Fusion of collaborative and multimodal embeddings for recommendation using LightGCN

Project for the exam of Semantics in Intelligent Information Access

Francesco Musci

A.A. 2024/2025

Contents

List of Tables

List of Figures

1 Introduction

The amount of information available on the Internet is enormous; therefore, systems are needed to sift through all this information and provide the user with information most relevant to them. A particular type of such systems are recommender systems. These take into account the user's history of interaction with some items (e.g. products on an e-commerce site, movies in a movie ratings site, etc...) and then suggest the next item the user may be interested in.

One of the most basic systems for recommendation is collaborative filtering[1]. This method takes into account exclusively the interactions between the users and the items. It exploits the fact that users which have interacted with the same items in the past have similar tastes and therefore are likely to interact with the same new items in the future. This approach, in its simplicity, provides a very effective baseline, since, even without explicit information, it is able to capture latent feature of users and items.

Graph Convolutional Networks[2] are convolutional networks which operate on graphs. These work by updating a node's embedding by aggregating the embeddings of its neighbors through a specific number of layers. For recommendation, these are exploited by creating a bipartite user-item graph and performing graph convolution on it. LightGCN[3] is a GCN in which most of the more complex operations are removed from the GCN process, as they are deemed not useful, or even harmful, for the task of recommendation.

Multimodal recommender systems are a type of recommender system which integrates information from different modalities about the items[4] (e.g. image, audio, video, etc...). These have been shown to be effective when integrated with collaborative filtering, especially in cases in which the collaborative information may be insufficient.

The aim of this project is to study how to fuse collaborative and multimodal information for recommendation using LightGCN. LightGCN's collaborative embeddings are fused with the multimodal embeddings and the resulting item embeddings are used for recommendation.

The data and implementation are available at `https://github.com/muscif/neo-mmlgcn`.

2 Related work

2.1 Collaborative filtering

Collaborative Filtering (CF) is a recommendation technique based on the user's past interactions and the interactions of similar users. CF tries to predict the future behavior of the user based on their past behavior and the behavior of users similar to them. Two users are more similar the more they have interacted with the same items. This is a purely collaborative approach, as this takes into account only the user-item interaction matrix and no information about either the users or the items.

A key statistic in datasets for recommendation is sparsity, i.e. how many interactions are there between users and items; technically, it is the proportion of the entries in the user-item interaction matrix that are zero. This is linked to the cold-start problem: when a new user or a new item is added to the system, it has no, or very few, interactions, so the recommender system has no way to know what to recommend to the user.

There are many architectures for CF, ranging from simple Matrix Factorization[5] (MF) to more complex ones like Graph Neural Networks[6] (GNNs).

2.1.1 Matrix Factorization (MF)

There are many specific architectures for MF; the general procedure is the following. Given a user-item interaction matrix the model learns two lower-dimensional matrices: a user matrix and an item matrix. Each row represents a user or an item and the columns represent the latent factors. These factors represent hidden (or latent) features. The effect of MF is to cluster together users with the same features and items with the same features. Therefore, for recommendation, the matrices are multiplied to obtain a reconstructed complete user-item matrix, which should capture new user-item interactions. The way the decomposition acts depends on the loss function.

2.1.2 Graph Neural Networks (GNNs)

As the name suggests, GNNs are neural networks which operate on graphs. The way these operate is through a mechanism called "message passing", which updates a node's embedding by aggregating information from that node's neighbors. An issue which arises in these cases is over-smoothing, i.e. as k increases, each node's embedding is learned from an increasing number of neighbors, thus leading the nodes in the graph to be too similar to each other. Thus, while the choice of k is task-specific, performance begins to

degrade (or hit a ceiling) after layer 4. Interestingly, MF can be seen as a special case of GNN where $k = 0$.

There are many operations that can be performed with this architecture, such as node classification, graph classification, node clustering, link prediction, etc. In this case, the operation performed is link prediction, i.e. try to predict missing user-item links for recommendation.

There are various architectures for GNNs. One such architecture is that of Graph Convolutional Networks (GCNs), which are used in this project. GCNs use a message passing function which uses a layer-wise propagation rule by aggregating information from neighboring nodes.

2.2 Multimodal recommender systems

Collaborative filtering works exclusively by considering user-item interactions. It does not take into account any kind of information about either users or items. However, a lot of data is available for the items and it can be used to add information to the system and improve performance. For instance, for a movie, we could add modalities for image (e.g. the poster image), text (e.g. the plot description), audio (e.g. the main theme of the soundtrack), etc.

Multimodal information can be useful to tackle the cold-start problem, as it adds information that can compensate for the lack of collaborative information[4].

There are many ways to combine multimodal information with collaborative information. In a multimodal recommender system, first, in the raw feature extraction phase, modality-specific encoders extract the information from its raw representation and embed it. Then, the feature interaction phase follows, in which the collaborative and multimodal embeddings need to be fused. The fusion methods can range from simple concatenation[7] to more complex attention mechanisms[8].

2.2.1 MultiModal LightGCN (MMLGCN)

MMLGCN[9] is a multimodal recommender system which fuses the multimodal item embeddings with the collaborative item embeddings, in various ways. The collaborative embeddings are obtained using a GCN recommender system, LightGCN[3], which greatly simplifies traditional GCN-based recommender systems by removing feature transformations and nonlinear activations, demonstrating that they can be detrimental to the task.

Three systems are proposed: MultiModal LightGCN (MMLGCN), Late-Fusion MultiModal LightGCN (LF-MMLGCN) and Early-Fusion MultiModal

LightGCN (EF-MMLGCN).

The idea of MMLGCN is, given n modalities, to train n LightGCN models and, for each model, the item's embeddings are initialized with a multimodal embedding. After they have been trained, the learned item representations are fused together and thus the final embedding is obtained. Formally:

$$e_i = AGG(LGCN_1(e_i^{(0)}) \cdot W_1, \ldots, LGCN_n(e_n^{(0)}) \cdot W_n) \cdot W_{out} \tag{1}$$

where

- $e_i^{(0)}$ is the initial embedding of the i-th modality
- W_i is the projection matrix to project the embeddings to the same dimensionality
- W_{out} is the optional projection matrix for the final embeddings

The AGG function used is the concatenation.

2.2.2 LF-MMLGCN

The idea is to initialize LightGCN's embeddings with random initialization, instead of initializing them with the multimodal embeddings. Then, at each forward pass of the network, the multimodal embeddings are fused, using an aggregation function, with the collaborative embeddings. Formally:

$$e = AGG(LGCN(e^{(0)}, e_1^{(0)}W_1, \ldots, e_n^{(0)}W_n) \cdot W_{out} \tag{2}$$

where

- $e^{(0)}$ is the initial embedding of the LightGCN model
- $e_i^{(0)}$ is the initial embedding for the i-th modality
- W_{out} is the projection matrix for the LightGCN embeddings

The embeddings are frozen during training.

