DEEP LEARNING FOR BOOK RECOMMENDATION

Tianyi Wang
PB21151800
wtyyy@mail.ustc.edu.cn

Guangyu Li
PB21081615
flipped@mail.ustc.edu.cn

Ming Yan
PB21020688
ym689@mail.ustc.edu.cn

November 1, 2024

ABSTRACT

This report investigates the application of advanced deep learning techniques to enhance book recommendation systems. We explore three main methods: Neural Collaborative Filtering (NCF), Light Graph Convolutional Networks (LightGCN), and Self-supervised Graph Learning (SGL). Each technique is analyzed for its effectiveness in addressing common challenges faced by traditional recommender systems, such as sparsity of data and the cold start problem. Our experimental results, derived from the Goodbooks-10k dataset, demonstrate that these methods not only improve recommendation accuracy but also offer scalability and efficiency in processing large datasets. LightGCN, in particular, shows significant promise due to its simplified computational model, which maintains high performance while reducing resource consumption. The report further discusses the implications of these findings for future research and practical applications in the field of personalized book recommendations.

***K*eywords** Deep Learning · Recommendation

1 Introduction

In the era of digital libraries and online bookstores, personalized book recommendations have become essential for enhancing user engagement and satisfaction. Traditional recommendation systems, primarily grounded in the collaborative filtering paradigm, often face challenges like sparsity of data and the inability to accommodate nuanced user preferences. This report delves into advanced deep learning techniques that aim to address these shortcomings and enhance recommendation accuracy. Specifically, our exploration encompasses a range of groundbreaking models including Singular Value Decomposition (SVD), Matrix Factorization (MF), Neural Collaborative Filtering (NCF), SASRec, Light Graph Convolutional Networks (LightGCN), SimGCN, and Self-supervised Graph Learning (SGL), each offering a unique approach to modeling complex user-item interactions.

Our study systematically tests these pivotal developments in recommendation technology. We start by revisiting earlier breakthroughs such as SVD and MF, and progressively move towards more recent innovations like NCF, which integrates deep neural networks into collaborative filtering to capture subtle patterns in user data. Following this, sequence modeling techniques like SASRec, along with graph-based methods such as LightGCN and SimGCN, are evaluated to understand their efficiency in leveraging the structure of user-item interactions. These methods represent the forefront of research in recommendation systems and are pivotal for advancing the field.

For those seeking state-of-the-art (SOTA) performance, the convergence of sequence modeling and graph-based approaches, particularly through models like SASRec and LightGCN, suggests a promising direction. Our comprehensive testing of these models not only benchmarks their effectiveness in a controlled environment using the Goodbooks-10k dataset but also highlights their practical applicability in real-world scenarios where the quality of recommendations can significantly impact user experience.

Through this investigation, we aim to contribute to the ongoing evolution of recommendation systems, offering insights that could inform future enhancements in the field of recommender systems. Our goal is to provide a robust understanding of where current technology can meet user needs and how emerging models can potentially reshape the landscape of personalized recommendations.

This research report focuses on the application of NCF, LightGCN, SGL to book recommendation data. Section2,3,4 briefly introduces these three methods, Section5 introduces the dimensionality reduction method needed in the method, and Section 6 introduces the experiments of these three methods on the book recommendation data, including the training process of the model and the final evaluation of the model. Secondly, we summarize our experience and thinking. Finally, we describe one of our previous unsuccessful attempts to fine-tune the KDD grand model. At the end of the report, we attach the pseudo-code of the code implementation to help students who understand these methods for the first time to quickly get started with the three models.

[1]

2 Neural Collaborative Filtering

Neural Collaborative Filtering (NCF) is a recommendation algorithm that uses neural networks to model user-item interactions. It was proposed as an improved approach to collaborative filtering, which is a widely used technique for making personalized recommendations based on user preferences.

NCF combines the strengths of both collaborative filtering and neural networks by leveraging the non-linear modeling capabilities of neural networks to capture complex patterns in user-item interactions. It can effectively handle sparse and implicit feedback data, where user-item interactions are often missing or only partially observed.

The core idea behind NCF is to create a neural network architecture that can learn the user and item embeddings from the input data. These embeddings capture the latent factors that influence user preferences and item characteristics. The neural network then uses these embeddings to predict the user's preference for a particular item. Typically we employ a multi-layer perceptron (MLP) or a deep neural network (DNN) architecture to learn the user and item embeddings. The network takes the user and item features as input and passes them through several fully connected layers to generate the final prediction.

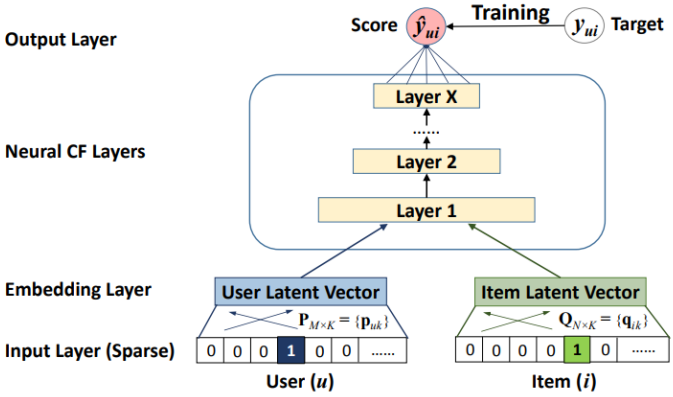


FIGURE 1: NCF

One advantage of NCF is that it can capture both the explicit and implicit feedback from users. Explicit feedback refers to explicit ratings or feedback given by users, while implicit feedback includes user interactions such as clicks, purchases, or browsing history. NCF can handle implicit feedback by treating it as binary feedback (e.g., whether a user has interacted with an item or not).

