A Survey of Deep Learning-Based Movie Recommendation Systems

Group #22: Chinmay Bhale, Truxten Cook, Kritshekhar Jha,
Kumarage Tharindu Kumarage, Kyle Otstot, Paras Sheth

School of Computing, Informatics, and Decision Systems Engineering, ASU
{cvbhale, tcook11, kjha9, kskumar, kotstot, psheth5}@asu.edu

[1]

***Abstract*—Recommender systems have proven to be a successful method of reducing information overload in light of the increasing volume of online information. In recent years, research on recommender systems and information retrieval has shown that deep learning has a wide-ranging impact. However, they still have issues when working with extremely sparse data or other problems like cold start. The field of deep learning in recommender systems is currently flourishing. The purpose of this review paper is to provide a comprehensive and systematic analysis of current research projects on recommender systems based on deep learning. We also provide a detailed taxonomy along with summaries for state-of-the-art algorithms providing a perspective on the future trends and research challenges of deep learning recommender systems.**

***Index Terms*—Recommender systems, Deep Learning models, survey**

I. Introduction

The rapid expansion of information has greatly impacted how individuals make decisions. Consequently, recommender systems [1] have drawn researchers' interest as a practical solution to the problem of "information overload." Recommender systems have been extensively employed in various domains and problem statements, including big data analysis [2], and medicine prescription [3]. Interest in this field is still high due to the rising need for real-world applications that can offer individualized recommendations and manage information overload. Addressing these issues, numerous cutting-edge strategies have been proposed, including content-based collaborative filtering [4], clustering-based filtering, merging item- and user-based similarity [5], and so on. To forecast a user's rating on an item, collaborative filtering (CF) takes into account the ratings of related users or things. On the other hand, information retrieval and information filtering are the foundations of content-based filtering– an algorithm that compares the information in the product with the user profile, using both the profile and product as its sources of data [4]. The current state-of-the-art (SoTA) recommender systems are very successful and yield high performance; however, these systems still suffer from critical issues.

Deep recommendation algorithms have a natural appetite for data; however, a considerable amount of training data is required to benefit from the deep architecture. Data acquisition in recommender systems is expensive since tailored recommendations rely on user-generated data, yet most people often can only consume a small fraction of innumerable items [6]. As a result, deep recommendation models are unable to perform to their full potential due to the data sparsity problem [**?**]. Self-supervised learning's (SSL) main goal is to extract useful and transferable knowledge from large amounts of unlabeled data via well-designed pretext tasks (also known as self-supervised tasks), where the supervision signals are generated semi-automatically. Early versions of self-supervised recommendation (SSR) might be linked to unsupervised techniques such as autoencoder-based recommendation models [7], [8], which rely on various corrupted data to rebuild the initial data to prevent overfitting. Another common issue, known as the cold start problem, refers to providing recommendations for users and items with no historical interaction records [9].

When it comes to implicit data collection, recommender procedure entails automatically determining the degree of things or products by recording, evaluating, and processing data gleaned from users' application-related actions. For instance, implicit feedback of user information is utilized in movies to track user behavior. As it does not require explicit (clear) ratings to receive recommendations, using this kind (implicit) when looking for material on the web improves the user experience.[10] Contrarily, explicit feedback is a method that enables a consumer to clearly convey their demand for a product. Users generally give these items points by scanning them, such as a 5-star (or occasionally 10-star) rating system, a yes/no question, or perhaps likes/dislikes, which are popular in social networks, to express their interest in a product. Typically, recommender systems gather user preferences using a few of the rating systems. As a result, it could be argued

[1] Project Code

that this type of feedback (explicit feedback) depends primarily on how users react to an item after rating it.

As an accurate predictor of offline evaluation, proper recommender system evaluation is an issue that is becoming more and more crucial [11]. The recsys group changed to using Top-N techniques for offline evaluation because using root mean squared error on top of a projected rating matrix can be quite deceptive. [12]. For each user in the Top-N recommendation scenario, the system recommends the N most pertinent items. The is frequently the case in domains like media, news, or e-commerce. For example, collaborative filtering with matrix factorization has been suggested as one method for solving this task. Sparse data encoders have recently received a lot of attention and have produced cutting edge outcomes in this research problem. Models such as denoising autoencoders, variational autoencoders [13] [14], and a shallow autoencoder called the EASE [15], which, despite being a straightforward linear model, is producing results that are competitive and understandable while addressing the main issue with sparse autoencoders: overfitting towards identity.

A. Problem Statement

Movie recommendation is a vast field of importance with a long history of different proposed solutions. In this paper, we aim to provide an overview of the the current work on movie recommendation by evaluating models across the major common paradigms: supervised learning, self-supervised learning, semi-weak supervised learning, and unsupervised learning. This survey intends to provide readers interested in systems with a general framework for selecting deep neural networks to handle specific recommendation tasks. The three main contributions of this survey are as follows: (1) we conduct an extensive review of recommendation models based on deep learning techniques and propose a classification scheme to position and organize the current work; (2) we provide an overview and summary for SoTA methodologies; and (3) we discuss the challenges and open issues, as well as identify new trends and future directions in this research field.

II. Related Works

In this section, we outline the standard and SoTA methods and algorithms used for the task of content recommendation. Figure 1 provides a detailed view of the recommender system taxonomy focused on the model-based recommender systems.

Content-Based: this method first builds a profile of the user's preferences based on the movies they have already seen, and for the movies they have not yet seen, it generates the top K movie recommendations. To represent the movie characteristics and user profiles in vector form, techniques like vector space models (VSM) [16], term-frequency inverse-document-frequency (TF-IDF) models [4], and latent semantic index (LSI) [17] are used. The content-based filtering technique can be broken down into the following steps: product representation, profile learning, and recommendation generation.