2.2.3 EF-MMLGCN

This method is analogous to LF-MMLGCN, but the multimodal embeddings are fused before training and not during the forward pass. During the forward pass the aggregated multimodal embeddings are fused with the collaborative embeddings. Formally:

$$e = [LGCN(e^{(0)}) \| AGG(e_1^{(0)}W_1, \ldots, e_n^{(0)}W_n) \cdot W_{agg}] \cdot W_{out} \tag{3}$$

where

- $e^{(0)}$ is the initial embedding of the LightGCN model
- $e_i^{(0)}$ is the initial embedding of the i-th modality
- W_i is the projection matrix for the i-th modality
- W_{agg} is the projection matrix for the aggregated embeddings
- W_{out} is the projection matrix for the final embeddings

The embeddings are frozen during training.

In addition the changes to the model architecture, two other changes have been made. The first is the usage of dropout. The second is the usage of another loss in addition to the BPR loss, the symmetric NCE loss. This loss enforces a better separation of the embeddings, making similar entities closer in the embedding space and pushing more dissimilar entites further apart.

3 Methodology

The system described in this work is based on MMLGCN. They share the same basic premise: fuse the collaborative LightGCN embeddings with the multimodal embeddings. The general procedure is to get the collaborative embeddings for the items, fuse them with the multimodal embeddings and then use the resulting embedding to perform the prediction (Figure 1).

Given an item i, let its collaborative embedding be e_i and its multimodal embeddings be $m^j \in M_i$, where M_i is the set of multimodal embeddings for the item i and m^j's are the individual multimodal embeddings.

First, the multimodal embeddings are normalized using $L2$ normalization. The multimodal embeddings are then projected, through a simple linear layer, to the same latent dimension d as the collaborative embeddings. Then, at each neighbor aggregation step:

1. Get LightGCN's item embeddings $e_i, \forall i$.
2. Get the multimodal embeddings $M_i, \forall i$.
3. Fuse the collaborative and multimodal embeddings.
4. Use the new augmented embeddings to perform the predictions.

The final prediction is computed with the inner product of the final item and user embeddings.

This project aims to answer the following research questions:

- **RQ1**: does multimodal information improve the performance of LightGCN?
- **RQ2**: which architecture provides the best results?
- **RQ3**: how does our methodology perform w.r.t. the baselines?

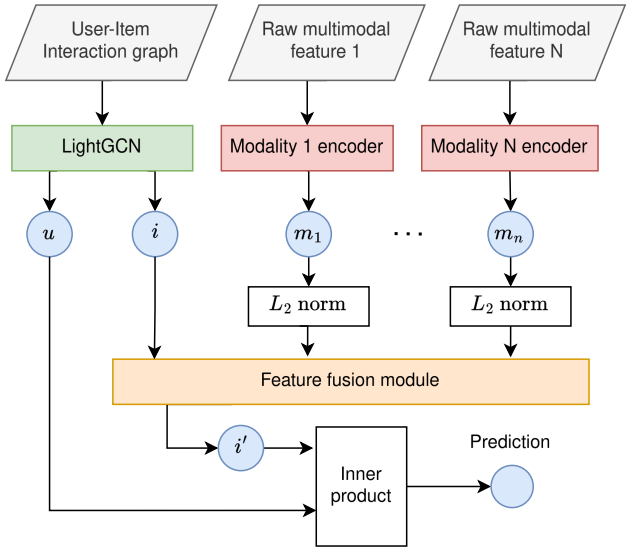

Figure 1: General architecture of Late Fusion MMLGCN

3.1 LightGCN

GCNs for recommendation fall under the category of Graph Recommender Systems, a class of recommender systems which use graphs to model user-item interactions. Specifically, the graph is set up as a heterogeneous bipartite graph, i.e. a graph in which there are two distinct types of nodes (user and item nodes) and each node has an edge only with the other type of node: users are connected exclusively to items and vice-versa.

In this way, a user's embedding is the aggregation of the embeddings of the items they have interacted with and the embedding of those items is the aggregation of the users that have interacted with them, and so on, based on the number of layers.

One specific architecture of GCNs for recommendation is LightGCN. LightGCN's authors claim that the other GCNs for recommendation in literature are too complex and thus strive to simplify the architecture of graph convolution for recommendation (hence the name "Light"). Specifically, they

claim that, in the convolution step, feature transformation layers and non-linear activation functions are useless (detrimental, even) for the recommendation task, as the structure of the graph with the user-item interactions is informative enough by itself. In fact, they show that by removing these functions, metrics improve. The only operation which is kept is the core of GCNs, i.e. the graph convolution operation. Formally:

$$e_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e_i^{(k)} \tag{4}$$

$$e_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e_u^{(k)} \tag{5}$$

In this case, the graph node itself isn't integrated. The only trainable parameters are $e_u^{(0)}$ and $e_i^{(0)}$. After K layers of the light graph convolution, the final embeddings are obtained like this:

$$e_u = \sum_{k=0}^{K} a_k e_u^{(k)} \tag{6}$$

$$e_i = \sum_{k=0}^{K} a_k e_i^{(k)} \tag{7}$$

where $a_k \geq 0$ denotes the importance of the k-th layer embedding. It can be treated as an hyperparameter of or as a learnable parameter. For the sake of simplicity, in line with the whole architecture, it is set to be $\frac{1}{(k+1)}$.

Finally, the model's prediction is defined as the inner product of user and item final representations:

$$\hat{y}_{ui} = e_u^T e_i \tag{8}$$

which is the ranking score for the recommendation step.

The loss function is the Bayesian Personalized Ranking (BPR)[10] function. The BPR loss tries to reproduce the positive ratings for unseen items. Formally:

$$L_{BPR} = -\sum_{u=1}^{M} \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|E^{(0)}\|^2 \tag{9}$$

It works under the All Missing Are Negative (AMAN) assumption: the user interacted with some items and didn't interact with some others, either because they disliked them or because they didn't know them; the AMAN

hypothesis states that all the items the user didn't interact with can be considered negative items, i.e. as if the user rated them negatively, independently of what the user might actually think about them.

