank of *i* in an ordered list such that ui = 0% implies that *i* is at the top of the list | [49, 50] |

The deep learning model with the highest Recall and the lowest Mean Percentile Ranking score is considered as the optimal model.

## Experiments

We set up our experiments with the objective of answering the following research questions:

1. How does the implicit-based simple cosine similarity model perform given various performance evaluation metrics?
2. How does the implicit-based non-negative matrix factorization model perform?
3. How does the implicit-based deep learning model perform?
4. How does the explicit-based deep learning model perform?
5. Compared to the benchmark model (non-negative matrix factorization), how does the implicit-based model perform?
6. Does increasing the size of data improve the performance of the collaborative filtering models?

In the following subsection, we present the experimental settings that were adopted including the general information regarding the datasets and the models as well as the evaluation protocols.

#### Datasets

We recommended the use of three datasets including *MovieLens 10M, Small MovieLens dataset, and learn from sets movies dataset.* Table 4 below provides an overview of the characteristics of the datasets.

Table 4: Dataset information

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset | Interaction | Number of items | Number of users | Sparsity | Sampled? |
| Small MovieLens | 100836 | 9724 | 610 | 0.79561% | Use all observations |
| MovieLens 10M | 10000054 | 10677 | 69878 | 0.56456% | Use all observations |
| Learn from sets data | 327899 | 12443 | 744 | 2.47935 | Use all observations |

To reduce the effect of sparsity on the performance of the recommender systems, we normalized the data i.e., conducted sparse normalization [66] before feeding it to the models and making predictions.

#### Initial Model parameters

As noted earlier, the current study will implement three types of models for the collaborative filtering problem i.e., simple cosine similarity, non-negative matrix factorization as well as neural collaborative filtering. Figure 5 below provides an overview of the parameter settings adopted for the initial neural collaborative model.

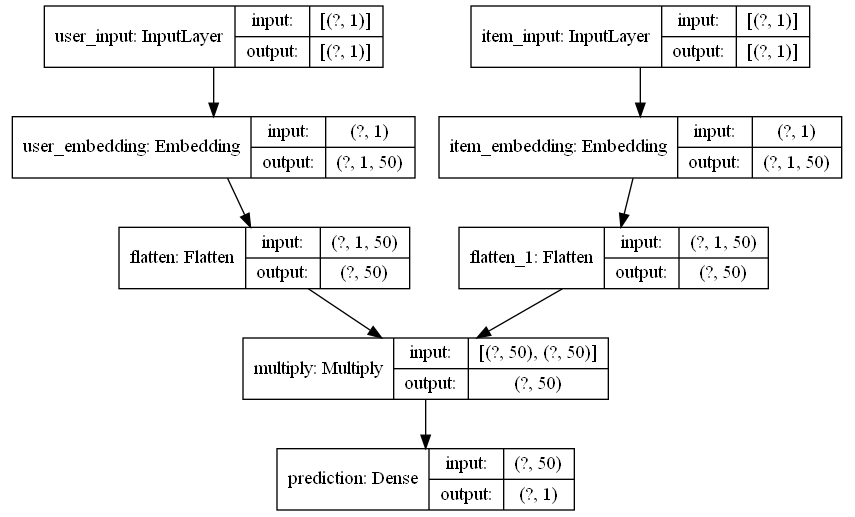


Figure 5: Structure of the initial neural collaborative filtering model

From figure 5 above, the initial model was defined using 5 layers with user and item input being supplied to the input layer both of which are transformed into user and item embeddings and fed into the embedding layer that is then converted to latent features from which the model learns and makes a prediction of what items a user would interact with.

#### Expected results

We expect that the models will perform relatively better when trained using the 10M benchmark dataset given that there will be more interactions from which to learn. Besides, we expect that the benchmark non-negative matrix factorization will perform relatively well compared to the other models. We also expect that the model will underperform when trained with the “Learning from sets” dataset given that it has fewer examples. That is, the current study will seek to test the hypothesis that the size of data and sparsity affect the performance of recommender systems.

## Research Limitations

Whereas the current study uses secondary data sources, the small 100k dataset is not suitable for publication in research as it keeps changing. Therefore, since the 10M MovieLens data is stable and suitable for research publication as per the official GroupLens website, the results from the small dataset are used for comparison purposes while the results from the 10M MovieLens data will be used to address the original research objectives. Moreover, the 10M dataset led to memory error, to handle this, we used ratings from the top 5 movie genres to generate models for each of the three categories of models.

# Chapter 4: Findings and Analysis

## Introduction

Following the current work’s objective i.e., to propose a deep learning-based recommender system for the recommendation of movies and deploy the optimal model on a web-based platform for use in production. In this study, both explicit and implicit-based deep learning models are proposed following the need to evaluate how well each of the models performs when making recommendations on movies to watch given a user’s ID. Moreover, the study proposes a non-negative Matrix Factorization implicit-based recommender system as a benchmark model for the implicit deep learning model. Also, the current work focus on deep learning in explicitly Collaborative Filtering-based approaches.

In addition, by the end of this chapter, it is expected that the following questions will be answered:

1. How does the implicit-based simple cosine similarity model perform given various performance evaluation metrics?
2. How does the implicit-based non-negative matrix factorization model perform?
3. How does the implicit-based deep learning model perform?
4. How does the explicit-based deep learning model perform?
5. Compared to the benchmark model (non-negative matrix factorization), how does the implicit-based model perform?
6. Does increasing the size of data improve the performance of the collaborative filtering models?

### Hypothesis

#### H0

There is a significant association between sparsity and the performance of implicit-based collaborative filtering models?

#### H1

Sparsity does not affect the performance of implicit-based collaborative recommender systems.