Collaborative Filtering (CF): these systems gather user feedback in the form of ratings for items in a specific domain, allowing them to recommend subsequent items based on trends in rating behavior across users. That is, CF-based recommender systems will suggest items to a user based on the preferences of other users. On the one hand, *memory-based* systems continuously derive their recommendations from incoming user data; as the number of user ratings accumulates over time, so does the system's accuracy. On the other hand, *model-based* techniques can forecast future user behavior by learning a model from current user behavior [18].

Hybrid: hybrid recommendation systems are combinations of content-based and CF-based methods. In general, the combination of two approaches into a single model [19] provides a thorough assessment of recommender systems. There are seven primary ways for creating a hybrid model, as listed by Isinkaye et al. [20]: weighted, switching, mixed, feature combination, feature augmentation, cascade and meta-level. Focusing on model-based recommender system, we can further classify these based upon their learning paradigm, which is discussed below.

A. Supervised Learning

A graph neural network (GNN) is applied to the complete network in Graph Autoencoder (GAE) [21], which is the foundation of most GNN-based matrix completion techniques, to learn a representation for each node. To forecast potential ratings, the representations of the user and item nodes are combined. A multi-graph CNN model, for instance, is suggested by Monti et al. [22] to extract user and item latent information from their nearest neighbor networks. AutoRec [23] a collaborative filtering autoencoder-based model with explicit ratings that surpasses all available baseline algorithms. Following the development of variational autoencoders (VAE), Liang et al. [14] introduced the collaborative filtering model MultVAE employing multinomial log-likelihood as data distribution. In place of deep architectures, H. Steck presented EASE [15], an embarrassingly shallow autoencoder with no hidden layers outperforming SOTA models. Many straightforward, shallow and intricate deep learning architectures has been put forth during the development of the recommender system.

B. Self-Supervised Learning

The early prototypes of self-supervised recommendation (SSR) can be traced back to unsupervised methods

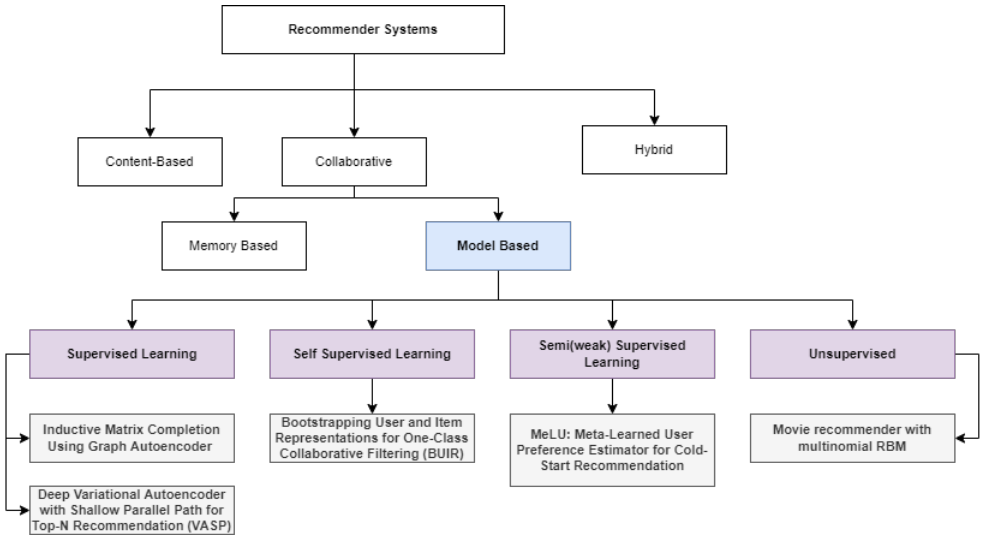


Fig. 1: Taxonomy of Recommender Systems

like autoencoder-based recommendation models which rely on different corrupted data to reconstruct the original input to avoid overfitting. Following that, SSR emerged as network embedding-based recommendation models [24], [25], in which random-walk proximity is used as self-supervision signals to capture similarities between users and items. During the same time period, some generative adversarial networks (GANs)-based recommendation models [26], [27] that augment user-item interactions can be viewed as another embodiment of SSR. The recommendation community then began to embrace SSL, and subsequent research [28], [29] focused on pre-training recommendation models with Cloze-like tasks on sequential data. For a more detailed analysis users can refer to [30]

C. Semi/Weak Supervised Learning

The primary motivation for incorporating semi-supervision in recommender systems was to address two known issues in supervised-learning techniques; (1) the data sparsity problem and (2) the cold-start problem– completely labeled data is unavailable for newly joining users [31]. Under the paradigm of SSML, we see that few-shot learning and meta-learning have been heavily utilized in recommender system applications such as movie recommendation [32]. Many previous works involving meta-learning in recommender systems tackle the cold start problem [33], [34]. Vartak et al. [33] present a meta-learning strategy to address item cold-start when new items arrive continuously. In other recommender system work, Bharadhwaj [34], and Lee et al. [35] use the model agnostic meta-learning framework to consider each user in the system as a task; the author then attempts to optimize each task's weights in a few gradient steps.

D. Unsupervised Learning

Unsupervised learning has a long history in for the Movie Recommendation task. These methods include things like K-Means Clustering and Principal Component Analysis, which both attempt to cluster similiar movies together to provide recommendations. Other types of unsupervised techniques that have been more model based have prominence in recent years such as AutoEncoders [36] and Generative Adversarial Networks (GANS). These models have benefited from the large amount of data that has become available for the movie recommendation task.

III. System Architecture & Algorithms

We tested several different models based on the categories above. Below we outline the architectures and the algorithms of each model we tested, along with their visualizations. We also talk about some of the salient features of each model and how they stand out and differentiate themselves from others.