[1] Guangyu Li: Implemented the NCF model on the dataset, conducted data dimensionality reduction analysis, wrote the main body of the report, and created visualizations. (Note: Guangyu also processed the KDD dataset as part of an initial effort in fine-tuning a large-scale model for the KDD project, which was a moderately successful attempt but not the main focus of our major assignment.)

Ming Yan: Implemented five models on the dataset: lightgcn, SAS, sgl, simgcn, and MF. (Note: Ming conducted preliminary research and preparation for the KDD dataset and replicated fine-tuning of a large model for the KDD project.)

Tianyi Wang: Implemented the NMF, SVD, and SGL models on the dataset, wrote the main body of the report, created visualizations, and wrote pseudo code. (Note: Tianyi also processed the KDD dataset as part of the efforts related to the KDD project.)

3 LightGCN

LightGCN[2]is also a recommendation algorithm that focuses on the simplicity and efficiency of collaborative filtering-based recommendation systems. It was proposed as an improvement over more complex and computationally expensive graph-based methods like Graph Convolutional Networks (GCN). However, unlike traditional GCN, LightGCN does not use complex aggregation rules or weight parameters. Instead, it simply averages the embeddings of neighboring nodes to obtain the updated embeddings.

The key idea behind LightGCN is to simplify the graph convolution operation used in GCN by **removing the non-linear activation functions** and **the layer-wise weight parameters**. This simplification helps to reduce the model complexity and speed up the training and inference process.

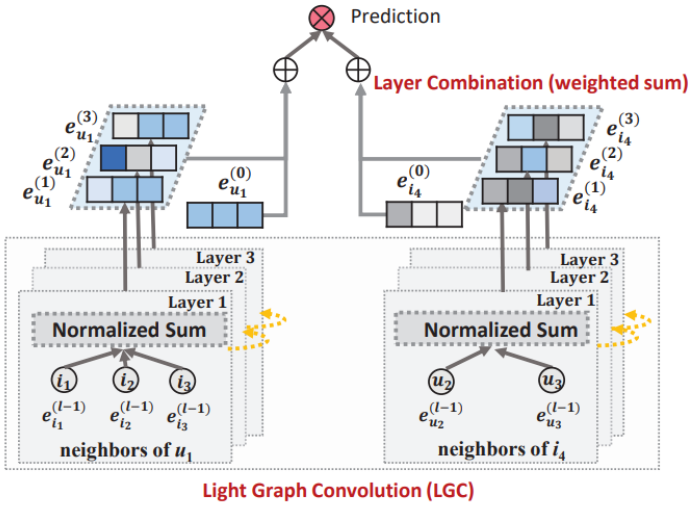


FIGURE 2: LightGCN

In LightGCN, the user-item interaction data is represented as a bipartite graph, where users and items are connected by edges indicating the interactions. The algorithm aims to learn low-dimensional embeddings for users and items that capture their latent features and preferences.

By simplifying the graph convolution operation, LightGCN achieves computational efficiency and scalability. It can handle large-scale recommendation tasks with millions of users and items without incurring a significant computational overhead.

One of the advantages of LightGCN is its simplicity. It has a straightforward architecture and does not rely on complex neural network layers or activation functions. This simplicity makes it easier to understand, implement, and interpret compared to more complex graph-based models.

Despite its simplicity, LightGCN has demonstrated competitive performance in recommendation tasks. It has been shown to achieve comparable or even superior recommendation accuracy compared to more complex methods like GCN, especially in scenarios with sparse and implicit feedback data.

4 Self-supervised Graph Learning

Self-supervised Graph Learning[4] is an approach that aims to learn meaningful representations of graph-structured data without relying on explicit supervision or labeled data. It leverages the inherent structure and patterns present in the graph to guide the learning process.

In contrast to the previous model, LightGCN, which is a collaborative filtering-based recommendation algorithm, self-supervised graph learning is a more general framework that can be applied to various tasks involving graph-structured data, including recommendation, node classification, link prediction, and graph clustering.

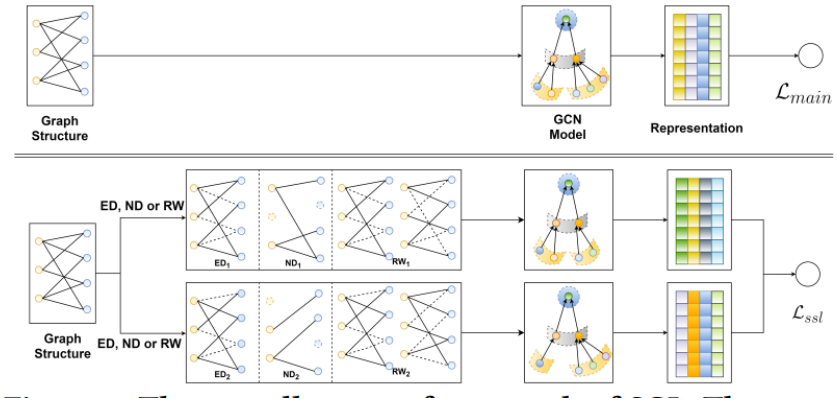


FIGURE 3: Self-supervised Graph Learning

The key difference lies in the learning paradigm and the type of information utilized. While LightGCN focuses on collaborative filtering and user-item interactions, self-supervised graph learning is concerned with capturing the underlying graph structure and features.