## Demographic Information

#### Movie distribution per genre

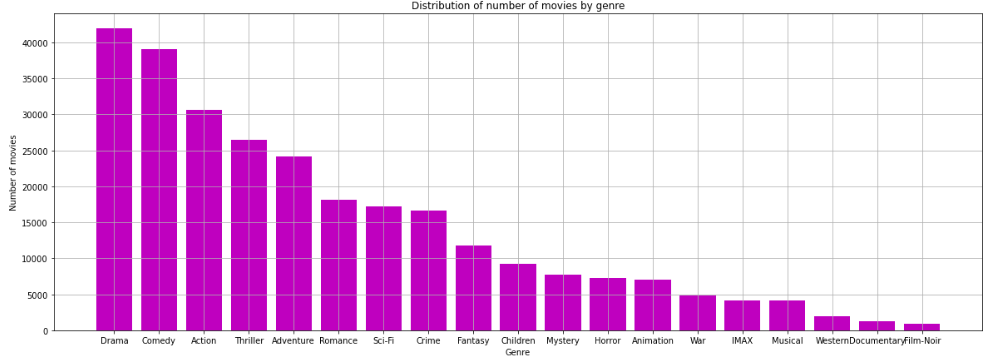


Figure 6: Distribution of the number of movies by genre (100K data)

From the MovieLens 100K data, it is noted that the top 3 most-watched movies included are categorized under drama, comedy, and Thriller while movies categorized under IMAX, Western, and Film-Noir genres are the least-watched movies (*see figure 6*). Similarly, in movies from figure 7 below, it is observed that drama, comedy, and thrillers are the most-watched movies in the sampled 10M dataset.

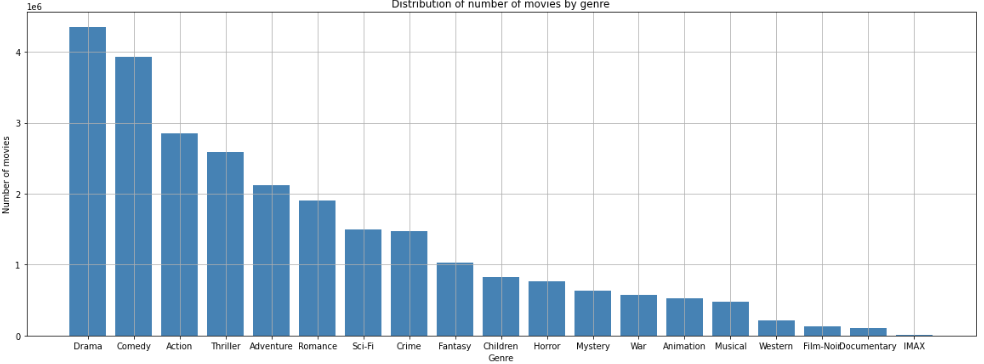


Figure 7: Distribution of the number of movies by genre (10M sampled data)

#### Rating distribution

Most of the users from the 100K data rated most of the movies approximately 4, 3, 3.5, and 5 respectively (see figure 8 below) while those from the sampled 10M dataset assigned a rating of 4, 3, and 5 respectively.

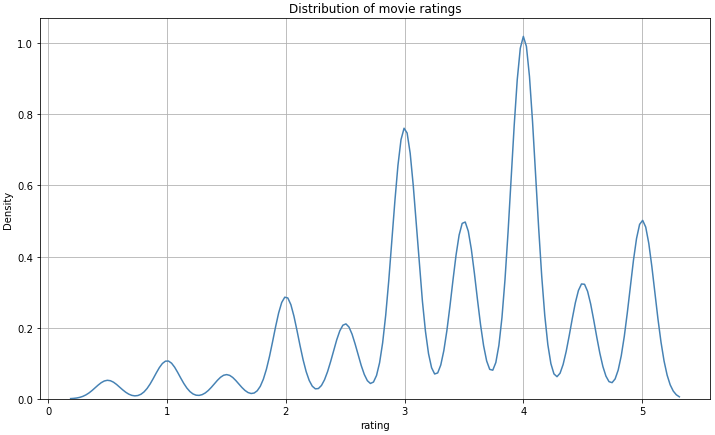


Figure 8: Distribution of movie ratings (100K dataset)

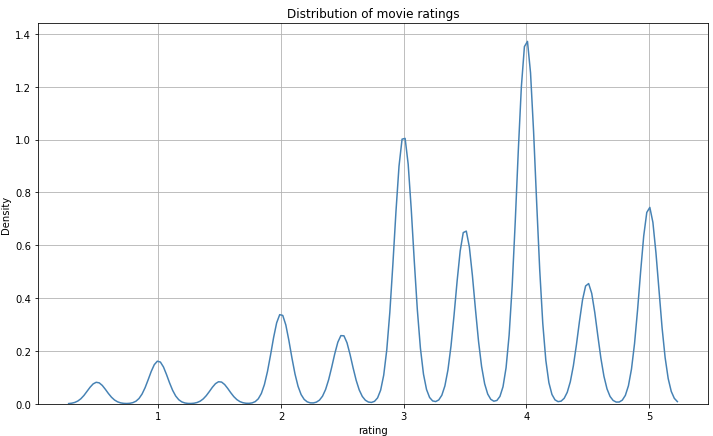


Figure 9: Distribution of movie ratings (10M dataset)

However, for the “learn from sets” dataset, most of the 4, 3.5, and 3 respectively as shown in figure 10 below.

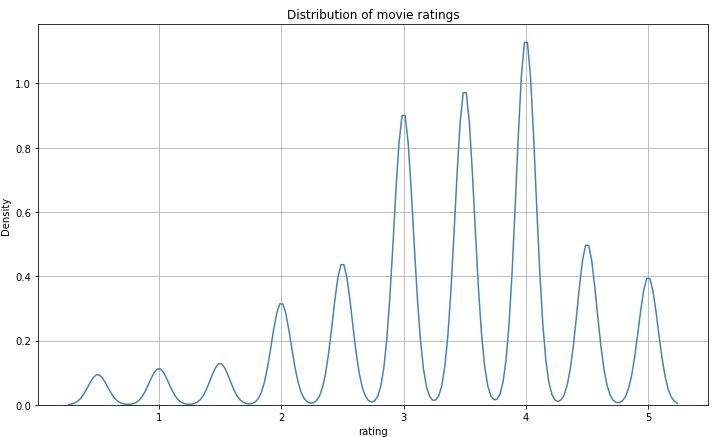


Figure 10: Distribution of movie ratings (learn from sets data)

## Descriptive Statistics

Examining the distribution of the top five genres in the 100K dataset (*see table 5 below*), we note that most of the users (32.7%) who rated Drama movies allocated a rating score of 4.0 while those who rated Comedy movies (27.4%) allocated a rating score of 3.0 while those who rated Action movies i.e., 26.8% allocated a rating of 4.0. 28.2% rated Thriller movies as 4.0 while 22.6% of the users rated romance movies as 4.0.