A. Supervised Learning

For the learning paradigm supervised learning, we choose the following two architectures;

(1) Deep Variational Autoencoder with Shallow Parallel Path for Top-N Recommendation (VASP):

The goal of this paper is to develop an recommender system model that is as elegant and understandable as EASE while utilizing the potential of deep autoencoders to describe intricate nonlinear patterns in the data with most obvious hurdle of overfitting to identify. In order

to accomplish this Dropout in the input layer has historically been used to combat overfitting towards identity [13], [14], [37] though this strategy, however, falls short in terms of effectiveness and does not allow for the use of extremely deep architectures. The major highlights from this paper are as follows

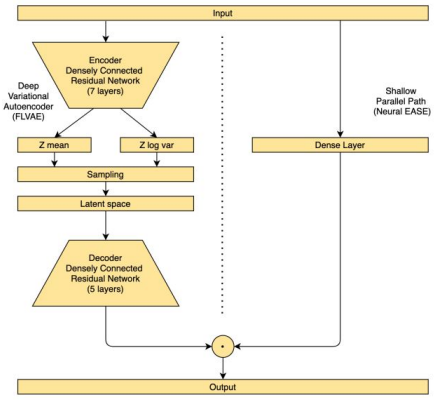


Fig. 2: Architecture of VASP

- The usage of focal loss for training autoencoders for TopN recommendation. Focal Loss (FL) is used for recommendation systems. The imbalance between the background class and other classes is addressed by this unique solution for object detection's unbalanced classes. This indicates that examples in the training set that are challenging to classify suffer greater losses. This is an excellent example of how collaborative filtering works in situations when certain items are more popular than others. Since there are fewer interactions with specialized products, recommending them is more challenging. The recommender algorithm is forced to concentrate more on niche and cold start products due to these items' higher loss rates.
- A simple yet effective data augmentation technique to prevent Top-N recommending autoencoders from overfitting towards identity. Preventing overfitting towards learning the identity function between the input and output layer of the autoencoder is another crucial concept when training an autoencoder on sparse data [38].
- A joint-learning technique based on the Hadamard product for training different combinations of various models. Proposed VASP architecture uses combination of two models, NEASE and FLVAE ensembled by element-wise multiplication.

(2) Inductive Matrix Completion Using Graph Autoencoder: This model[39] uses a Graph Autoencoder (GAE) to learn both user and item specific representations for personalized recommendations and local graph patterns for inductive matrix completions. The paper claims that the model is more efficient at learning local graph patterns in GAE and has good scalability with superior expressiveness compared to other GNN-based matrix completion methods. The model aims to make propose the following improvements:

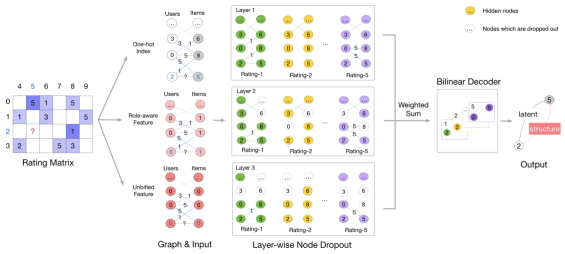


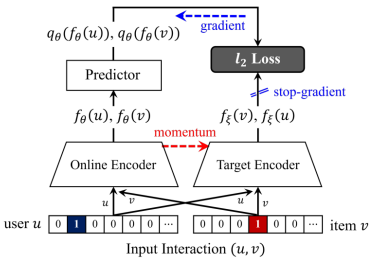
Fig. 3: Architecture of IMC-GAE

- The paper wants to better understand local graph patterns and to that end does quantitative analysis on 5 datasets. The observations from this analysis motivated the creation of the architecture for this model.
- The paper designs an identital feature and a role aware feature for the model to learn expressive graph patterns.
- The paper designs a layer-wise node dropout schema that drops more nodes in the higher levels. This makes link representation in the model contain more node information in a 1-hop local graph around the target link. This makes the model learn local graph patterns related to the target link, forwarding the capability of inductive learning.

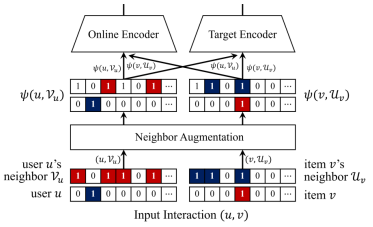
B. Self-Supervised Learning

For the learning paradigm self-supervised learning, we choose the following architecture:

Bootstrapping User and Item Representations for One-Class Collaborative Filtering (BUIR): The goal of this paper is to improve the One Class Collaborative Filtering (OCCF) which aims to identify the user-item pairs that are positively-related but have not been interacted yet, where only a small portion of positive user-item interactions (e.g., users' implicit feedback) are observed. The authors propose a novel framework called BUIR for which no negative sampling is necessary. BUIR uses two separate encoder networks that learn from one another to avoid a collapsed solution and make the representations of positively-related users and items similar to one another. Additionally, BUIR effectively resolves the OCCF data sparsity problem by adding stochastic data to the encoder inputs. Every time it encodes, BUIR randomly creates the augmented views of each successful interaction based on the user and item's neighborhood information. This self-supervision



(a) BUIR's framework



(b) BUIR's Data Augmentation

Fig. 4: The overall architecture of BUIR

then further trains the model. The architecture can be seen in Figure. 4. BUIR works as follows:

- BUIR use a network that is similar to a student-teacher in which only the student's output u (and v) is optimized to forecast the target v (and u) supplied by the teacher. In particular, BUIR bootstraps the representations of people and things directly by using two different encoder networks, referred to as the target encoder and the online encoder. The item (and user) vectors produced by the target encoder are more similar to those of the online encoder thanks to optimization. Simultaneously, the target encoder is updated using a momentum-based moving average to gradually approach the online encoder, encouraging the use of improved representations as the target for the online encoder. Thus, the positive correlation between u and v can be captured by the online encoder.
- To deal with the data-sparsity problem, BUIR employs a stochastic data augmentation technique. It exploits augmented views of an input interaction, which are generated based on the neighborhood information of each user and item (i.e., the set of the items interacted with a user, and the users interacted with an item). The stochastic augmentation is applied to positive user-item pairs when they are passed to the encoder, so as to produce the different views of the pairs. In the end, BUIR is allowed to learn various views of each positive user-item pair.