Self-supervised graph learning typically involves designing pretext tasks that encourage the model to learn meaningful representations of the graph. These pretext tasks are designed to exploit the graph's local or global structural properties. By solving these pretext tasks, the model is forced to learn to encode important graph information into its representations.(Node attribute prediction or Graph context prediction and so on)

The self-supervised learning process in graph learning involves training a model on the pretext tasks using unsupervised techniques such as contrastive learning, generative modeling, or graph autoencoders. Once the model is trained, the learned representations can be used for downstream tasks such as recommendation, node classification, or link prediction.

5 Dimensionality Reduction Methods

Problems such as sparse data samples and difficult distance calculation in high-dimensional situations are serious obstacles faced by all machine learning methods, which are called "dimensionality disasters". [1]An important way to alleviate the dimensionality disaster is dimensionality reduction, that is, to transform the original high-dimensional attribute space into a low-dimensional "subspace" through some mathematical transformation, in which the sample density is greatly increased, and the distance calculation also becomes easier.

Since the dataset we chose has 10000 dimensions, we applied dimensionality reduction techniques to our dataset for calculation and visualization easily.

5.1 Principal component analysis(PCA)

Principal component analysis is one of the most commonly used methods for dimensionality reduction. The main idea of PCA is to map n-dimensional features to k(< n) dimension, which is a new orthogonal feature, also known as principal components, which is a reconstructed k-dimensional feature on the basis of the original n-dimensional feature. We expect it to have the following characteristics:

- **Re-proximal reconstruction**: The sample points are close enough to the hyperplane.
- **Maximum separability**: The specimen points are projected in this hyperplane as far apart as possible.

PCA only needs to retain the mean vector of W and the sample mean, and the new sample can be projected into the low-dimensional space by simple vector subtraction and matrix multiplication. We discard some of the information, but it is often necessary for the sake of dimensionality reduction and noise reduction;

Algorithm 1 Principal Component Analysis (PCA)

Input: Sample set $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$;
Number of principal components: k
Output: Projection matrix $\mathcal{W}$ and Sample Mean $\overline{\boldsymbol{x}}$
Centralization:
$\overline{\boldsymbol{x}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{x}_i$
for $i = 1$ to m **do**
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \overline{\boldsymbol{x}}$
end for
Compute Covariance Matrix:
$\boldsymbol{S} \leftarrow \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{x}_i \boldsymbol{x}_i^{\top}$
Eigenvalue Decomposition:
Compute eigenvalues and eigenvectors of $\boldsymbol{S}$
Sort eigenvectors by decreasing eigenvalues
Take the eigenvectors corresponding to the largest k eigenvalues:
$\mathcal{W} \leftarrow [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k]$
return $\mathcal{W}, \overline{\boldsymbol{x}}$

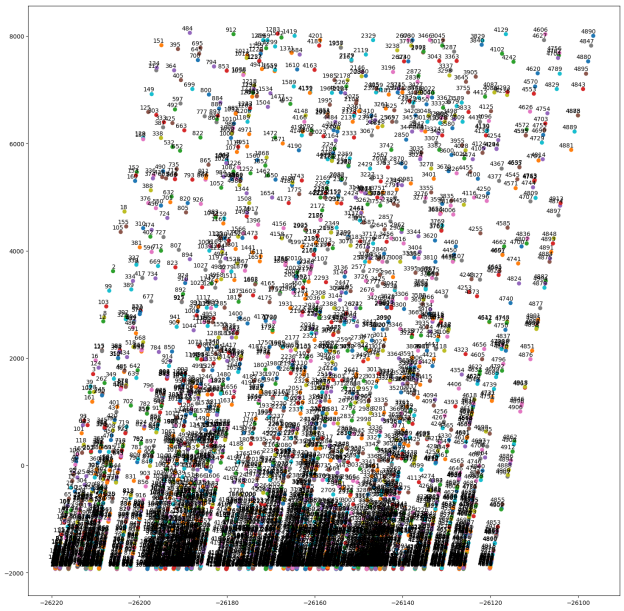


FIGURE 4: PCA

5.2 Mainfold Learning

Manifold learning is a class of dimensionality reduction methods that draw on the concept of topological manifolds. Manifold learning assumes that the data is uniformly sampled in a high-dimensional Euclidean space in a low-dimensional manifold structure, and it aims to recover the low-dimensional manifold structure from the high-dimensional sampled data, that is, to find the low-dimensional manifold in the high-dimensional space, and find the corresponding embedding map to achieve dimension reduction or data visualization.

Manifold learning techniques, such as t-SNE, enable effective visualization of high-dimensional data, revealing underlying patterns and structures. [3] They preserve local relationships between nearby data points, accurately representing the intrinsic structure of the data. Also these methods are robust to noise and outliers, focusing on capturing underlying structure rather than irrelevant features.