Table 5: Summary statistics for 100K movies

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Grouped by genres | | | | | | | |
|  |  | **Missing** | **Overall** | **Action** | **Comedy** | **Drama** | **Romance** | **Thriller** | **P-Value** |
|  |  |  |  |  |  |  |  |  |  |
| n |  |  | 14363 | 186 | 7196 | 6291 | 62 | 628 |  |
| rating, n (%) | **0.5** | 0 | 191 (1.3) | 4 (2.2) | 136 (1.9) | 44 (0.7) |  | 7 (1.1) | <0.001 |
| **1.0** |  | 491 (3.4) | 13 (7.0) | 348 (4.8) | 99 (1.6) | 6 (9.7) | 25 (4.0) |  |
| **1.5** |  | 326 (2.3) | 6 (3.2) | 256 (3.6) | 49 (0.8) | 3 (4.8) | 12 (1.9) |  |
| **2.0** |  | 1197 (8.3) | 31 (16.7) | 828 (11.5) | 287 (4.6) | 6 (9.7) | 45 (7.2) |  |
| **2.5** |  | 854 (5.9) | 13 (7.0) | 515 (7.2) | 295 (4.7) | 5 (8.1) | 26 (4.1) |  |
| **3.0** |  | 2978 (20.7) | 51 (27.4) | 1614 (22.4) | 1144 (18.2) | 12 (19.4) | 157 (25.0) |  |
| **3.5** |  | 1787 (12.4) | 22 (11.8) | 854 (11.9) | 830 (13.2) | 11 (17.7) | 70 (11.1) |  |
| **4.0** |  | 3859 (26.9) | 27 (14.5) | 1586 (22.0) | 2055 (32.7) | 14 (22.6) | 177 (28.2) |  |
| **4.5** |  | 1047 (7.3) | 11 (5.9) | 403 (5.6) | 593 (9.4) | 3 (4.8) | 37 (5.9) |  |
| **5.0** |  | 1633 (11.4) | 8 (4.3) | 656 (9.1) | 895 (14.2) | 2 (3.2) | 72 (11.5) |  |

Table 6 below provides an overview of the % of users who rated movies from the top five genres in the 10M dataset.

Table 6: Summary statistics for 10M movies

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Grouped by genres | | | | | | | |
|  |  | **Missing** | **Overall** | **Action** | **Adventure** | **Comedy** | **Drama** | **Thriller** | **P-Value** |
|  |  |  |  |  |  |  |  |  |  |
| n |  |  | 1728711 | 27208 | 2549 | 778596 | 815084 | 105274 |  |
| rating, n (%) | **0.5** | 0 | 16147 (0.9) | 428 (1.6) | 53 (2.1) | 11301 (1.5) | 3895 (0.5) | 470 (0.4) | <0.001 |
| **1.0** |  | 76002 (4.4) | 2666 (9.8) | 245 (9.6) | 50350 (6.5) | 19796 (2.4) | 2945 (2.8) |  |
| **1.5** |  | 19674 (1.1) | 614 (2.3) | 62 (2.4) | 13280 (1.7) | 4938 (0.6) | 780 (0.7) |  |
| **2.0** |  | 145057 (8.4) | 3750 (13.8) | 365 (14.3) | 86532 (11.1) | 46460 (5.7) | 7950 (7.6) |  |
| **2.5** |  | 61721 (3.6) | 1400 (5.1) | 137 (5.4) | 35875 (4.6) | 20886 (2.6) | 3423 (3.3) |  |
| **3.0** |  | 422363 (24.4) | 9217 (33.9) | 698 (27.4) | 209441 (26.9) | 175215 (21.5) | 27792 (26.4) | |
| **3.5** |  | 142369 (8.2) | 1763 (6.5) | 243 (9.5) | 65123 (8.4) | 66364 (8.1) | 8876 (8.4) |  |
| **4.0** |  | 500639 (29.0) | 5344 (19.6) | 524 (20.6) | 192115 (24.7) | 267806 (32.9) | 34850 (33.1) | |
| **4.5** |  | 88418 (5.1) | 434 (1.6) | 69 (2.7) | 31050 (4.0) | 52679 (6.5) | 4186 (4.0) |  |
| **5.0** |  | 256321 (14.8) | 1592 (5.9) | 153 (6.0) | 83529 (10.7) | 157045 (19.3) | 14002 (13.3) | |

From table 7 below it is observed that on average, the ratings for the different genres are statistically different implying that the users who had watched the corresponding movies had a varying voting distribution with action movies being given the lowest average vote i.e., 2.9 while drama movies were voted the highest i.e., 3.7.

Table 7: Test for trends in ratings (10M dataset)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Grouped by genres | | | | | | | |
|  |  | **Missing** | **Overall** | **Action** | **Adventure** | **Comedy** | **Drama** | **Thriller** | **P-Value** |
|  |  |  |  |  |  |  |  |  |  |
| n |  |  | 1728711 | 27208 | 2549 | 778596 | 815084 | 105274 |  |
| genres, n (%) | **Action** | 0 | 27208 (1.6) | 27208 (100.0) | |  |  |  | <0.001 |
| **Adventure** |  | 2549 (0.1) |  | 2549 (100.0) | |  |  |  |
| **Comedy** |  | 778596 (45.0) | |  | 778596 (100.0) | |  |  |
| **Drama** |  | 815084 (47.1) | |  |  | 815084 (100.0) | |  |
| **Thriller** |  | 105274 (6.1) | |  |  |  | 105274 (100.0) | |
| rating, mean (SD) |  | 0 | 3.5 (1.1) | 2.9 (1.1) | 3.0 (1.1) | 3.2 (1.1) | 3.7 (1.0) | 3.5 (1.0) | <0.001 |

However, whereas there a statistically significant difference in the voting patterns for users who watched movies from the top 5 genres for the 100K dataset, it is noted that on average, voters rated movies categorized under Adventure as the least favorite among the five genres i.e., with a rating score of 2.7 while Drama movies were highly rated i.e., with a rating score of 3.7 (*see table 8*).