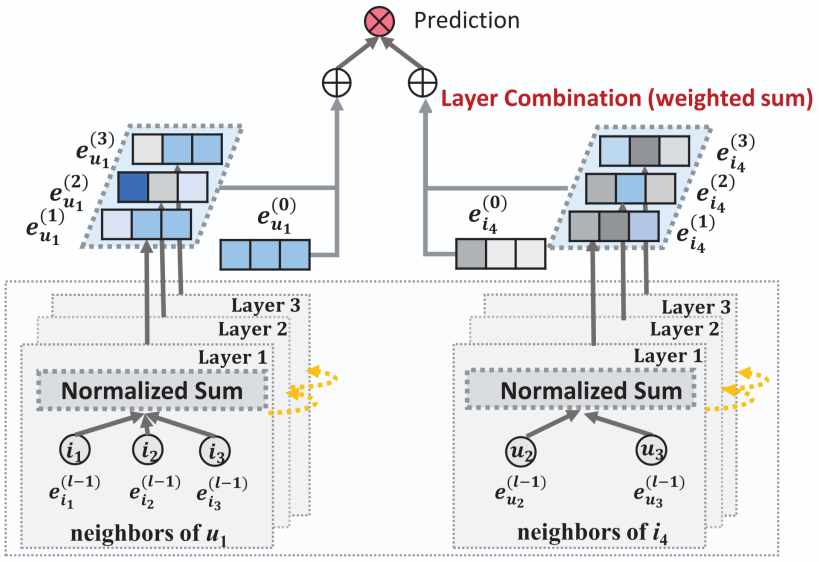

Figure 2: Architecture of LightGCN

The results show that LightGCN achieves a substantial improvement both in metrics and in performance, which proves the effectiveness of removing feature transformation and nonlinear activation functions.

3.2 Feature interaction

In the feature interaction phase, the multimodal embeddings are processed so that they can fit the task of recommendation.

Let $fuse$ be the fusion function. Conceptually, the general fusion function is formalized as:

$$fuse(e_i, m_i^1, \ldots, m_i^n). \tag{10}$$

This function as-is has some issues. Namely, the dimension and normalization of multimodal embeddings. Each multimodal embedding is extracted from a different encoder, thus it will have a different dimensionality and magnitude from the other multimodal embeddings. If the embeddings have different magnitude, they would have different weights in the fusion process, which is undesirable, as it would be a side effect of the encoding process rather than an explicit decision. Therefore, before fusing the multimodal embeddings, they should be normalized and projected to the same dimension d as

the LightGCN embeddings. Formally:

$$fuse(e_i, \|m_i^1\|_2 W_1, \ldots, \|m_i^n\|_2 W_n) \tag{11}$$

where the W_js are the projection matrices for modality m^j.

3.2.1 Late fusion

In general, the fusion stage can be either early or late, where early fusion consists in fusing the multimodal embeddings before training the model and late fusion consists in fusing the multimodal embeddings with the collaborative embeddings during the model training. They can be formalized as:

$$\begin{array}{c|c} \text{Early fusion} & \text{Late fusion} \\ fuse(e_i, fuse(m_i^1, \ldots, m_i^n)) & fuse(e_i, m_i^1, \ldots, m_i^n) \end{array} \tag{12}$$

The EF approach also allows for the freedom of using different fusion methods for multimodal embeddings and for collaborative-multimodal embeddings, e.g. $fuse_1(e_i, fuse_2(m_i^1, \ldots, m_i^n))$, whereas the LF does not. However, the possibility of using two different fusion functions isn't explored in this work; the same fusion function is used. The fusion type used in this work is Late Fusion.

3.2.2 Average fusion function

In literature, fusion architectures can be relatively elaborate, such as RNN, attention, gating mechanisms, etc. In this work, the used fusion function is the average.

When using the average fusion function, it becomes a weighted average whose weights change based on whether the fusion is early or late. In late fusion, the average of the collaborative and multimodal embeddings is taken in one step, thus, given n multimodal embeddings, each embedding will have a weight of $\frac{1}{n+1}$, where the $+1$ takes into account the collaborative embedding. This means that the collaborative embedding will have less weight compared to the entirety of multimodal embeddings as the number of multimodal embeddings grows. Formally:

$$\begin{array}{cc} & \text{Late fusion (average)} \\ \text{Formula} & \dfrac{e_i + \sum_{j \in M_i} m_i^j}{n+1} \\ \text{Weights} & e_i : \dfrac{1}{n+1}, m_i^j : \dfrac{n}{n+1} \end{array} \tag{13}$$

In this work, this weighting is kept as the base weighting scheme.

4 Experimentation

MMLGCN extends the original LightGCN implementation. However, some issues arise with this implementation. Therefore, in this work PyTorch Geometric's[11] implementation is used, as it is better documented, more modular and thus easier to extend.

The datasets and metrics used are the same as MMLGCN. The datasets are dividend into training, test and validation splits. The validation set is split from the training set with a 80/20 ratio. Here is some information about the training set for these datasets.

	DbBook	MovieLens1M
Users	4528	4828
Items	3902	2464
Sparsity	99.92%	98.69%
Interactions	26533	346501
Modalities	Text, Image	Text, Image, Audio, Video

Table 1: Information about the two datasets.

There are various parameters that are used in the system. They are the following, along with some example values:

Parameter	**Value**	**Description**
Batch size	2048	The batch size to be used
Epochs	500	The number of epochs
Learning rate	0.001	The learning rate of the model
Top K	[5, 10, 20, 50]	A list of the top K metrics
Embedding size	512	The dimension of the embeddings
Freeze	False	Whether to freeze the embeddings
Fusion function	Mean	The fusion function to use
Number of layers	2	The number of convolution layers

Table 2: Example values for the parameters.

For DbBook, the batch size is 2048, whereas for MovieLens1M it is 8192, due to the higher number of interactions.

4.1 Raw feature representation

In order for the model to work, we need a representation of the multimodal features in vector form. These features (text, audio, image, etc...) are embedded using modality-specific encoders.

For DbBook, the modalities are text and image; for MovieLens1M, the modalities are text, image, audio and video.

4.1.1 Text encoder: MiniLM

Pre-trained large language models achieve excellent performance; however, it comes at the cost of having billions of parameters, which results in a high computational cost. Therefore, a technique called "distillation" is employed to compress the model into a smaller one, while retaining most of its performance. This is achieved by having a "teacher" model, the original large model, and a "student" model, which tries to mimic the teacher's output.

The MiniLM[12] model is one such distilled model. The teacher model is BERT[13]. Specifically, MiniLM distills the teacher's self-attention module of the last Transformer[14] layer.

Given a student model S and a teacher model T, knowledge distillation is modeled as minimizing the differences between student and teacher features:

$$\mathcal{L}_{KD} = \sum_{e \in \mathcal{D}} L\left(f^S(e), f^T(e)\right) \tag{14}$$

where $\mathcal{D}$ denotes the training data, f^S and f^T denote the features of the student and teacher, respectively, and L is the loss function.

There are three main ideas behind MiniLM:

- Self-attention distribution transfer: the student mimics the teacher's last layer of the self-attention module. The objective is to minimize the KL divergence.
- Self-attention value-relation transfer: the student mimics the relations of the values in the self-attention module. The objective is to minimize the KL divergence.
- Teacher assistant: an intermediate-size model is introduced between the teacher and the student, to bridge the gap in the sizes of the models.

The teacher model used was $\text{BERT}_{\text{BASE}}$. The results show that MiniLM is able to achieve 99% of the teacher's performance while being twice as fast.