We expect it to have the following characteristics:

- **Local preservation**: The local relationships between nearby data points are preserved in the low-dimensional embedding.

- **Non-linear mapping**: Manifold learning techniques aim to capture and represent non-linear structures in the data.
- **Dimensionality reduction**: Manifold learning methods reduce the dimensionality of the data while preserving its essential structure.
- **Topology preservation**: The topological structure of the data, such as clusters and boundaries, is preserved in the low-dimensional embedding.
- **Intrinsic geometry**: Manifold learning techniques aim to capture the intrinsic geometry of the data, revealing its underlying organization.
- **Non-Euclidean distances**: Manifold learning methods often utilize non-Euclidean distance measures to capture the complex relationships between data points.
- **Robustness to noise**: Manifold learning algorithms are designed to be robust to noise and outliers in the data.
- **Visualization**: Manifold learning techniques are often used for visualizing high-dimensional data in a lower-dimensional space.

Algorithm 2 Manifold Learning

Input: Sample set $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$; The number of low-dimensional spatial dimensions: d'
Output: Low-dimensional embedding $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m\}$
for $j = 1, 2, \ldots, m$ **do**
    Preprocessing step for data point $\boldsymbol{x}_j$;
end for
Compute pairwise distances or affinity matrix $\boldsymbol{S}$ based on D;
Construct the graph or neighborhood structure based on $\boldsymbol{S}$;
Perform dimensionality reduction to obtain a low-dimensional embedding $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m\}$;
return $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m\}$;

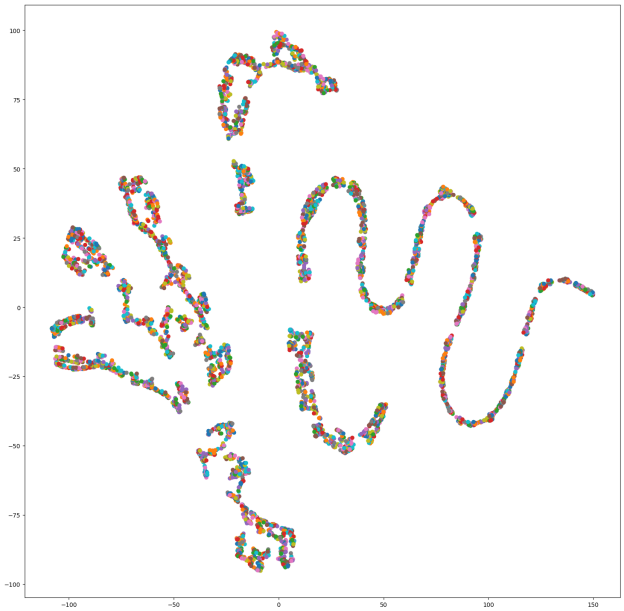


FIGURE 5: T-SNE On 5k points

Obviously, these data are implicitly categorized, and it would be helpful to know the names of these class-specific data to help us make book recommendations.

At first, we decide to use the data whose dimension has been reduced to neural networks, but it seems that nothing happened except the speed could be faster. To avoid the information loss, we don't take the methods and it is just used for visualization.

6 Application

6.1 Dataset

The Goodbooks-10k dataset is derived from the publicly available dataset from Goodreads, a website that provides book recommendations and social features for readers. It consists of approximately 6,000,000 ratings for 10,000 books. The dataset has been processed into an implicit feedback dataset, which is widely used in recommender systems.

The goal of the Goodbooks-10k dataset is to predict the next potential interaction for users by analyzing their behavioral patterns and rating data. This dataset enables personalized book recommendations, helping users discover books of interest and enhancing their reading experience.

By utilizing the Goodbooks-10k dataset, researchers and developers can explore and experiment with various recommendation algorithms to provide more accurate and personalized book recommendations. Its openness and accessibility also foster research and collaboration in the field of recommender systems.

6.2 Experiment

6.2.1 Definitions of Measurement

Firstly, before the experiment part, we introduce all the evaluation metrics used in this paper.

- **Accuracy (ACC):** Accuracy (ACC) is defined as the best match between the truth and the predicted labels.

$$ACC = \sum_{m} \frac{\mathbf{1}\{y_i = m(i)\}}{m} \tag{1}$$

  where y represents the ground truth labels, $m(i)$ represents the labels, and m enumerates mappings between clusters and labels.

- **F1-Score:**

$$F1 = \frac{2 * P * R}{P + R} \tag{2}$$

  where P represents the precision $P = \frac{TP}{TP+FP}$, R represents the recall $R = \frac{TP}{TP+FN}$.

- **Recall**: Recall measures the fraction of relevant items that have been retrieved over the total amount of relevant items.

$$\text{Recall} = \frac{\text{Number of relevant items recommended}}{\text{Total number of relevant items}} \tag{3}$$

- **NDCG (Normalized Discounted Cumulative Gain)**: NDCG accounts for the position of the recommended items, giving higher importance to items at the top of the recommendation list. It is calculated by dividing the DCG (Discounted Cumulative Gain) of the recommendations by the ideal DCG (IDCG):

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}} = \frac{\sum_{i=1}^{p} \frac{2^{rel_i}-1}{\log_2(i+1)}}{\max\left(\sum_{i=1}^{p} \frac{2^{rel_i}-1}{\log_2(i+1)}\right)} \tag{4}$$

- **MAP (Mean Average Precision)**: MAP calculates the average precision at different recall levels for each query and averages these values over all queries. It effectively captures the quality across recall levels:

$$\text{MAP} = \frac{1}{Q} \sum_{q=1}^{Q} \left(\frac{1}{m_q} \sum_{k=1}^{m_q} \text{Precision@k} \right) \tag{5}$$

  where m_q is the number of relevant items for query q.