Table 8: Test for trends in ratings (100K dataset)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Grouped by genres | | | | | | | |
|  |  | **Missing** | **Overall** | **Action** | **Adventure** | **Comedy** | **Drama** | **Thriller** | **P-Value** |
|  |  |  |  |  |  |  |  |  |  |
| n |  |  | 14320 | 186 | 19 | 7196 | 6291 | 628 |  |
| genres, n (%) | **Action** | 0 | 186 (1.3) | 186 (100.0) | |  |  |  | <0.001 |
| **Adventure** |  | 19 (0.1) |  | 19 (100.0) |  |  |  |  |
| **Comedy** |  | 7196 (50.3) | |  | 7196 (100.0) | |  |  |
| **Drama** |  | 6291 (43.9) | |  |  | 6291 (100.0) | |  |
| **Thriller** |  | 628 (4.4) |  |  |  |  | 628 (100.0) | |
| rating, mean (SD) |  | 0 | 3.4 (1.1) | 2.9 (1.1) | 2.7 (1.0) | 3.2 (1.1) | 3.7 (0.9) | 3.4 (1.0) | <0.001 |

## Recommender Experiments

The following subsection explores the experimentation of the three implicit-based collaborative filtering recommender systems i.e., non-negative matrix factorization (NMF), cosine similarity, and neural collaborative filtering using the three datasets. The performance of the three models is evaluated using both Recall and the Mean Percentile Ranking Score. Ideally, the model with the highest recall and low MPR is considered the best performing.

Sparsity was observed to be relatively low for the 100K dataset i.e., approximately 0.79561%. the data was preprocessed by transforming the numerical movie and user IDs to categorical and split using a 70:30 ratio for the train and test data respectively. Since based on the original objective of developing an implicit-based model, the data was subset to contain only observations related to movies (items) and user IDs. To handle the memory error, the model was trained using 30 chunks. Table 9 below provides an overview of the performance of the cosine similarity model compared to the benchmark non-negative Matrix factorization model.

Table 9: Cosine Similarity model performance vs non-negative Matrix factorization

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Model | Recall | MPR |
| Cosine Similarity | 0.59451 | 0.42323 |
| Non-negative Matrix factorization | 0.60521 | 0.40487 |

On average, it is noted from Table 9 above that the cosine similarity model (*Recall = 59.451%, MPR = 42.323%)* underperforms the NMF model (*Recall = 56.774%, MPR = 44.224%)* when trained with the 100K dataset that as noted had a sparsity of approximately 0.79561%.

Studies such as

Similarly, table 10 below shows the performance of the neural collaborative filtering model compared to the non-negative Matrix factorization benchmark model.

Table 10: Neural collaborative filtering model performance vs non-negative Matrix factorization

|  |  |  |
| --- | --- | --- |
| Model | Recall | MPR |
| Neural collaborative filtering | 0.50442 | 0.49279 |
| Non-negative Matrix factorization | 0.60521 | 0.40487 |

The neural collaborative filtering model scores an MPR of approximately 49.991% which is about 0.009% closer to making random recommendations compared to the MPR model which attains an MPR of 40.487% implying that the NMF model performs better in terms of making relevant decisions.

On the other hand, the sparsity for the 10M MovieLens dataset was observed to be approximately 0.56456% which is relatively lower than that of the 100K dataset implying that it had more information from which the models would learn. Table 11 below provides an overview of the performance of the cosine similarity model compared to the benchmark non-negative matrix factorization.

Table 11:Cosine Similarity model performance vs non-negative Matrix factorization (10M dataset)

|  |  |  |
| --- | --- | --- |
| Model | Recall | MPR |
| Cosine Similarity | 0.64690 | 0.40586 |
| Non-negative Matrix factorization | 0.69477 | 0.37234 |

As noted from table 11 above, the performance of the cosine similarity improves slightly from an MPR of approximately 0.42323 to approximately 0.40586 implying that the model makes more relevant predictions compared to the model trained with the 100K dataset. Similarly, the neural collaborative model attains an MPR of approximately 0.49250 (*see table 12 below*) a negligible improvement from approximately 0.49250 for the 100K dataset model.

Table 12: Neural collaborative filtering model performance vs non-negative Matrix factorization (10M dataset)

|  |  |  |
| --- | --- | --- |
| Model | Recall | MPR |
| Neural collaborative filtering | 0.51197 | 0.49250 |
| Non-negative Matrix factorization | 0.69477 | 0.37234 |

The NMF model has a relatively high improvement in performance when trained with the 10M dataset. Ideally, since the only variable between the model trained in the 100K dataset and the 10M dataset is the size of the data and sparsity, we can argue that the NMF model is highly affected by sparsity compared to both the neural collaborative filtering model and cosine similarity.

To further test whether sparsity affects the performance of the collaborative filtering models, the models were tested on the “learning from sets 2019) data which had the highest sparsity of approximately 2.4793%. The performance of the cosine similarity and the benchmark NMF are given in table 13 below.

Table 13: Cosine Similarity model performance vs non-negative Matrix factorization (learn from sets data)

|  |  |  |
| --- | --- | --- |
| Model | Recall | MPR |
| Cosine Similarity | 0.52412 | 0.47469 |
| Non-negative Matrix factorization | 0.53403 | 0.46404 |

As shown in Table 13, the cosine similarity model attains the lowest recall score (52.412%) and highest MPR (47.569%) compared to the other three cosine similarity models, when trained using the *learning from sets* data. A similar trend is observed for the NMF model with an MPR of approximately 46.404% and 53.403% the lowest NMF score of the three models with similar observations being made for the neural collaborative filtering model i.e., MPR of approximately 49.794% and a recall of 50.034% (*see table 14*).

Table 14: Neural collaborative filtering model performance vs non-negative Matrix factorization (learn from sets data)

|  |  |  |
| --- | --- | --- |
| Model | Recall | MPR |
| Neural collaborative filtering | 0.50034 | 0.49794 |
| Non-negative Matrix factorization | 0.53403 | 0.46404 |

#### Graphical comparison of the relationship between sparsity, recall, and MPR

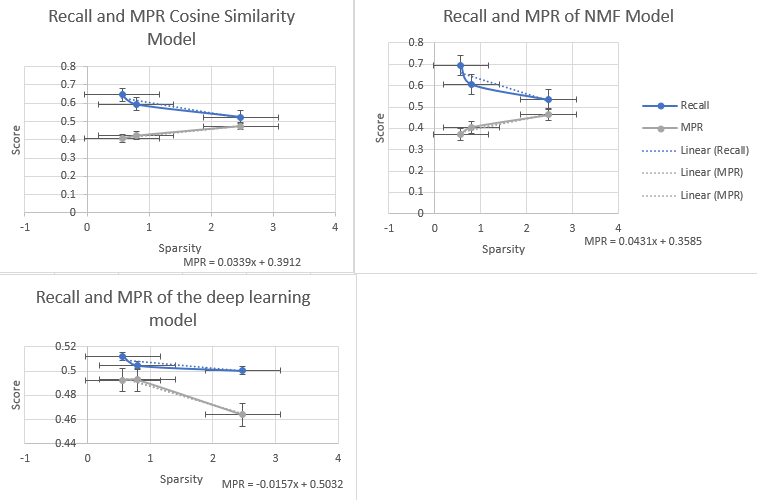


Figure 11: Graphical comparison of how each model performs under various levels of data sparsity.