4.1.2 Image encoder: ViT

Transformers were originally conceived and have traditionally been used for NLP tasks. ViT[15] uses transformers for image embedding. This is done by embedding linearly 16x16 patches of the image and then treating each patch

embedding as a word in the traditional tranformer usage for NLP tasks. The model is trained on image classification in a supervised way.

The image patches, which are naturally in a 2D grid arrangement, are flattened before being fed to the model.

The model achieves comparable or even higher performance compared to the state of the art models.

4.1.3 Video encoder: R(2+1)D

Traditional video embedding approaches apply 2D CNNs to each frame of the video. R(2+1)D[16] applies 3D convolutions on frames over time, with separate spatial and temporal dimensions.

R(2+1)D explicitly factorizes 3D convolution into two successive operations: a 2D spatial convolution and a 1D temporal convolution. The rationale behind this separation is twofold: the first is that this introduces another layer of nonlinearity, thus allowing for the representation of more complex functions; the second is that this separation of concerns makes the model easier to optimize.

4.1.4 Audio encoder: VGGish

Traditionally, CNNs have been used for image embedding. VGGish[17] uses CNNs for audio classification.

The core idea is, instead of taking as input the audio directly, compute its spectrogram, in the form of an image, and analyze it with a CNN. This is effective in extracting information from the audio.

4.2 Architectures

The experiments are aimed at measuring the impact of the multimodal augmentation. Multimodal information is more effective in cold-case scenarios, where the collaborative information is insufficient. Thus, the hypothesis is that adding multimodal information is more effective with fewer layers, as fewer layers capture less collaborative information and multimodal information can compensate. Also, multimodal information is more effective in datasets that are more sparse; therefore, it should be more effective on DbBook. Specifically, five experiments are performed for each dataset:

1. Collaborative filtering only: used for baseline comparison, without multimodal information;

2. Fusion with frozen multimodal embeddings: the collaborative embeddings are fused with the frozen multimodal embeddings, i.e. the embeddings are fixed, they are not trained with the model;

3. Fusion with an α weighting parameter: the collaborative and multimodal embeddings are weighted with a scheme of

$$fuse(\alpha e_i, (1-\alpha)m_i^1, (1-\alpha)m_i^2, \ldots, (1-\alpha)m_i^n)$$

where $0 < \alpha < 1$ is a model parameter. This is done to make the model learn the best weight and importance of collaborative and multimodal feature relative to each other;

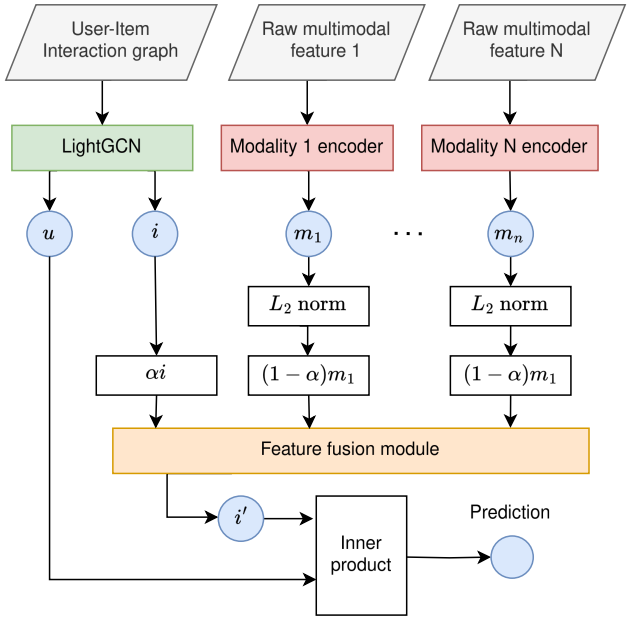

Figure 3: Fusion with an α weighting parameter

4. Fusion with unfrozen multimodal embeddings: the collaborative embeddings are fused with the unfrozen multimodal embeddings, i.e. the embeddings are treated as model parameters. Since the embeddings are pretrained, they are task-agnostic: by making the embeddings learnable, the system may be able to improve their representation for the recommendation task;

5. Fusion with collaborative embeddings normalization: the embeddings are normalized with a scheme of

$$fuse(n\|e_i\|_2, m_i^1, m_i^2, \ldots, m_i^n)$$

to balance the fact that while collaborative embeddings grow in magnitude, multimodal embeddings stay fixed.

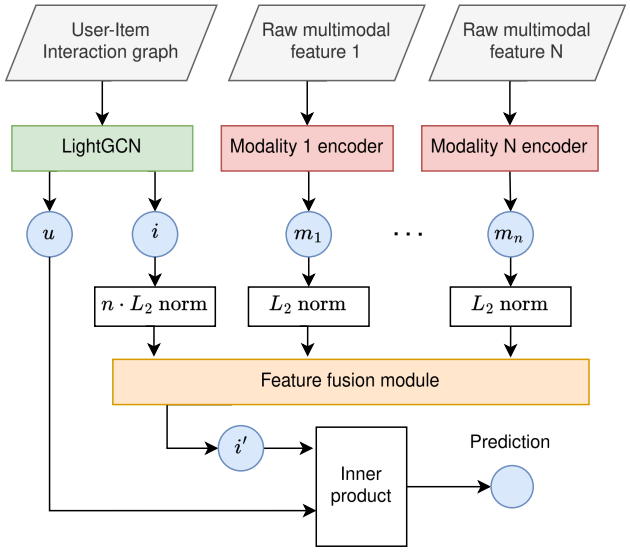

Figure 4: Fusion with collaborative embeddings normalization

All of these experiments are performed with the Late Fusion and the average fusion function.

The metrics used to evaluate the system are precision@k, recall@k and NDCG@k for $k \in \{5, 10, 20, 50\}$.

The models have been trained on a NVIDIA A16 GPU.

5 Results

5.1 LightGCN

The LightGCN results, overall, show that 3 convolution layers achieve the best performance. For both MovieLens1M (Table 4) and DbBook (Table 3), this is true for all ks.

This indicates that message passing and graph convolution are effective in capturing more useful information.

It should be noted that MovieLens1M largely overfits (Figure 5), since the validation loss rises while the training loss decreases; DbBook doesn't.