- **MRR (Mean Reciprocal Rank)**: MRR calculates the average of the reciprocal ranks of results for a set of queries. The reciprocal rank is the inverse of the rank at which the first relevant item is found:

$$\text{MRR} = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{\text{rank}_q} \tag{6}$$

  where rank_q is the rank position of the first relevant item for query q.

6.2.2 NCF

There are results from the NCF under 30 epochs.

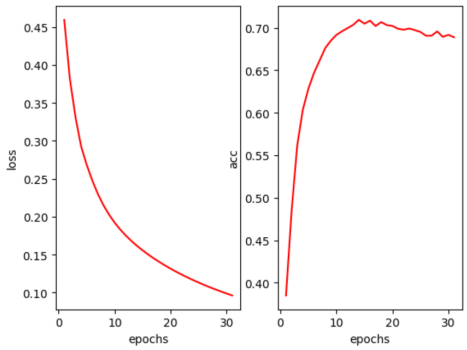
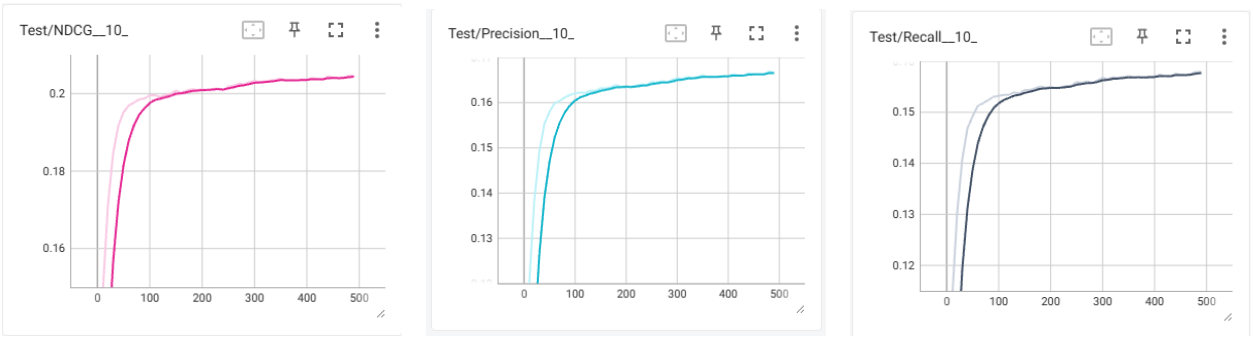


FIGURE 6: NCF under 30 epochs

From 15 epochs, the loss still decrease but the accuracy is not improve. So we take early stop. And we will take it as **baseline**.

6.2.3 Light GCN

There are results from the LightGCN under 500 epochs.



(A) NDCG@10 (B) Precision@10 (C) Recall@10

FIGURE 7: Prediction Results On Goodbook by LightGCN

And the corresponding loss curve is as below.

6.2.4 Self-supervised Graph Learning(SGL)

We take SGL for 50 epoch and 1000 epoch. In the experiment with the SGL model, I monitored the performance of the model during training by calculating the Precision, Recall, NDCG, MAP, MRR on the validation set. These five metrics have defined as above.

And the change curve of these five indicators is shown in the following figure.

Note that after training about 70 epochs, the five indicators of the model tend to be stable, which means it may cost only 70-100 epochs to get a well-trained SGL model.

6.2.5 Results Overview

We combine all results from above.