Generally, the MPR of the NMF model as noted from figure 11 above increases by approximately 0.4016 (*40.16%*) with an increase in the sparsity by one unit using the regression equation *MPR* = *0.0431x + 0.3585*. Sparsity of the NMF model had a strong positive correlation with MPR (*r* = 0.9697). On the other hand, the MPR of the cosine similarity model increases by approximately 0.4251 (*42.51%*) with an increase of sparsity by a single unit with a strong positive correlation (*r* = 0.9909). Contrary, the MPR of the neural collaborative model decreases by approximately -0.0157 (*1.57%*) with an increase in the sparsity of data by 1 unit. The NCF’s MPR remains relatively high regardless of the slight decrease with an observed MPR of 48.75% per unit change in data sparsity with a strong negative association with MPR (*r* = -0.9928) and a strong positive association with Recall (*r* = 0.7629).

# Chapter 4: Conclusion

By evaluating the performance of the cosine similarity and neural collaborative filtering models against the benchmark non-negative matrix factorization (NMF) model and by evaluating the effect of sparsity on each of the models, in light of the findings of this research, it can be concluded that there are fundamental differences in the recommendation approaches of the three models. The research has demonstrated that on average, each of the three models make actual recommendations and not just assign random movie recommendations to users as demonstrated using the MPR which was noted to be such that 0% is the best, 100% is the worst, anything over 50% is worse than random guessing. Besides since the neural network model depends on sparse vectors to learn, the research findings show that on average, the performance of the neural model tends to increase with an increase in data sparsity.

Conceptually, data sparsity arises from the phenomenon that users, in general, tend to rate only a limited number of items and since the functionality of CF-based recommenders is to aggregate the ratings of like-minded users, the fact that most user-item matrices are very sparse with a sparsity of up to 99% as a result of lack of knowledge by the users or lack of incentives to rate the items meaning that there is little information from which to learn. The proposed deep learning model handles the problem of lack of sufficient information in the data relatively well especially given the fact that an increase in sparsity does not change the mean percentile rank as well as the recall dramatically compared to the cosine similarity model as well as the non-negative matrix factorization model that are on average highly sensitive to sparsity.

## Performance of Neural Collaborative Filtering Compared to Memory-based models

### Matrix Factorization

Embedding based models such as matrix factorization which have been considered state-of-art in collaborative filtering. Traditionally, such models have used the dot product or higher order equivalent during the combination of two or more embeddings for example most notably in matrix factorization. In neural collaborative filtering, the dot product is replaced by learned similarity such as a multilayer perceptron (MLP) hence the process of neural collaborative filtering. Previous studies support the argument that overall matrix factorization recommender systems will tend to outperform deep learning models [67]. In this research, the findings suggest that the dot product still outperforms the learned similarity with the NMF attaining better recommendation performance especially with low sparsity present in the data.

It is understood that using the secondary research using the benchmark datasets regarding the effect of sparsity show that there is a negative association between sparsity and the performance of matrix factorization models while neural collaborative models have a positive association with sparsity an indication of the variation in functionality between the two classes of recommender systems.

### Model Performance

The current research has shown that memory-based models under which cosine filtering is categorized tend to perform better with variation in sparsity as opposed to the size of data. Since in the current study data sparsity had the strongest effect on the performance of both memory-based models, it can be argued that to optimize the performance of memory-based models it is important to handle the problem of sparsity depending on the adopted model. Evidence shows that on average, the cosine similarity model underperforms the NMF model but performs better than the neural model.

The problem with data sparsity regardless of the stable performance of neural models was observed to be related to consumption of more resources including memory and time taken to train the models. Other factors such as data preprocessing approaches adopted including normalization to handle sparsity were noted to affect the performance of memory-based models i.e., both the NMF and Cosine Similarity models had better performance when fitted with normalized data implying that besides sparsity the underlying data preprocessing played a role on the performance of the models.

## Recommendations

Following the completion of the research, the following recommendations are made:

#### Data preprocessing

Data preprocessing has a significant effect on the quality of data and since the performance of both machine and deep learning models depend on the quality of data that they are fed with to generate corresponding predictions and in the case of this research, recommendations, it is important to adopt appropriate data preprocessing methods such as removing bias brought about by lack of interactions between users and items i.e., handling sparsity through data preprocessing.

#### Model Development

Often, neural models perform relatively better when their learning capabilities have been optimized. One way with which to improve the learning capability of neural models is through hyperparameter optimization such as tuning for the optimal number of layers, learning rate, number of epochs, etcetera. In this research, the neural model was fitted with 5 layers which arguably were not enough to learn and generate meaningful insights about the user-item latent features. It is therefore proposed that deep learning models for collaborative filtering recommender systems should be fitted with sufficient layers to enable efficient learning and hence generation of relevant item recommendations to users.

#### Hybrid Models

Hypothetically, combining the ability of using sparsity to advantage and using dot products would yield to an optimally performing model. In the current research, the NMF model which follows learning by embeddings was noted to perform well but was affected by sparsity. The neural model on the other had had lower performance relative to the NMF model but did not display volatility with the variation of sparsity in data. As such, combining the power of the two models would allow the handling of both sparsity and the effect of learned similarity in user-item interaction. That is, hypothetically, a hybrid model between NMF and deep learning would yield better results.