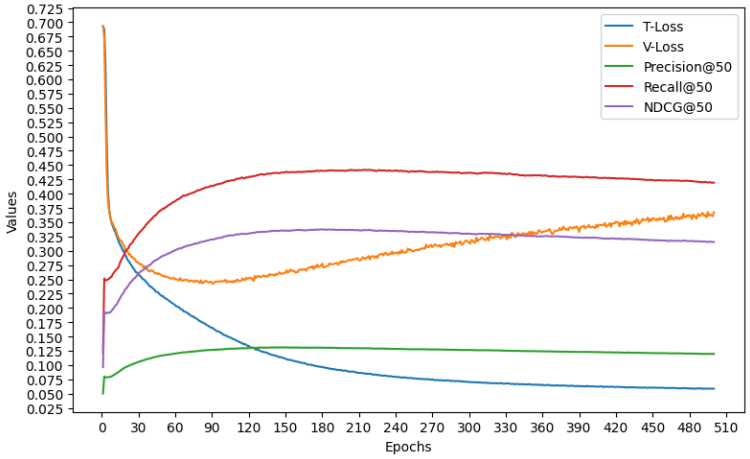

Figure 5: LightGCN - MovieLens1M - 2 layers

	Top 5			Top 10			Top 20			Top 50		
#	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
0	0.0583	0.0468	0.0693	0.0442	0.0699	0.0696	0.0327	0.1017	0.0820	0.0223	0.1696	0.1065
1	0.0621	0.0495	0.0739	0.0470	0.0740	0.0740	0.0350	0.1078	0.0873	0.0235	0.1787	0.1128
2	0.0630	0.0507	0.0756	0.0485	**0.0759**	0.0760	0.0359	0.1114	0.0898	0.0242	0.1840	0.1161
3	**0.0640**	**0.0515**	**0.0762**	**0.0488**	**0.0759**	**0.0763**	**0.0369**	**0.1139**	**0.0911**	**0.0247**	**0.1876**	**0.1177**

Table 3: LightGCN - DbBook

	Top 5			Top 10			Top 20			Top 50		
#	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
0	0.1848	0.0777	0.2012	0.1599	0.1288	0.1970	0.1337	0.2059	0.2085	0.0989	0.3536	0.2538
1	0.2253	0.0950	0.2468	0.1921	0.1555	0.2392	0.1569	0.2399	0.2488	0.1117	0.3956	0.2950
2	0.2475	0.1051	0.2694	0.2095	0.1689	0.2600	0.1704	0.2586	0.2696	0.1195	0.4185	0.3163
3	**0.2603**	**0.1093**	**0.2823**	**0.2210**	**0.1758**	**0.2728**	**0.1797**	**0.2700**	**0.2824**	**0.1249**	**0.4339**	**0.3292**

Table 4: LightGCN - MovieLens1M

5.2 Multimodal with frozen embeddings

For both DbBook (Table 5) and MovieLens1M (Table 6) the number of layers seems less important. For DbBook, the best results are achieved in most cases with 1 or 2 layers; for MovieLens1M, the best results are most often achieved at layer 3 with some exception in which the performance is best with 2 layers.

For DbBook, the performance compared to its collaborative counterpart is considerably higher. The overall improvement is about 20%-25%. For MovieLens1M, however, the situation is different. The improvements are in the range of 5%-10%.

These results suggest that for MovieLens1M, which is denser than DbBook, adding multimodal information is less beneficial, as the collaborative

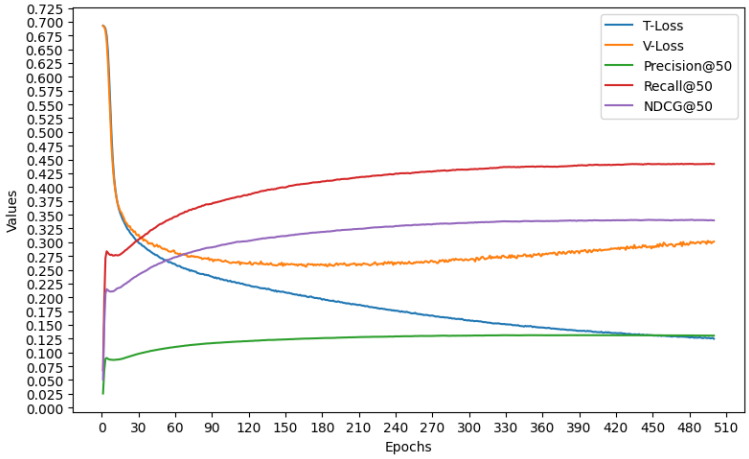

Figure 6: LF-MMLGCN - MovieLens1M - 2 layers

information is already sufficient. Conversely, for DbBook, multimodal information is useful to compensate for the scarcity of collaborative information.

It is important to note that this method greatly reduces overfitting on MovieLens1M compared to the collaborative counterpart (Figure 6).

	Top 5			Top 10			Top 20			Top 50		
#	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
0	0.0745	0.0613	0.0906	0.0579	0.0925	0.0920	0.0426	0.1352	0.1082	0.0271	0.2103	0.1353
1	**0.0790**	**0.0650**	**0.0941**	0.0607	0.0972	0.0954	0.0444	0.1396	0.1118	0.0285	0.2185	0.1405
2	**0.0790**	0.0644	0.0940	**0.0618**	**0.0985**	**0.0961**	**0.0451**	**0.1412**	**0.1126**	0.0293	0.2231	**0.1423**
3	0.0779	0.0631	0.0924	0.0615	0.0972	0.0948	**0.0451**	0.1409	0.1116	**0.0296**	**0.2255**	0.1422

Table 5: LF-MMLGCN - DbBook - Frozen

	Top 5			Top 10			Top 20			Top 50		
#	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
0	0.2295	0.0960	0.2503	0.1951	0.1578	0.2426	0.1592	0.2438	0.2522	0.1134	0.3995	0.2983
1	0.2635	0.1112	0.2850	0.2232	0.1763	0.2751	0.1800	0.2675	0.2829	0.1252	0.4328	0.3299
2	0.2786	**0.1145**	0.3021	0.2339	**0.1802**	0.2886	0.1889	**0.2739**	0.2950	0.1307	**0.4426**	0.3414
3	**0.2829**	0.1131	**0.3074**	**0.2389**	0.1796	**0.2935**	**0.1915**	0.2736	**0.2976**	**0.1318**	0.4383	**0.3415**

Table 6: LF-MMLGCN - MovieLens1M - Frozen

5.3 LF-MMLGCN with α weighting

Weighting the embeddings with an α parameter penalizes the higher layers for both DbBook (Table 7) and MovieLens1M (Table 8), as most of the best results are obtained with two layers.

Regarding the values of the α parameter, they are all equal to 0.9999 in MovieLens1M, whereas in DbBook they start from 0.8827 at layer 0 and

progressively increase as the layers increase, up to 0.9739 at layer 3. This is probably due to the fact that MovieLens1M is denser, therefore multimodal information is less needed. In fact, in DbBook, which is sparser, more weight is given to multimodal embeddings in lower layers, where collaborative information is scarce.