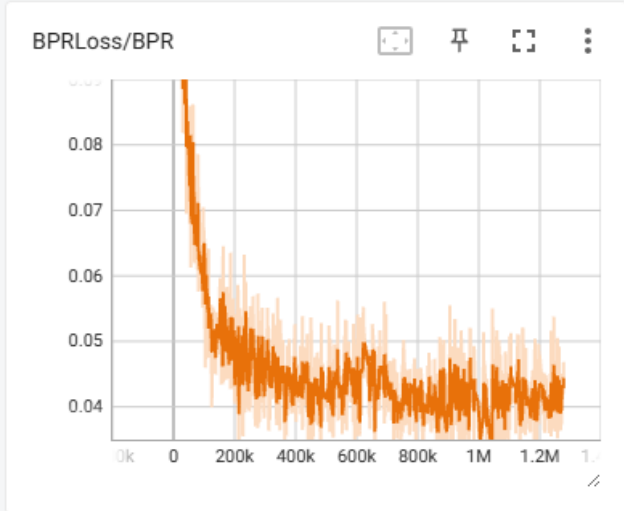


FIGURE 8: BPRLoss

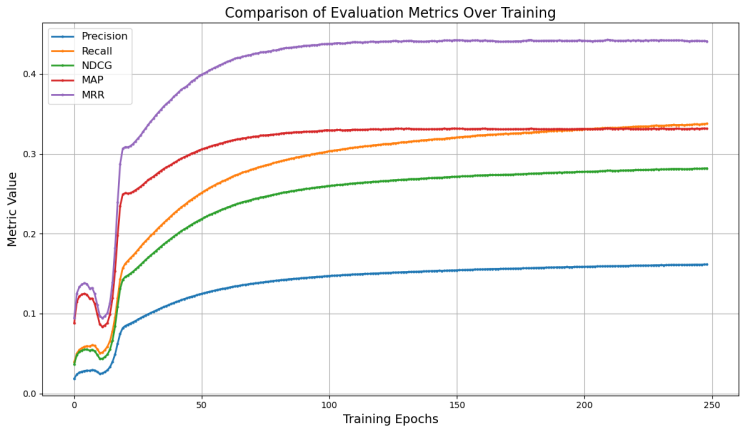


FIGURE 9: SGL Output

Table 1: Results of Goodbook dataset in different method

Methods	F1-Score
Single Value Decomposition(SVD)	0.00378
Matrix Factorization(MF)	0.00282
Nonnegative Matrix Factorization(NMF)	0.00103
Neural Collaborative Filtering(NCF)	0.00532
Simple Graph Contrastive Learning (SimGCN)	0.00198
LightGCN	0.00885
Self-supervised Graph Learning(SGL)	**0.03028**
Self-Attentive Sequential Recom.(SAS)	0.00511

Tips: Here we emit the introduction for SimGCN[5]. It simplifies some argument methods in GCN. More details could look for the paper.

Of the above methods, the first three are classical matrix-based recommendation methods originating from the turn of the century; the fourth attempts to incorporate neural networks based on matrix decomposition with the intention of

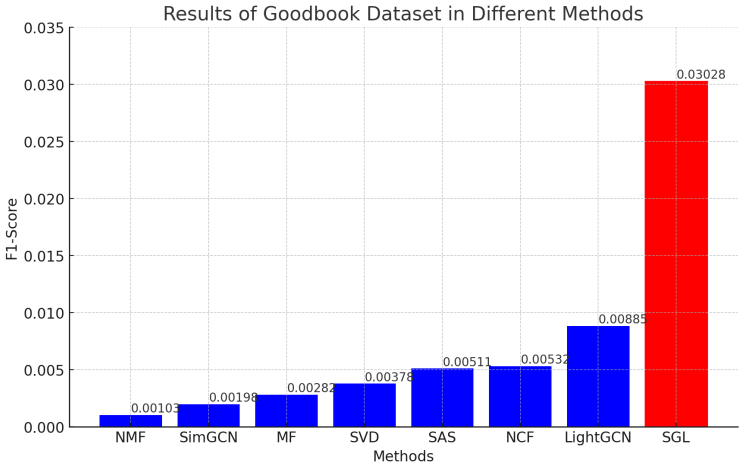


FIGURE 10: Barplot of the Results

exploiting their feature-capturing properties. In the middle are neural networks based on graph structures, which include a simplification of GCN: LightGCN, throwing away the activation function, self-loop, and other components in graph neural networks, and deriving a number of variant algorithms on its basis, which include graph recommendation with self-supervised learning(SGL), weakened data augmentation algorithms in graphs(SimGCN), and so on. In addition to this, there is the method SASRec that incorporates sequence modeling, which is another class of recommendation algorithms parallel to the graph methods mentioned above.

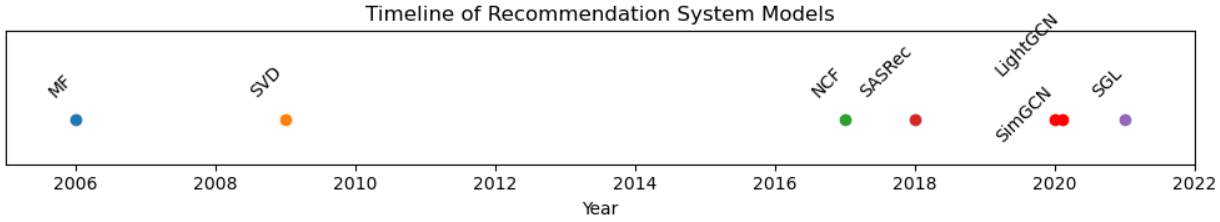


FIGURE 11: Timeline of Recommendation Methods

7 Conclusion

This research report has extensively evaluated the application of three advanced deep learning techniques—Neural Collaborative Filtering (NCF), Light Graph Convolutional Networks (LightGCN), and Self-supervised Graph Learning (SGL)—on the Goodbooks-10k dataset. Our experiments demonstrate that these methodologies significantly enhance the performance of book recommendation systems, addressing both the accuracy and efficiency challenges posed by traditional models.

NCF has shown its strength in harnessing deep neural networks to capture complex user-item interactions, while LightGCN streamlined graph convolutional operations, achieving high accuracy with reduced computational overhead. SGL, by leveraging unsupervised learning, effectively utilized data structure and patterns without explicit labels, proving to be exceptionally effective in scenarios characterized by sparse data.

The evaluation metrics used, including accuracy, F1-score, recall, NDCG, MAP, and MRR, have provided a comprehensive view of each model's capabilities, confirming that the integration of deep learning into recommendation systems offers a substantial improvement over classical methods. Notably, SGL outperformed other techniques in several metrics, highlighting its potential as a leading approach for future recommendation systems.

Moving forward, we recommend further exploration into the integration of sequence modeling and graph-based techniques, as evidenced by the promising results of LightGCN and SGL. Additionally, the application of these methods in other domains, such as music or video content, could be investigated to validate their versatility and effectiveness in different recommendation scenarios.