C. Semi/Weak Supervised Learning

For the learning paradigm semi-supervised learning, we choose the following architecture:

Meta-Learned User Preference Estimator for Cold-Start Recommendation (MeLU)

This study [35] suggests a recommender system that can predict user preferences based on just a few items. They mainly try to address the drawback of insufficient evidence candidate generation for the cold-start setting. It consists of a user preference estimator that utilizes model agnostic meta-learning theory to quickly adopt a new *task* with a few examples. Here the *task* is equivalent to estimating the item preference of a given user.

Architectures-wise MeLU consists of an ordinary neural matrix factorization machine. However, it's process differs from an ordinary neural matrix factorization due to the following two key components. **1)** MeLU incorporates supplementary information about the user (gender, age, ...) and item (title, genres, ...) while learning the user/item embeddings and, **2)** It implements the MAML theory by dividing the training into two-levels (learning-to-learn concept); namely local update and global update. For these two updates, they split a given user's consumed items into two sets: support and query. This training framework is as seen in figure 5.

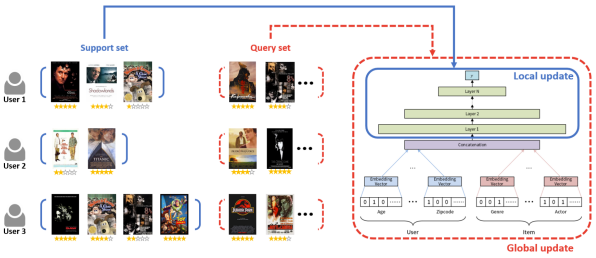


Fig. 5: Architecture of MeLU

D. Unsupervised Learning

For our Unsupervised Learning model, we choose to use a multinomial Restricted Boltzmann Machine [40]. A Restricted Boltzmann Machine is a type of a generative stochastic two layer neural network. They work by attempting to learn the probability distribution that the given dataset was sampled from. Once this probability distribution has been approximated, it can be sampled from to generative new rankings for movies that did not appear in the input dataset. The first layer, the visible layer, corresponds to visible features in the dataset. The second layer, the hidden layer, attempts to model the latent variables that affect the underlying probability distribution. Each hidden node in the Restricted Boltzmann Machine is connected to every input node. There are no connections between nodes in a layer in

a Restricted Boltzmann machine. We can consider this model as using a collaborative filtering based approach, as user data and movie data are used simultaneously to create the estimated probability distribution.

In general, Restricted Boltzmann Machines tend to perform well when the the conditions that the inference data were collected are similar to the conditions that the training data were collected with. This is because the Restricted Boltzmann Machine tries to learn the underlying probability distribution that the training data was sampled from, so if conditions change that affect the actual probability distribution that the real data emerges from, than the results of the Restricted Boltzmann Machine may degrade. An advantage of this model for movie recommendation is that the simplicity of the network allows for very fast training and inference.

IV. Datasets

Description: For this project, we will primarily work with one of the prominent movie recommendation benchmark datasets: the *MovieLens* dataset [41]. This dataset was extracted from the movie recommendation service MovieLens, containing the rating values for different movies and free-text labeling activities. The MovieLens dataset has different versions that offer additional data based on the use case. We primarily use MovieLens 1M and MovieLens 20M versions in our work. Table I summarizes the statistics of the two versions we used.

Version	Users	Movies	Ratings (range)
1M	6040	4000	1M (0.5 -5)
20M	138493	62000	20M (0.5 -5)

TABLE I: Dataset Description

Preproessing: The MovieLens dataset was preprocessed according to the different requirements of the above-mentioned selected SOTA methods. In particular, some arcitectures work with explicit ratings while others work with implicit ratings. One notable example was the training of the VASP model; the dataset was preprocessed in accordance with [14]. The data was changed to implicit interactions by taking into account legitimate interactions only rating of four or higher as this dataset contains explicit ratings. Moreover, for the MeLU approach, we had to separately create warm (users/items with enough interactions more than 5) and cold states (users/items with less than 5 interactions) training data files. However, to make a fair comparative study, we hold out an evaluation set and share it across all the model evaluations.

V. Evaluations

A. Evaluation Metrics

To determine the caliber of the approaches, methodologies, and algorithms for predictions and recommendations in recommender systems' research, assessment metrics and quality measurements are needed. Evaluation frameworks and metrics make it easier to compare several approaches to the same issue and choose from a variety of fruitful lines of inquiry that provide superior outcomes. The following are the most popular quality metrics: (1) Prediction assessments, (2) recommendations evaluations as sets, and (3) recommendations evaluations as ranked lists.

For the purpose of our survey, we utilize Mean-Absolute Error (MAE), and Root-Mean Sqaured Error (RMSE) for prediction assessments, Recall for recommendations evaluations as sets, and Hit-Ratio and Normalized Discounted Cumulative Gain (NDCG) for recommendation evaluations as ranked lists.

1) Prediction Assessment: The calculation of some of the most used prediction error metrics, among which the Mean Absolute Error (MAE) and its related metrics Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) stand out, is typically used to assess the accuracy of the findings of a recommender system. These metrics are formulated as,

$$\text{MAE} = \frac{1}{|U|} \sum_{u \in |U|} (\frac{1}{|I|} \sum_{i \in |I|} |p_{u,i} - r_{u,i}| \tag{1}$$

$$\text{RMSE } = \frac{1}{|U|} \sum_{u \in |U|} \sqrt{\frac{1}{|I|} \sum_{i \in |I|} (p_{u,i} - r_{u,i})^2} \tag{2}$$

where $|U|$ represents the number of users, $|I|$ represents the number of items, $p_{u,i}$ represents the predicted rating for user u and item i, and $r_{u,i}$ represents the true rating for user u and item i.