	Top 5			Top 10			Top 20			Top 50			
#	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	α
0	0.0736	0.0605	0.0886	0.0569	0.0909	0.0897	0.0415	0.1309	0.1050	0.0268	0.2063	0.1323	0.8827
1	0.0769	**0.0631**	**0.0914**	0.0596	0.0946	0.0928	0.0435	0.1365	0.1090	0.0281	0.2146	0.1374	0.9422
2	**0.0772**	0.0625	0.0908	**0.0605**	**0.0962**	**0.0931**	0.0444	0.1383	**0.1094**	0.0289	0.2192	0.1388	0.9635
3	0.0764	0.0618	0.0901	0.0600	0.0945	0.0920	**0.0445**	**0.1384**	0.1091	**0.0292**	**0.2221**	**0.1393**	0.9739

Table 7: LF-MMLGCN - DbBook - Frozen - α

	Top 5			Top 10			Top 20			Top 50			
#	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	α
0	0.2337	0.0987	0.2544	0.1971	0.1597	0.2454	0.1607	0.2456	0.2547	0.1151	0.4052	0.3025	0.9999
1	0.2673	0.1125	0.2902	0.2259	0.1777	0.2791	0.1828	0.2718	0.2874	0.1271	0.4375	0.3343	0.9999
2	**0.2794**	**0.1137**	**0.3018**	0.2358	**0.1811**	**0.2892**	0.1893	**0.2750**	**0.2948**	**0.1322**	**0.4454**	**0.3423**	0.9999
3	0.2783	0.1098	0.3017	**0.2359**	0.1767	0.2886	**0.1897**	0.2701	0.2931	0.1314	0.4356	0.3376	0.9999

Table 8: LF-MMLGCN - MovieLens1M - Frozen - α

5.4 LF-MMLGCN with unfrozen embeddings

The results are mixed. For DbBook (Table 9), the performance is worse than the other two multimodal methods, but still higher than the collaborative only method. For MovieLens1M (Table 10), the performance is worse than the collaborative only counterpart, except with 0 layers (which is equivalent to matrix factorization). However, it is important to note that unfreezing the embeddings causes a strong overfitting on MovieLens1M.

	Top 5			Top 10			Top 20			Top 50		
#	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
0	0.0688	0.0563	0.0835	0.0528	0.0841	0.0840	0.0390	0.1238	0.0992	0.0255	0.1963	0.1255
1	0.0725	0.0593	0.0871	0.0554	0.0886	0.0876	0.0407	0.1287	0.1030	0.0263	0.2024	0.1297
2	**0.0736**	**0.0596**	**0.0881**	0.0560	0.0895	0.0887	0.0414	0.1310	0.1046	0.0266	0.2049	0.1314
3	0.0733	0.0594	0.0879	**0.0563**	**0.0901**	**0.0890**	**0.0419**	**0.1320**	**0.1053**	**0.0268**	**0.2062**	**0.1322**

Table 9: LF-MMLGCN - DbBook - Unfrozen

	Top 5			Top 10			Top 20			Top 50		
#	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
0	0.1846	0.0769	0.1996	0.1600	0.1288	0.1963	0.1351	0.2082	0.2092	0.1005	0.3585	0.2559
1	0.2106	0.0881	0.2288	0.1800	0.1450	0.2224	0.1491	0.2296	0.2340	0.1080	0.3848	0.2807
2	0.2245	0.0938	0.2436	0.1906	0.1535	0.2359	0.1566	0.2398	0.2465	0.1121	0.3976	0.2934
3	**0.2331**	**0.0983**	**0.2535**	**0.1975**	**0.1585**	**0.2447**	**0.1615**	**0.2469**	**0.2548**	**0.1149**	**0.4053**	**0.3017**

Table 10: LF-MMLGCN - MovieLens1M - Unfrozen

5.5 LF-MMLGCN with collaborative embedding normalization

For DbBook (Table 11), normalization has a negative effect, as the results are lower than the unnormalized version, but still higher than the collaborative-only counterpart. Interestingly, the best results are at layer 0. One possible explanation could be that normalizing the embeddings prevents their growth, which is higher the higher the number of layers. For MovieLens (Table 12), normalization tends to perform slightly better than the unnormalized version.

	Top 5			Top 10			Top 20			Top 50		
#	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
0	**0.0747**	**0.0605**	**0.0886**	**0.0577**	**0.0923**	**0.0902**	**0.0422**	**0.1313**	**0.1054**	0.0285	0.2166	**0.1363**
1	0.0740	0.0592	0.0877	0.0567	0.0903	0.0888	0.0419	0.1300	0.1043	**0.0286**	**0.2172**	0.1358
2	0.0723	0.0579	0.0859	0.0554	0.0882	0.0869	0.0411	0.1266	0.1020	0.0284	0.2142	0.1337
3	0.0698	0.0562	0.0844	0.0538	0.0852	0.0851	0.0405	0.1245	0.1006	0.0278	0.2096	0.1315

Table 11: LF-MMLGCN - DbBook - Frozen - Normalized

	Top 5			Top 10			Top 20			Top 50		
#	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
0	0.2654	0.1109	0.2896	0.2231	0.1765	0.2773	0.1778	0.2654	0.2825	0.1235	0.4275	0.3282
1	0.2849	0.1163	0.3108	0.2396	0.1840	0.2964	0.1903	0.2764	0.2996	0.1306	0.4422	0.3443
2	**0.2892**	**0.1158**	**0.3146**	**0.2442**	**0.1843**	**0.3001**	**0.1943**	**0.2776**	**0.3027**	**0.1324**	**0.4431**	**0.3455**
3	0.2860	0.1128	0.3100	0.2427	0.1806	0.2962	0.1933	0.2733	0.2986	0.1320	0.4376	0.3408

Table 12: LF-MMLGCN - MovieLens1M - Frozen - Normalized

5.6 Comparison between LF-MMLGCN variants

We can make a comparison of all models. For DbBook, the results (Table 13) show that the best performing model for all metrics and values of k is LF-MMLGCN. Figure 7 groups the models by metric for $k = 10$, showing the best model for each metric. Figure 8 presents a comparison of all models with a fixed $k = 10$, which is practical for most common scenarios. We can see how all multimodal models perform better than LightGCN.

	Top 5			Top 10			Top 20			Top 50		
Model	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
LightGCN	0.0640	0.0515	0.0762	0.0488	0.0759	0.0763	0.0369	0.1139	0.0911	0.0247	0.1876	0.1177
LF-MMLGCN	**0.0790**	**0.0650**	**0.0941**	**0.0618**	**0.0985**	**0.0961**	**0.0451**	**0.1412**	**0.1126**	**0.0296**	**0.2255**	**0.1423**
LF-MMLGCN (α)	0.0772	0.0631	0.0914	0.0605	0.0962	0.0931	0.0445	0.1384	0.1094	0.0292	0.2221	0.1393
LF-MMLGCN (unfrozen)	0.0736	0.0596	0.0881	0.0563	0.0901	0.0890	0.0419	0.1320	0.1053	0.0268	0.2062	0.1322
LF-MMLGCN (normalized)	0.0747	0.0605	0.0886	0.0577	0.0923	0.0902	0.0422	0.1313	0.1054	0.0286	0.2172	0.1363