# References

|  |  |
| --- | --- |
| [1] | V. Shchutskaya and K. Spirina, “Big Data Behind Recommender Systems,” InData Labs, 2019. |
| [2] | Q. Zhang, J. Lu and Y. Jin, “Artificial intelligence in recommender systems,” *Complex Intell. Syst.,* vol. 7, no. 2021, pp. 439-457, 2021. |
| [3] | F. Isinkaye, Y. Folajimi and B. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” *Egyptian Informatics Journal,* vol. 16, no. 3, pp. 261-273, 2015. |
| [4] | Section IO, “Recommender System Explained,” 2021. [Online]. Available: https://www.section.io/engineering-education/recommender-system-explained/. [Accessed 10 August 2021]. |
| [5] | J. Gu, “Building a Deep-Learning-Based Movie Recommender System,” 2020. [Online]. Available: https://dzone.com/articles/building-a-deep-learning-based-movie-recommender-s. [Accessed 12 August 2021]. |
| [6] | J. Redpath, D. Glass, S. McClean and L. Chen, “Collaborative Filtering: The Aim of Recommender Systems and the Significance of User Ratings,” in *Advances in Information Retrieval*, G. C. e. al, Ed., Berlin, Springer, 2010. |
| [7] | R. Gaston, “Introduction to Recommender Systems,” 2021. [Online]. Available: https://tryolabs.com/blog/introduction-to-recommender-systems/. [Accessed 10 August 2021]. |
| [8] | J. Konstan and J. Riedl, “Recommender systems: from algorithms to user experience,” *User Model User-Adapt Interact,* vol. 22, no. 2012, pp. 101-123, 2012. |
| [9] | C. Pan and W. Li, “Research paper recommendation with topic analysis,” *In Computer Design and Applications IEEE,* vol. 4, no. 2010, pp. V4-264, 2010. |
| [10] | U. Sivarajah, M. M. Kamal, Z. Irani and V. Weerakkody, “Critical analysis of Big Data challenges and analytical methods,” *Journal of Business Research,* vol. 70, no. 2017, pp. 263-286, 2017. |
| [11] | D. Singh and C. Reddy, “A survey on platforms for big data analytics,” *Journal of Big Data,* vol. 2, no. 2015, p. 8, 2015. |
| [12] | B. Opeyemi, “Deploy a Machine Learning Model as a Web Application (Part 1),” 2020. [Online]. Available: https://heartbeat.fritz.ai/deploy-a-machine-learning-model-as-a-web-application-part-1-a1c1ff624f7a. [Accessed 12 August 2021]. |
| [13] | T. Shin, “Five Technologies to Deploy Your Machine Learning Models,” 2021. [Online]. Available: https://towardsdatascience.com/five-technologies-to-deploy-your-machine-learning-models-bddaad69e0d4. [Accessed 12 August 2021]. |
| [14] | K. A. Reshak, B. N. Dhannon and Z. N. Sultani, “Explicit Feedback Based Movie Recommendation System: a Survey,” in *The 8th International Conference on Applied Science and Technology (ICAST 2020)*, 2020. |
| [15] | R. Saccomandi1, “Recommender Systems - Understanding​ Implicit Feedback,” 2020. [Online]. Available: https://blog.kickdynamic.com/recommender-systems-understanding-implicit-feedback. [Accessed 12 August 2021]. |
| [16] | M. N. Saadat and M. Shuaib, “Advancements in Deep Learning Theory and Applications: Perspective in 2020 and beyond,” in *Advances and Applications in Deep Learning*, M. A. Aceves-Fernandez, Ed., IntechOpen, 2020. |
| [17] | D. L. Yohn, “6 Ways to Build a Customer-Centric Culture,” Harvard Business Review, 2018. |
| [18] | M. Holmlund, Y. V. Vaerenbergh, R. Ciuchita and A. Ravald, “Customer experience management in the age of big data analytics: A strategic framework,” *Journal of Business Research,* vol. 116, no. 2020, pp. 356-365, 2020. |
| [19] | Tiempo Development, “Data Analytics Role in Improving Customer Experience,” 2020. [Online]. Available: https://www.tiempodev.com/blog/data-analytics-role-in-improving-customer-experience/. [Accessed 13 August 2021]. |
| [20] | D. K. Sahu, “What is Big Data?,” 2020. [Online]. Available: https://www.digitalvidya.com/blog/big-data-applications/. [Accessed 13 August 2021]. |
| [21] | N. Joshi, “3 applications of Deep Learning in Big Data analytics,” Allerin, 2017. |
| [22] | M. Shakourifar, “The math behind...Online Recommender Systems,” Society for Industrial and Applied Mathematics, Ontario, Canada, 2012. |
| [23] | W. Jackson, “Types of Recommender Systems,” Medium, 2019. |
| [24] | J. Shen, Y. Wei and Y. Yang, “Collaborative filtering recommendation algorithm based on two stages of similarity learning and its optimization,” in *13th IFAC Symposium on Large Scale Complex Systems: Theory and Applications*, Shanghai, China, 2013. |
| [25] | X. Su and T. M. Khoshgoftaar, “A Survey of Collaborative Filtering Techniques,” *Advances in Artificial Intelligence,* vol. 12, no. Article ID 421425, 2009. |
| [26] | R. Chen, Q. Hua, Y. Chang and B. Wang, “A survey of collaborative FIltering-based recommender systems: From traditional methods to hybrid methods based on social networks,” *IEEE Access,* vol. 6, no. 2018, p. 64301–64320, 2018. |
| [27] | C. Aggarwal, Recommender Systems, Volume 1 ed., Cham, Switzerland: Springer International Publishing, 2016. |
| [28] | M. Srifi, A. Oussous, A. Ait Lahcen and S. Mouline, “Recommender Systems Based on Collaborative Filtering Using Review Texts—A Survey,” *Information,* vol. 11, no. 2020, p. 317, 2020. |
| [29] | J. Bobadilla, F. Ortega, A. Hernando and A. Gutiérrez, “Recommender systems survey,” *Knowl. Based Syst.,* vol. 46, no. 2013, pp. 109-132, 2013. |
| [30] | S. Gupta, “Top 5 Distance Similarity Measures implementation in Machine Learning,” 2019. [Online]. Available: https://medium.com/@gshriya195/top-5-distance-similarity-measures-implementation-in-machine-learning-1f68b9ecb0a3. [Accessed 13 August 2021]. |
| [31] | R. Francesco, L. Rokach and B. Shapira, “Recommender systems: Introduction and challenges,” in *Recommender Systems Handbook*, Boston, MA, USA, Springer, 2015, pp. 1-34. |
| [32] | D. Christian and G. Karypis, “A comprehensive survey of neighborhood-based recommendation methods,” in *Recommender Systems Handbook*, Boston, MA, USA, Springer, 2011, pp. 107-144. |
| [33] | L. Greg, B. Smith and J. York, “Amazon. com recommendations: Item-to-item collaborative filtering,” *IEEE Internet Comput.,* vol. 7, no. 2003, pp. 76-80, 2003. |
| [34] | V. Profillidis and G. Botzoris, “Chapter 5 - Statistical Methods for Transport Demand Modeling,” in *Modeling of Transport Demand*, V. Profillidis and G. Botzoris, Eds., Elsevier, 2019, pp. 163-224. |
| [35] | V. Arnaboldi, A. Passarella, M. Conti and R. I. Dunbar, “Evolutionary Dynamics in Twitter Ego Networks,” in *Computer Science Reviews and Trends*, V. Arnaboldi, A. Passarella, M. Conti and R. I. Dunbar, Eds., Elsevier, 2015, pp. 75-92. |
| [36] | J. Lee, M. Sun and G. Lebanon, “A Comparative Study of Collaborative Filtering Algorithms,” arXiv, 2012. |
| [37] | Z. Yang, B. Wu, K. Zheng and X. Wang, “A survey of collaborative filtering-based recommender systems for mobile internet application,” *IEEE Access,* vol. 1, no. 2016, pp. 3273-3287, 2016. |
| [38] | S.-T. Wang and M.-H. Li, “Mobile Phone Recommender System Using Information Retrieval Technology by Integrating Fuzzy OWA and Gray Relational Analysis,” *Information,* vol. 9, no. 2018, p. 326, 2018. |