2) Recommendation Evaluations as Sets: Users' trust in a particular recommender system is not directly correlated with the collection of possible forecasts' accuracy. When a user accepts a smaller selection of recommendations provided by the RS, the user gains confidence in the recommender system. In this part, we establish the Recall recommendation quality metric, which measures the proportion of recommended items that are relevant relative to the total number of relevant items.

$$\text{R} = \frac{1}{|U|} \sum_{u \in |U|} \frac{\#(i \in Z_u)}{\#(i \in Z_u) + \#(i \in Z_u^c)} \tag{3}$$

where Z_u represents the set of items recommended to user u by the system, θ represents the relevance rating threshold and $r_{u,i}$ represents the rating for item i given by user u. In the equation 3, the following condition

Learning Paradigm	Algorithm	RMSE	MAE	NDCG@10	HR@10
Supervised Learning	IMC-GAE	**0.83**	0.74	0.78	**0.69**
Semi-Supervised Learning	MeLU	0.85	0.77	**0.84**	-
Self-Supervised Learning	BUIR	-	-	0.11	0.25
Unsupervised Learning	Multinomial Restricted Boltzmann Machine	**0.83**	**0.54**	0.69	-

TABLE II: Results obtained across different metrics for different models for the MovieLens-1M dataset.

Learning Paradigm	Algorithm	NDCG@100	Recall@20	Recall@50
Supervised Learning	VASP	0.444	0.411	0.548

TABLE III: Results obtained across different metrics for VASP for the MovieLens-20M dataset.

should holds: $r_{u,i} \geq \theta$. Z_u^c represents the set of items the user interacted with but were not present in the recommendation set.

3) Recommendation Evaluations as Ranked Lists: Users give the first recommendations on the list more weight when there are a large number of recommended things. Compared to the last item on the list, the errors made in these items are more serious. The ranking metrics take into account this circumstance. Among the ranking measures most often used are Hit-Ratio@K and NDCG@K. Hit-Ratio@K refers to the proportion of users for which the model is able to correctly include the items user has interacted with in the list of top-K recommended items:

$$HR = \frac{|U_{\text{hit}}^K|}{|U_{\text{all}}|}, \tag{4}$$

where $\left|U_{\text{hit}}^K\right|$ is the number of users for which the recommender systems was able to include the items user has interacted with in the top-K recommended items list and $|U_{\text{all}}|$ denotes the total number of users in the test dataset. For user-item interactions, gain for an item refers to the relevance score. To take order of the ranking into account, Discounted Cumulative Gain (DCG) is formulated as $DCG_K^i = \sum_{r=1}^{K} \frac{2^{y_{u,r}-1}}{\log_2(1+r)}$, where $y_{u,r}$ refers to the ground truth rating for user u and r^{th} ranked item. The normalized discounted gain (nDCG) is

$$nDCG_K = \frac{1}{|U|} \sum_{u \in U} \frac{DCG_K^u}{IDCG_K^u}, \tag{5}$$

where $|U|$ refer to the total number of users in the test set and $IDCG_K^u$ refers to the best value of DCG_K^u.

B. Results and Findings

We ran the different learning paradigm based models and obtained the results as shown in the Table. II for MovieLens-1M Dataset, and Table. III for MovieLens-20M Dataset. We made the following observations regarding the different models:

- Both the Supervised Learning and Unsupervised Learning models tend to have lower prediction error when compared to the Semi-Supervised Learning models. The Self-Supervised Learning model, BUIR were not evaluated for error rates as they deal with implicit feedback.
- The Semi-Supervised Learning model MeLU, performs best in terms of the ranking metrics, as it utilizes auxiliary information about the users such as the user demographics. In terms of the ranking performance, supervised learning models outperform the unsupervised models.
- Explicit Feedback based models tend to have better performance for the MovieLens-1M dataset when compared to the implicit feedback model, BUIR.
- Models such as MeLU that leverage additional cues such as user demographics to better understand the user preferences, lead to better ranking performance, indicating that capturing user preferences is crucial in recommender systems.

VI. Visualization

In this section we describe the visualization results from the various model architectures.

A. Supervised Learning

(1) Deep Variational Autoencoder with Shallow Parallel Path for Top-N Recommendation (VASP):

The Figure: 6 shows the trend of the training and the validation loss over 40 epochs. The decrease in the training loss is expected but we also see the validation loss increasing which means the model is loosing its capability to generalize on unseen data.