Table 13: Comparison of LightGCN and the LF-MMLGCN variants - DbBook

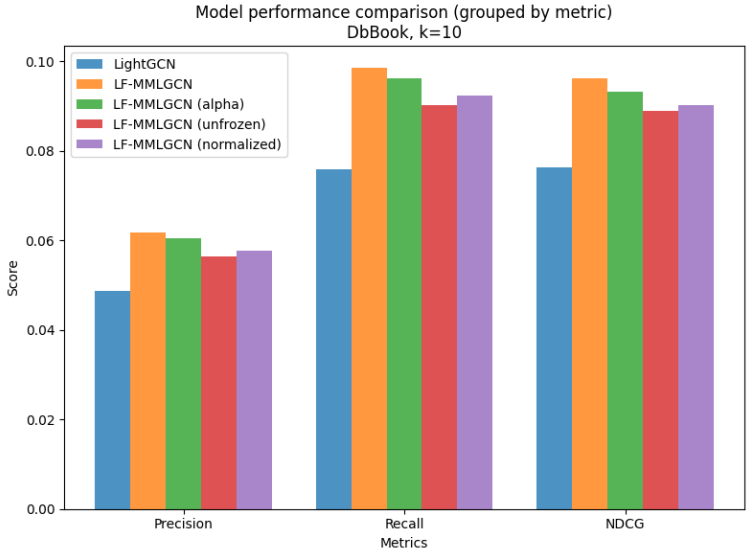

Figure 7: DbBook comparison (grouped by metric), k=10

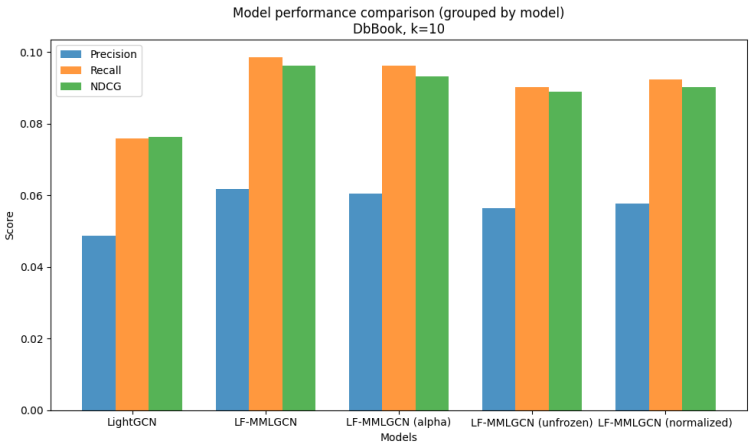

Figure 8: DbBook comparison (grouped by model), k=10

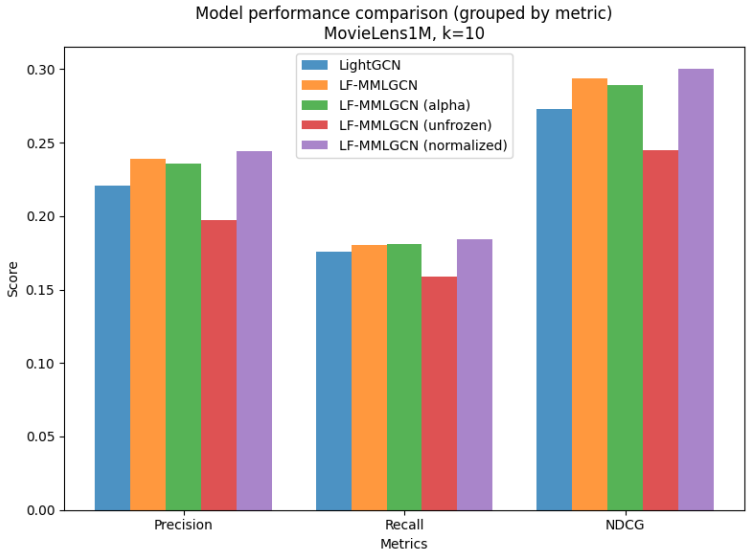

Figure 9: MovieLens1M comparison (grouped by metric), k=10

For MovieLens1M, the results (Table 14) show that the best performing model for all metrics and values of k is LF-MMLGCN (normalized), except for recall@50, in which the best model is LF-MMLGCN (alpha). Figure 9 groups the models by metric for $k = 10$, showing the best model for each metric. Figure 10 shows how the difference between models is less marked compared to DbBook. This is due to the fact that MovieLens1M has more collaborative information, so adding multimodal information doesn't change the model a lot. As already mentioned, the best model is LF-MMLGCN (normalized); the chart shows that the worst is LF-MMLGCN (unfrozen).

	Top 5			Top 10			Top 20			Top 50		
Model	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
LightGCN	0.2603	0.1093	0.2823	0.2210	0.1758	0.2728	0.1797	0.2700	0.2824	0.1249	0.4339	0.3292
LF-MMLGCN	0.2829	0.1145	0.3074	0.2389	0.1802	0.2935	0.1915	0.2739	0.2976	0.1318	0.4426	0.3415
LF-MMLGCN (alpha)	0.2794	0.1137	0.3018	0.2359	0.1811	0.2892	0.1897	0.2750	0.2948	0.1322	**0.4454**	0.3423
LF-MMLGCN (unfrozen)	0.2331	0.0983	0.2535	0.1975	0.1585	0.2447	0.1615	0.2469	0.2548	0.1149	0.4053	0.3017
LF-MMLGCN (normalized)	**0.2892**	**0.1158**	**0.3146**	**0.2442**	**0.1843**	**0.3001**	**0.1943**	**0.2776**	**0.3027**	**0.1324**	0.4431	**0.3455**

Table 14: Comparison of LightGCN and the LF-MMLGCN variants - MovieLens1M

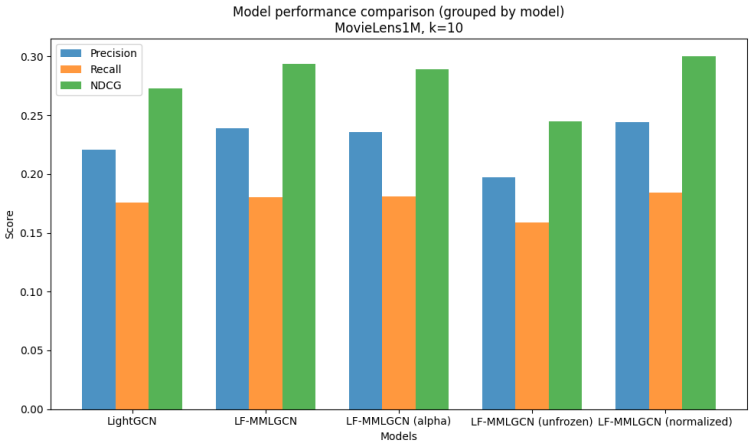

Figure 10: MovieLens1M comparison (grouped by model), k=10

5.7 Comparison with baseline

The model has been compared with other multimodal recommendation models. These are LATTICE[18], MMGCN[7] and VBPR[19].