| [39] | S. Zhang, L. Yao, A. Sun and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM Comput. Surv.,* vol. 52, no. 2018, pp. 1-38, 2019. |
| [40] | Z. Xia, Y. Dong and G. Xing, “Mobile Phone Recommender System Using Information Retrieval Technology by Integrating Fuzzy OWA and Gray Relational Analysis,” *Information,* vol. 9, no. 2018, p. 326, 2018. |
| [41] | K. Yehuda, R. Bell and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer,* vol. 42, no. 2009, pp. 30-37, 2009. |
| [42] | L. Qiu, S. Gao, W. Cheng and J. Guo, “Aspect-based latent factor model by integrating ratings and reviews for recommender system,” *Knowl. Based Syst,* vol. 110, no. 2016, pp. 233-243, 2016. |
| [43] | F. G. Toudeshki, “Non-Negative Matrix Factorization (NMF) for Recommender Systems,” 2020. [Online]. Available: https://medium.com/@farnazgh73/non-negative-matrix-factorization-nmf-4798d29e8c62#:~:text=Non%2DNegative%20Matrix%20Factorization%20(NMF)%20for%20Recommender%20Systems,-Farnaz%20Ghasemi%20Toudeshki&text=Matrix%20factorization%20is%20a%20collaborative,is%2. [Accessed 14 August 2021]. |
| [44] | H. Han, M. Huang, Y. Zhang and U. Bhatti, “An Extended-Tag-Induced Matrix Factorization Technique for Recommender Systems,” *Information,* vol. 9, no. 2018, p. 143, 2018. |
| [45] | I. Rabiu, N. Salim, A. Da’u and A. Osman, “Recommender System Based on Temporal Models: A Systematic Review,” *Appl. Sci.,* vol. 10, no. 2020, p. 2204, 2020. |
| [46] | F. Ricci, L. Rokach, B. Shapira and P. Kantor, Recommender Systems Handbook, 1st ed., Berlin, Germany: Springer, 2011. |
| [47] | D. Garg, P. Gupta, P. Malhotra and L. Vig, “Sequence and time aware neighborhood for session-based recommendations: Stan,” in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Paris, France, 2019. |
| [48] | J. Herlocker, J. Konstan, L. Terveen and J. Riedl, “Evaluating collaborative filtering recommender systems,” *ACM Trans. Inf. Syst.,* vol. 22, no. 2004, pp. 5-53, 2004. |
| [49] | T. Silveira, M. Zhang, X. Lin and Y. Liu, “How good your recommender system is? A survey on evaluations in recommendation,” *Int. J. Mach. Learn. Cybern,* vol. 10, no. 2019, pp. 813-831, 2019. |
| [50] | D. Parra and S. Sahebi, “Recommender Systems: Sources of Knowledge and Evaluation Metrics,” in *Advanced Techniques in Web Intelligence*, J. V. e. al, Ed., Berlin Heidelberg, Springer-Verlag, 2013, pp. 149-175. |
| [51] | J. Hintz, “Matrix Factorization for Collaborative Filtering Recommender Systems,” 2015. |
| [52] | E. Kirzhner, “Machine Learning. Explanation of Collaborative Filtering vs Content Based Filtering.,” 2018. [Online]. Available: https://codeburst.io/explanation-of-recommender-systems-in-information-retrieval-13077e1d916c. [Accessed 14 August 2021]. |
| [53] | Adpresso Team, “What is content-based Filtering?,” Educative, 2021. |
| [54] | D. Wang, Y. Liang, D. Xu and X. Feng, “A content-based recommender system for computer science publications,” *Knowledge-Based Systems,* vol. 157, no. 2018, pp. 1-9, 2018. |
| [55] | R. Mooney and L. Roy, “Content-based book recommending using learning for text categorization,” in *Proceedings of the Fifth ACM Conference on Digital Libraries*, 2000. |
| [56] | M. Schedl, “Deep Learning in Music Recommendation Systems,” *Frontiers in Applied Mathematics and Statistics ,* vol. 5, no. 2019, p. 44, 2019. |
| [57] | M. Mao, J. Lu, G. Zhang and J. Zhang, “Multirelational social recommendations via multigraph ranking,” *IEEE Trans. Cybern,* vol. 47, no. 12, pp. 4049-4061, 2017. |
| [58] | Q. Shambour and J. Lu, “A trust-semantic fusion-based recommendation approach for e-business applications,” *Decis. Support Syst,* vol. 54, no. 1, pp. 768-780, 2012. |
| [59] | G. Guo, “Resolving Data Sparsity and Cold Start in Recommender Systems,” in *User Modeling, Adaptation, and Personalization*, vol. 7379, J. Masthoff, B. Mobasher, M. Desmarais and R. Nkambou, Eds., Berlin, Heidelberg, Springer, 2012. |
| [60] | H. Hwangbo and Y. Kim, “An empirical study on the effect of data sparsity and data overlap on cross domain collaborative filtering performance,” *Expert Systems with Applications,* vol. 89, no. 2017, pp. 254-265, 2017. |
| [61] | S. Zhang, L. Yao, A. Sun and Y. Tay, “Deep Learning based Recommender System: A Survey and New Perspectives,” *ACM Comput. Surv,* vol. 1, no. 1, p. 35, 2018. |
| [62] | C. McDonald, “How to Build a Deep Learning Powered Recommender System, Part 2,” *Data science,* vol. 12, no. 8, 2021. |
| [63] | J. Lee, “Building a Web Application to Deploy Machine Learning Models,” 2020. [Online]. Available: https://towardsdatascience.com/building-a-web-application-to-deploy-machine-learning-models-e224269c1331. [Accessed 14 August 201]. |
| [64] | S. Jagdale, “Edge AI is Overtaking Cloud Computing for Deep Learning Applications,” 2021. [Online]. Available: https://www.embeddedcomputing.com/technology/iot/edge-computing/edge-ai-is-overtaking-cloud-computing-for-deep-learning-applications. [Accessed 15 August 2021]. |
| [65] | J. Schmidt, M. Marques and S. Botti, “Recent advances and applications of machine learning in solid-state materials science,” *npj Comput Mater,* vol. 5, no. 2019, p. 83, 2019. |
| [66] | M. Sharma, F. Harper and G. Karypis, “Learning from Sets of Items in Recommender Systems,” in *Proceedings of ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2019. |
| [67] | F. M. Harper and J. A. Konstan, “The MovieLens Datasets: History and Context,” *ACM Transactions on Interactive Intelligent Systems (TiiS),* vol. 5, no. 4, p. 19, 2015. |
| [68] | X. He, L. Liao, H. Zhang and L. Nie, “Neural Collaborative Filtering,” *NExT,* vol. 12, no. 5, 2017. |
| [69] | M. I. Khan, “Recommendation Systems for Beginners,” 2020. [Online]. Available: https://affine.ai/recommendation-systems-for-beginners/. [Accessed 16 August 2021]. |
| [70] | L. Metcalf and W. Casey, “Metrics, similarity, and sets,” in *Cybersecurity and Applied Mathematics*, L. Metcalf and W. Casey, Eds., Syngress, 2016, pp. 3-22. |
| [71] | Y.-T. Teddy, “Implicit Feedback Recommendation System (II) — Collaborative Filtering,” 2019. [Online]. Available: https://medium.com/@teddywang0202/implicit-feedback-recommendation-system-ii-collaborative-filtering-27be600197f1. [Accessed 16 August 2021]. |
| [72] | J. Yi, J. Lee, S. J. Hwang and E. Yang, “Sparsity Normalization: Stabilizing the Expected Outputs of Deep Networks,” *DeepAI,* no. 2019, 2019. |
| [73] | S. Rendle, W. Krichene, L. Zhang and J. Anderson, “Neural Collaborative Filtering vs. Matrix Factorization Revisited,” in *RecSys '20: Fourteenth ACM Conference on Recommender Systems*, New York, NY, USA, 2020. |
| [74] | B. Dickson, “What Is Deep Learning?,” 2019. [Online]. Available: https://medium.com/pcmag-access/what-is-deep-learning-5785391fee39#:~:text=Is%20Machine%20Learning%3F-,Deep%20learning%20is%20a%20subset%20of%20machine%20learning%2C%20a%20branch,a%20process%20called%20%E2%80%9Ctraining.%E2%80%9D. [Accessed 14 August 2021]. |