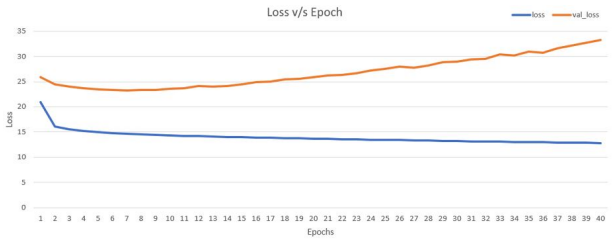


Fig. 6: Training and Validation loss per Epoch

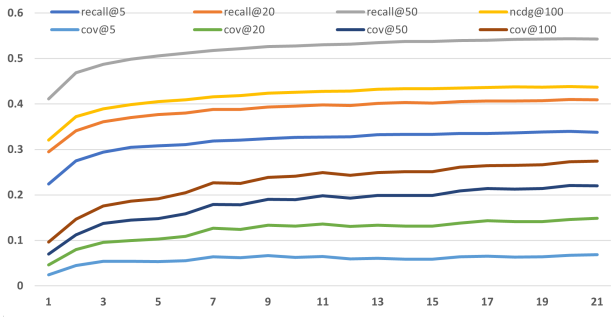


Fig. 7: Metric evaluation results per Epoch

We expect that if the model is able to generalize well the loss is decreasing and the accuracy is increasing but this is an ideal scenario. When we implement and reproduce the results from the VASP model we see in the Figure: 7 the coverage metric results after epoch 8 plateaus out and do not change significantly over further epochs. At this point the model start over fitting on the training data and causing the validation loss to increase as seen in the Figure: 6. We argue increasing loss and stable accuracy could also be caused by good prediction classified as a little worse. This is the point where the model start to overfit and both validation loss increases and stability in the accuracy happens simultaneously. This is when the network starts to learn the patterns only relevant to the training dataset and therefore lack in generalization. For the final evaluation we do 5-fold validation on the user's interaction history. It is more objective measure than only hide sampled 20% of the user's interactions randomly.5-fold mean NCDG@100: 0.444, Recall@20: 0.411 and Recall@50: 0.548

(2) Inductive Matrix Completion Using Graph Autoencoder:

Below 8 is the train and validation loss as the training progresses. The model trained for a total of 1578 epochs. There are no signs of over or underfitting of the model.

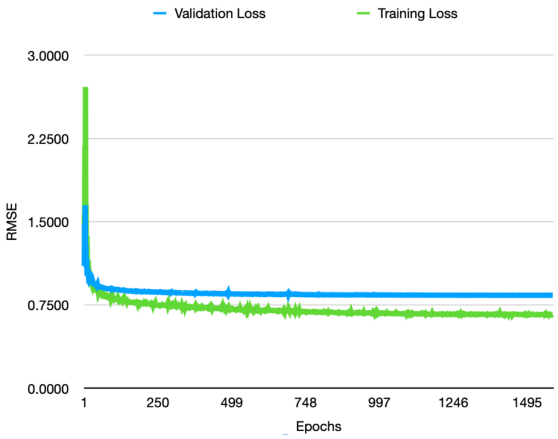


Fig. 8: Training and Validation loss per Epoch

B. Self-Supervised Learning

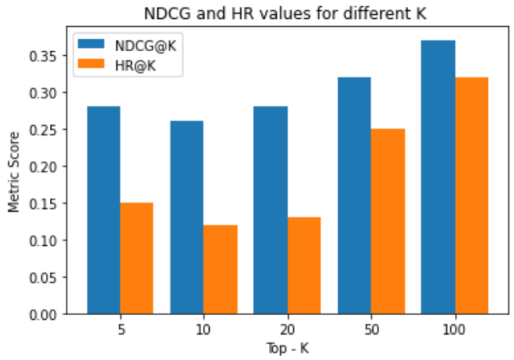


Fig. 9: Metric evaluation results per Epoch

Since BUIR is designed for implicit feedback, we aimed to evaluate its ranking performance over different values of K. The resulting figure can be seen in Figure. 11. For the ranking metrics' as K increases the Hit-Ratio and NDCG values should also increase. This happens because the larger the K is more likely the model is able to find a hit for the items in the relevant items set. We observe this happening as seen in the figure. As the number of K increases both the NDCG@K and HR@K increase. Finally, the increment is significant as the HR@5 starts at 0.14 and HR@100 is close to 0.30. Similarly, NDCG@5 increases from 0.29 to NDCG@100 0.37. The increment in NDCG is not as huge as HR because along with the presence of the item in the relavant list, NDCG also checks if the order of the item is preserved or not between the relevant item set and the recommended item set.

C. Semi/Weak Supervised Learning

As mentioned in the architecture section, MeLU is specially catered to solve the cold start problem while

making recommendations. Therefore, the authors define the following four states of recommendations to investigate this claim.

R1- Recommendation of existing items for existing users
R2- Recommendation of existing items for new users
R3- Recommendation of new items for existing users
R4- Recommendation of new items for new users

Here R2 represents the user cold start while R3 represents the item cold-start. And R4 is the most severe cold start case where we have both user and item cold starts. The following Figure 10 shows the rating prediction and ranking task performances of MeLU under each of these four states.

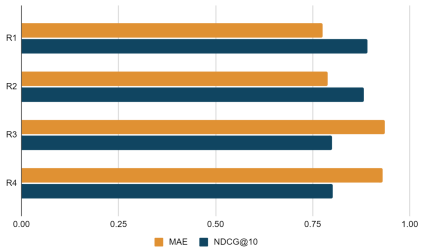


Fig. 10: Metric evaluation results per Epoch

We can observe that even though MeLU's rating prediction performance decreases (MAE) when the cold-start occurs, it still performs well in item ranking task (NDCG), i.e., user preference estimation. In a real-world recommender system, what is most important in the cold-start state is to have a good evidence candidate generator. Thus, from this visualization, we can see that MeLU is a worthy candidate model.

D. Unsupervised Learning

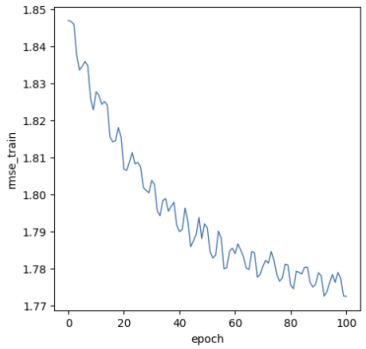


Fig. 11: RMSE over training epochs for the Mulitinominal Restricted Boltzmann Machine

Over the course of training the Restricted Boltzmann Machine, we find that the RMSE steadily decreases, plateuing around epoch 100. Going beyond epoch 100 for training did not significtly decrease the loss and decreases our other metrics, such as MAE and NDCG@10, so training was stopped at 100 epochs.