Ultimately, this report contributes to the ongoing advancement of recommendation technologies, providing a robust framework for future research and development. Our findings underscore the importance of continual innovation in this field, aiming to deliver more personalized, accurate, and user-centric recommendation systems that cater to diverse consumer needs.

8 Some Thinking

When we tried the classic matrix and graph-based recommendation algorithm, our classmates told us that there are some recommendation algorithms that are more suitable for this dataset, such as the sequence-based SASRec, which can achieve an F1 score of 0.1 or higher, so we also tried that approach, but due to the lack of time or the hyper-parameters were not adjusted in place, our SAS didn't show superior results, and this is the first time that we tried this approach. We have to try it in the following study.

9 Amazon KDD Cup 2024 Challenge

We participation in the KDD Cup 2024, focusing on "User Behavior Alignment." We investigated dependency-based multi-action patterns in users for recommendation systems, exploring traditional methods like ESMM and ESM2, and newer approaches integrating large language models like MoELoRA and LoRAMoE. They chose to model using MoE+LoRA but faced challenges due to limited time, computational resources, and inexperience with git for managing large files.

During the competition, they reproduced the official baseline and chose the Phi-3-mini-4k-instruct for fine-tuning, using a dataset selected from Amazon-M2 for a multiple-choice task. Fine-tuning with LoRA improved the model's performance, particularly on multiple-choice scores.

Overall, despite not achieving all their goals, the experience was enriching. They gained insights from extensive literature reviews and learned the practical aspects of training large models. The process improved their research skills and understanding of current model training methodologies.

Following images are images showing the model's performance scores before and after training, indicating improvements in various tasks, especially in the multiple-choice and retrieval scores.

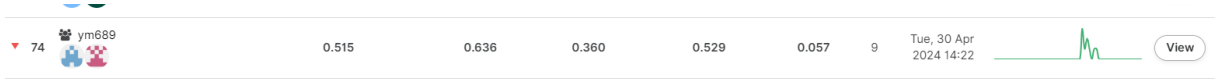


FIGURE 12: KDD Result

References

[1] V. de Silva and J.B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, volume 15, pages 721–728, Cambridge, MA, USA, 2003. The MIT Press.

[2] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation, 2020.

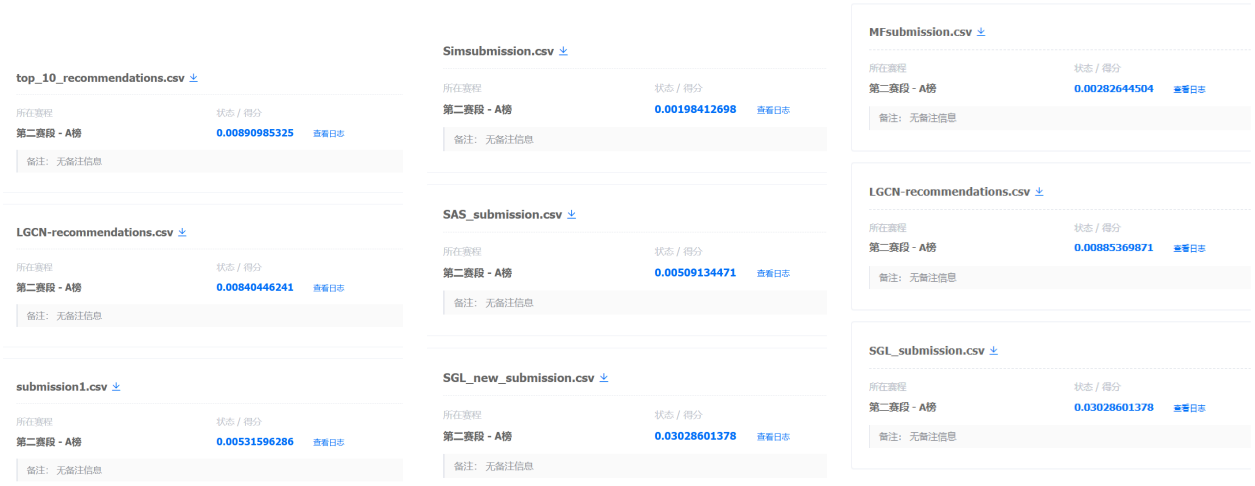
[3] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[4] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21. ACM, July 2021.

[5] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary? simple graph contrastivenbsp;learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 1294–1303, New York, NY, USA, 2022. Association for Computing Machinery.

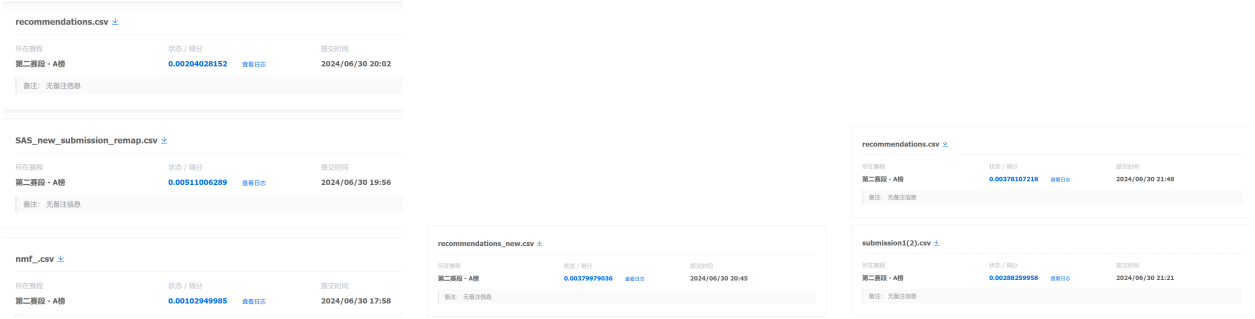
A Website Result

Here are the real snapshots.