# Appendix

## Correlation statistics

Table 15

|  |  |  |  |
| --- | --- | --- | --- |
| Neural Model | |  |  |
|  | *Sparsity* | *Recall* | *MPR* |
| Sparsity | 1 |  |  |
| Recall | -0.83458 | 1 |  |
| MPR | -0.99286 | 0.762928 | 1 |
|  |  |  |  |
| Cosine Similarity | |  |  |
|  |  |  |  |
|  | *Sparsity* | *Recall* | *MPR* |
| Sparsity | 1 |  |  |
| Recall | -0.94656 | 1 |  |
| MPR | 0.990992 | -0.98123 | 1 |
|  |  |  |  |
| Non-Negative Matrix Factorization | | | |
|  | *Sparsity* | *Recall* | *MPR* |
| Sparsity | 1 |  |  |
| Recall | -0.88757 | 1 |  |
| MPR | 0.969733 | -0.97318 | 1 |

## Neural Model Performance per Epoch

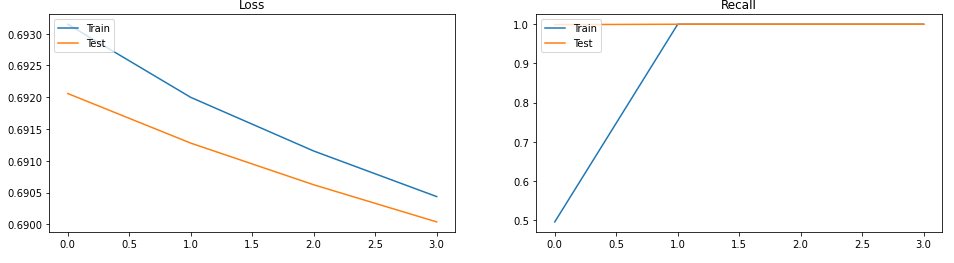


Figure 12

#### 100K Movielens data

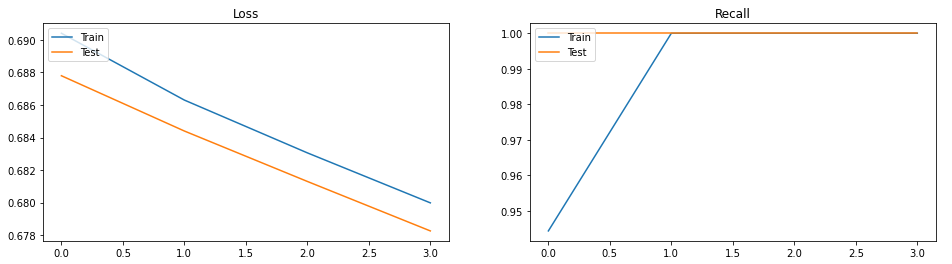


Figure 13

#### Learning from sets data

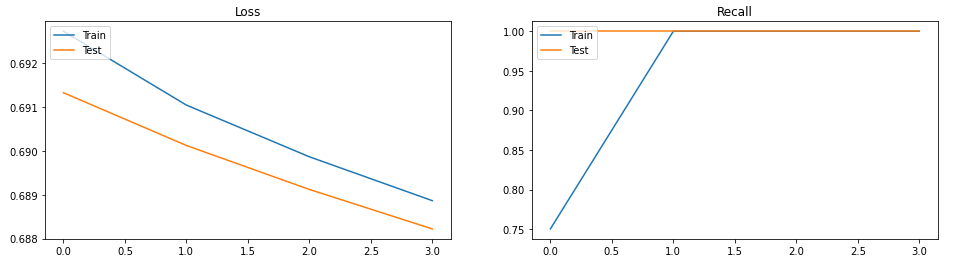


Figure 14