VII. Bonus: Movie & User Rankings

For the supplementary portion of our project, we were tasked with the proposal of a method that ranks top movies and expert users by genre. In doing so, we identified the characteristics of a "top movie" or "expert user", quantified them using standard metrics, and developed a strategy for synthesizing metric pairs into one representative measure; then a natural ordering emerges, which leads us to rankings of movies and users.

First, we brainstormed qualities of movies and users that would deem them popular and knowledgeable, respectively. For movies, a top-ranked one should not only achieve a high average rating but also receive a respectable amount of reviews; a steady balance between the two objectives would ensure that high-rated, hardly-watched movies and low-rated, commonly-watched movies would not outrank movies that are rated both highly and frequently. Similarly, we would expect an expert user to not only give ratings that are consistent with public opinion, but also rate a sufficiently large amount of movies; this way, we know the user has good judgment and possesses enough domain knowledge to give confident claims. Therefore, we decided to consider two metrics for each ranking; the movie ranking will consider the *number of reviews* and *average rating* for each movie, and the user ranking will consider the *number of reviews* and *average rating error* for each user. The last metric is computed as follows:

$$\text{Error}(u) = \frac{1}{|u\text{-Ratings}|} \sum_{m,r \in u\text{-Ratings}} |r - \text{AvgRating}(m)|$$

where u-Ratings is the set of movie (m) ratings (r) done by user u. Clearly, this creates a 2D attribute distribution for each of the rankings; however, in order to establish an ordering, we must project the 2D data onto 1D space. Dimensionality reduction techniques have received interest in many domains, but for our purposes, we consider Principal Component Analysis (PCA)– a method that aims to find a projection onto 1D space that maximizes the variance between the data points. This characteristic of PCA is particularly desirable in our application because a larger spread of data ensures greater ranking robustness to slight distributional changes (e.g., an incoming batch of new ratings). Specifically, Figures 12 and 13 showcase the results of PCA performed on the metric pairs described for each of the two rankings, filtered by the *comedy* genre; the projections are illustrated for a select group of ranks (e.g., 1st, 2nd, 5th, 10th, 20th), which gives better insight into how the algorithm balances the two metrics.

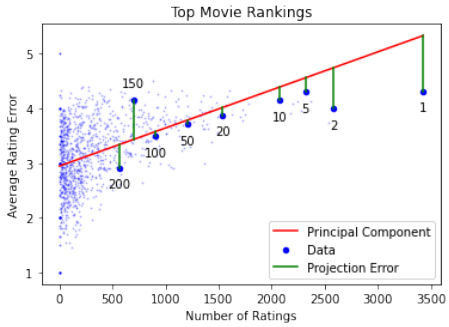


Fig. 12: Top movies ranked by PCA

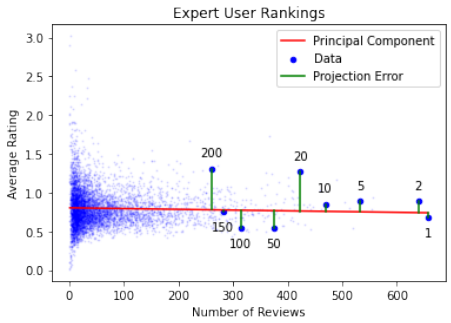


Fig. 13: Expert users ranked by PCA

Although the preliminary application of PCA gives intriguing results, one cannot help but question if a line is the best 1D space to project the data onto; after all, data for both cases appear to follow the direction of a curve rather than a line. As a result, we also consider the implementation of PCA on a *transformed* space, one that converts the values in each dimension to percentiles of their respective distributions; this way, the 2D is uncluttered and unbiased, encouraging a line to be a more appealing candidate for projection. After careful transformation of the axes (see `notebook.ipynb` in the `bonus_section` directory of the repository), we perform PCA in the percentile space to find the line of best fit, then transform everything back to the original space. In Figures 14 and 15, we see that the principal component is now a curve instead of a line, and appears to be much more representative of the data. For comparison, in the repository notebook we listed the top 10 movies determined by both types of PCA implementations, and the results do differ in some of the positions. Specifically, the latter implementation places more of an emphasis on average rating when the movies possess a sufficiently-high number of reviews. Overall, the second attempt qualitatively appears to identify a better set of comedy movies, but both results are generally consistent with public opinion.

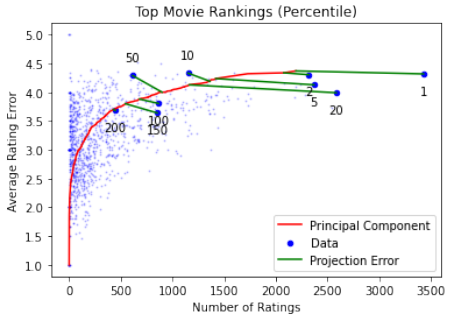


Fig. 14: Top movies ranked by PCA in percentile space

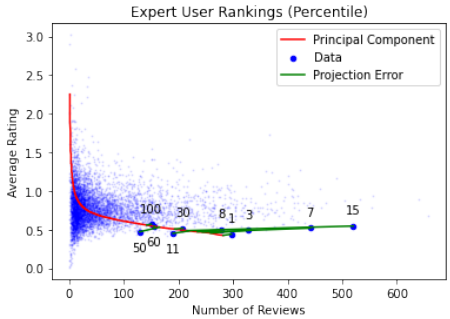


Fig. 15: Expert users ranked by PCA in percentile space

VIII. Conclusion

Recommending pertinent content to users, RSs have been utilized to address the issues associated with information overload. To obtain high-quality RSs, numerous developments and breakthroughs have been made in this area. The key tasks of deep learning-based recommender systems are organizing the enormous amounts of heterogeneous data from several sources, creating better user models based on user requirements and preferences, and enhancing performance. In comparison to traditional recommender systems, deep learning-based recommender systems can model the sequence patterns of user behavior, more accurately reflect the user's various preferences, and increase recommendation accuracy by using deep learning techniques to automatically learn the latent features of user and item by integrating various types of multi-source heterogeneous data. This study focuses on the most prevalent challenges and issues, including cold start issues, sparsity issues, scalability issues, and evaluation issues, as well as the most current attempts to address them.