(A) Website 1 (B) Website 2 (C) Website 3

FIGURE 13: Screenshot



(A) Website 4 (B) Website 5 (C) Website 6

FIGURE 14: Screenshot

B Some Algorithms

Algorithm 3 Neural Collaborative Filtering (NCF) for Recommendations

Input: User set U, Item set I, interactions R
Output: Prediction scores $\hat{y}$ for user-item pairs
function EMBEDDINGLAYERS(U, I)
    $P \leftarrow$ User embeddings initialized randomly
    $Q \leftarrow$ Item embeddings initialized randomly
    return P, Q
end function
function GMF(p_u, q_i)
    return $p_u \odot q_i$ ▷ Element-wise product
end function
function MLP(p_u, q_i)
    $z_0 \leftarrow$ concatenate(p_u, q_i)
    for $l \leftarrow 1$ **to** L **do**
        $z_l \leftarrow \sigma(W_l z_{l-1} + b_l)$ ▷ σ is activation function
    end for
    return z_L
end function
function NEUMF(p_u, q_i)
    $x_{\text{gmf}} \leftarrow$ GMF(p_u, q_i)
    $x_{\text{mlp}} \leftarrow$ MLP(p_u, q_i)
    $x \leftarrow$ concatenate$(x_{\text{gmf}}, x_{\text{mlp}})$
    $\hat{y} \leftarrow \sigma(Wx + b)$ ▷ Final prediction layer
    return $\hat{y}$
end function
$P, Q \leftarrow$ EMBEDDINGLAYERS(U, I)
for each user $u \in U$ and item $i \in I$ **do**
    $p_u \leftarrow P[u]$, $q_i \leftarrow Q[i]$
    $\hat{y}_{ui} \leftarrow$ NEUMF(p_u, q_i)
    Store $\hat{y}_{ui}$ for recommendations
end for

Algorithm 4 Light Graph Convolutional Network (LightGCN) for Recommendations

Input: User-item bipartite graph $G(V, E)$, adjacency matrix A
Output: User and item embeddings for recommendation
function NORMALIZEADJACENCY(A)
    $D \leftarrow$ diagonal matrix with $D_{ii} = \sum_j A_{ij}$
    return $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
end function
function LIGHTGCNLAYER($H^{(l)}, \tilde{A}$)
    return $\tilde{A} H^{(l)}$
end function
$\tilde{A} \leftarrow$ NORMALIZEADJACENCY(A)
Initialize embeddings $H^{(0)}$ for all nodes
for $l \leftarrow 1$ to L **do**
    $H^{(l)} \leftarrow$ LIGHTGCNLAYER$(H^{(l-1)}, \tilde{A})$
end for
Final embeddings $Z \leftarrow \frac{1}{L+1} \sum_{l=0}^{L} H^{(l)}$
return Z

Algorithm 5 Self-supervised Graph Learning (SGL) for Recommendations

Input: User-item interaction graph $G(V, E)$
Output: Enhanced node representations
function NODEDROPOUT(G, p_{nd})
    for each node v in G **do**
        Draw $r \sim \text{Uniform}(0, 1)$
        if $r < p_{nd}$ **then**
            Remove node v and its associated edges from G
        end if
    end for
    return G
end function
function EDGEDROPOUT(G, p_{ed})
    for each edge (u, v) in G **do**
        Draw $r \sim \text{Uniform}(0, 1)$
        if $r < p_{ed}$ **then**
            Remove edge (u, v) from G
        end if
    end for
    return G
end function
function RANDOMWALK(G, length)
    $G_{\text{new}} \leftarrow$ empty graph
    for each node u in G **do**
        $v \leftarrow u$
        for $i \leftarrow 1$ to length **do**
            $v \leftarrow$ RandomNeighbor(v)
            Add edge (u, v) to G_{new}
        end for
    end for
    return G_{new}
end function
function SGCLAYER(H, G)
    $H' \leftarrow$ Zero matrix of same shape as H
    for each node u in G **do**
        $N_u \leftarrow$ Neighbors of u
        $H'_u \leftarrow \text{Aggregate}(H_{N_u})$
    end for
    return H'
end function
$G_{\text{augmented}} \leftarrow$ NODEDROPOUT(G, p_{nd})
$G_{\text{augmented}} \leftarrow$ EDGEDROPOUT($G_{\text{augmented}}, p_{ed}$)
$G_{\text{augmented}} \leftarrow$ RANDOMWALK($G_{\text{augmented}}$, length)
Initialize node features $H^{(0)}$
for $l \leftarrow 1$ to L **do**
    $H^{(l)} \leftarrow$ SGCLAYER($H^{(l-1)}, G_{\text{augmented}}$)
end for
return $H^{(L)}$