REFERENCES

[1] J. Dong *et al.*, "A recommendation system based on multi-attribute," in *2016 ICSS*. IEEE.

[2] X. Zhang *et al.*, "Lshiforest: A generic framework for fast tree isolation based ensemble anomaly analysis," in *2017 IEEE ICDE*. IEEE, 2017.

[3] M. Wang *et al.*, "Safe medicine recommendation via medical knowledge graph embedding," *arXiv preprint arXiv:1710.05980*, 2017.

[4] Y.-L. Chen *et al.*, "A movie recommendation method based on users' positive and negative profiles," *Information Processing & Management*, 2021.

[5] A. Y. Mohamad *et al.*, "Collaborative filtering approach for movie recommendations," in *ECTI-CON*. IEEE, 2022.

[6] B. M. Sarwar, *Sparsity, scalability, and distribution in recommender systems*. University of Minnesota, 2001.

[7] Y. Wu *et al.*, "Collaborative denoising auto-encoders for top-n recommender systems," in *ACM WSDM*, 2016.

[8] S. Li *et al.*, "Deep collaborative filtering via marginalized denoising auto-encoder," in *CIKM*, 2015.

[9] J. Liu *et al.*, "A survey on heterogeneous information network based recommender systems: Concepts, methods, applications and resources," *AI Open*, 2022.

[10] J. Chen *et al.*, "An implicit information based movie recommendation strategy," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2018.

[11] T. Rehorek *et al.*, "Comparing offline and online evaluation results of recommender systems," in *RecSyS 2018*, 2018.

[12] G. Karypis, "Evaluation of item-based top-n recommendation algorithms," in *CIKM*, 2001.

[13] D. Kim *et al.*, "Enhancing vaes for collaborative filtering: flexible priors & gating mechanisms," in *ACM RecSys*, 2019.

[14] D. Liang *et al.*, "Variational autoencoders for collaborative filtering," in *WWW*, 2018.

[15] H. Steck, "Embarrassingly shallow autoencoders for sparse data," in *WWW*, 2019.

[16] B. Walek *et al.*, "A hybrid recommender system for recommending relevant movies using an expert system," *Expert Systems with Applications*, 2020.

[17] K. Bougiatiotis *et al.*, "Enhanced movie content similarity based on textual, auditory and visual information," *Expert Systems with Applications*, 2018.

[18] B. Bhatt *et al.*, "A review paper on machine learning based recommendation system 1," 2014.

[19] J. Lu *et al.*, "Recommender system application developments: a survey," *Decision Support Systems*, 2015.

[20] F. O. Isinkaye *et al.*, "Recommendation systems: Principles, methods and evaluation," *Egyptian informatics journal*, 2015.

[21] T. N. Kipf *et al.*, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[22] F. Monti *et al.*, "Geometric matrix completion with recurrent multi-graph neural networks," *NeurIPS*, 2017.

[23] J. P. Zhou *et al.*, "Noise contrastive estimation for autoencoding-based one-class collaborative filtering," *arXiv preprint arXiv:2008.01246*, 2020.

[24] M. Gao *et al.*, "Bine: Bipartite network embedding," in *ACM SIGIR*, 2018.

[25] C. Zhang *et al.*, "User collaborative network embedding for social recommender systems," in *SIAM ICDM*, 2017.

[26] Q. Wang *et al.*, "Enhancing collaborative filtering with generative augmentation," in *ACM SIGKDD*, 2019.

[27] J. Yu *et al.*, "Enhance social recommendation with adversarial graph convolutional networks," *IEEE TKDE*, 2020.

[28] F. Sun *et al.*, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *ACM CIKM*, 2019.

[29] X. Chen *et al.*, "Bert4sessrec: Content-based video relevance prediction with bidirectional encoder representations from transformer," in *ACM MM*, 2019.

[30] J. Yu *et al.*, "Self-supervised learning for recommender systems: A survey," *arXiv preprint arXiv:2203.15876*, 2022.

[31] A. H. Khan *et al.*, "A survey of recommender systems based on semi-supervised learning," in *ICICC*. Springer, 2022.

[32] Z. Du *et al.*, "Sequential scenario-specific meta learner for online recommendation," in *ACM SIGKDD*, 2019.

[33] M. Vartak *et al.*, "A meta-learning perspective on cold-start recommendations for items," in *NeurIPS*, 2017.

[34] H. Bharadhwaj, "Meta-learning for user cold-start recommendation," in *IJCNN*. IEEE, 2019.

[35] H. Lee *et al.*, "Melu: Meta-learned user preference estimator for cold-start recommendation," in *ACM SIGKDD*, 2019.

[36] P. Srikanth *et al.*, "Movie recommendation system using deep autoencoder," in *ICECA*, 2021.

[37] I. Shenbin *et al.*, "Recvae: A new variational autoencoder for top-n recommendations with implicit feedback," in *ACM WSDM*, 2020.

[38] H. Steck, "Autoencoders that don't overfit towards the identity," *NeurIPS*, 2020.

[39] W. Shen *et al.*, "Inductive matrix completion using graph autoencoder." arXiv, 2021. [Online]. Available: https://arxiv.org/abs/2108.11124

[40] R. Salakhutdinov *et al.*, "Restricted boltzmann machines for collaborative filtering," ser. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007. [Online]. Available: https://doi.org/10.1145/1273496.1273596

[41] F. M. Harper *et al.*, "The movielens datasets: History and context," vol. 5, no. 4, 2015. [Online]. Available: https://doi-org.ezproxy1.lib.asu.edu/10.1145/2827872