In Table 15 for DbBook, we can see how LATTICE and LF-MMLGCN are the best models. LATTICE is the best overall, with LF-MMLGCN being better in precision@5, precision @10 and NDCG@20. LF-MMLGCN manages to be competitive with LATTICE, being slightly worse. There is a very large improvement for LF-MMLGCN compared to MMGCN and VBPR.

	Top 5			Top 10			Top 20			Top 50		
Model	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
LATTICE	0.0789	**0.0653**	**0.0948**	0.0616	**0.0994**	**0.0971**	**0.0465**	**0.1450**	0.1154	**0.0305**	**0.2338**	**0.1473**
MMGCN	0.0362	0.0266	0.0388	0.0311	0.0456	0.0422	0.0273	0.0813	0.0569	0.0213	0.1568	0.0844
VBPR	0.0225	0.0547	0.0466	0.0159	0.0772	0.0553	0.0108	0.1059	0.0640	0.0064	0.1525	0.0753
LF-MMLGCN	**0.0790**	0.0650	0.0941	**0.0618**	0.0985	0.0961	0.0451	0.1412	**0.1162**	0.0296	0.2255	0.1423

Table 15: Comparison between baseline models and LF-MMLGCN - DbBook

In Table 16 for MovieLens1M, the best performing model is LF-MMLGCN for all metrics and all values of k. LATTICE manages to be a close second. LF-MMLGCN outperforms by a large margin MMGCN and VBPR also for MovieLens1M.

	Top 5			Top 10			Top 20			Top 50		
Model	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG	Precision	Recall	NDCG
LATTICE	0.2256	0.0984	0.2433	0.1947	0.1612	0.2395	0.1616	0.2508	0.2528	0.1174	0.4187	0.3051
MMGCN	0.1567	0.0605	0.1658	0.1360	0.1013	0.1615	0.1156	0.1650	0.1708	0.0876	0.2950	0.2118
VBPR	0.1640	0.0627	0.1745	0.1438	0.1050	0.1703	0.1220	0.1725	0.1799	0.0928	0.3089	0.2224
LF-MMLGCN	**0.2829**	**0.1145**	**0.3074**	**0.2389**	**0.1802**	**0.2935**	**0.1915**	**0.2739**	**0.2976**	**0.1322**	**0.4454**	**0.3415**

Table 16: Comparison between baseline models and LF-MMLGCN - MovieLens1M

5.8 Discussion

The research questions can now be addressed.

To answer **RQ1** we can observe how multimodal information greatly benefits LightGCN, especially in DbBook, due to its greater sparsity compared to MovieLens1M. To answer **RQ2**, which architecture is the best depends on the dataset: for DbBook, it's the regular LF-MMLGCN with frozen embeddings, whereas for MovieLens1M it's LF-MMLGCN with item embedding normalization. Finally, to answer **RQ3**, for DbBook, LF-MMLGCN performs much better than MMGCN and VBPR, and LATTICE performs slightly better than LF-MMLGCN. For MovieLens1M, LF-MMLGCN, again, greatly outperforms MMGCN and VBPR; it also outperforms LATTICE, albeit by a smaller margin compared to MMGCN and VBPR.

The results also reveal some interesting insights:

- Adding more layers is very useful, but reduces the relative improvement compared to the collaborative results, as the multimodal information compensates less for the lack of collaborative information
- The α parameter is very high, giving little to no weight to multimodal information. However, the measurement is taken at the end of training. Therefore, what matters, is how the parameter changes over the epochs and how it affects training;
- Unfreezing the embeddings has an overall negative effect and strongly overfits on MovieLens1M;
- Normalizing the collaborative embeddings has an overall negative effect on DbBook and an overall positive effect on MovieLens1M;
- MovieLens1M has a tendency to overfit, probably due to the higher density; multimodal information greatly reduces this phenomenon, probably because it allows the model to better generalize the users' preferences.

Overall, the results are in line with what the literature reports, i.e. adding multimodal information is more beneficial in scenarios in which collaborative information is insufficient, due to the issues of sparsity or cold-start.

6 Conclusion and future work

This work has explored how to combine collaborative and multimodal information. Specifically, how to fuse the LightGCN embeddings with the

multimodal embeddings. The fusion method used is Late Fusion with the average fusion function. The datasets used are DbBook and MovieLens1M; their main difference is that DbBook is sparser than MovieLens1M, which is very relevant. In fact, the results show that adding multimodal information is more beneficial in cases where the collaborative information is lacking, i.e. in sparse datasets like DbBook or with few layers of graph convolution.

Future work could focus on these points:

- Explore differences between early and late fusion;
- Try different weighting schemes for balancing collaborative and multimodal embeddings;
- Explore different fusion functions (e.g. sum, concatenation, etc...)
- Study in depth how the magnitude of the embeddings affects the results. Since the final prediction formula is the dot product between users and items, modifying the items with multimodal information has the side effect of changing their magnitude and thus changing the prediction. Therefore, we can't be sure how much of the change is due to multimodal information and how much is due to a simple change in magnitude;
- Try to align the embedding spaces of the collaborative and multimodal embeddings.

Other methods have been briefly tried to improve the effectiveness of multimodal information, but their results were uncertain and require further inquiry: single-branch embedding, autoencoder, info NCE loss and ensemble fusion.

References

[1] P. Lops, C. Musto, F. Narducci, and G. Semeraro, *Semantics in Adaptive and Personalised Systems - Methods, Tools and Applications*. Springer, 2019, ISBN: 978-3-030-05617-9. DOI: 10.1007/978-3-030-05618-6. [Online]. Available: https://doi.org/10.1007/978-3-030-05618-6.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016. arXiv: 1609.02907. [Online]. Available: http://arxiv.org/abs/1609.02907.

[3] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '20, Virtual Event, China: Association for Computing Machinery, 2020, pp. 639648, ISBN: 9781450380164. DOI: 10.1145/3397271.3401063. [Online]. Available: https://doi.org/10.1145/3397271.3401063.

[4] Q. Liu, J. Hu, Y. Xiao, *et al.*, "Multimodal recommender systems: A survey," *ACM Comput. Surv.*, vol. 57, no. 2, 26:1–26:17, 2025. DOI: 10.1145/3695461. [Online]. Available: https://doi.org/10.1145/3695461.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009. DOI: 10.1109/MC.2009.263.

[6] G. Gkarmpounis, C. Vranis, N. Vretos, and P. Daras, "Survey on graph neural networks," *IEEE Access*, vol. 12, pp. 128 816–128 832, 2024. DOI: 10.1109/ACCESS.2024.3456913.

[7] Y. Wei, X. Wang, L. Nie, X. He, R. Hong, and T. Chua, "MMGCN: multi-modal graph convolution network for personalized recommendation of micro-video," in *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*, L. Amsaleg, B. Huet, M. A. Larson, *et al.*, Eds., ACM, 2019, pp. 1437–1445. DOI: 10.1145/3343031.3351034. [Online]. Available: https://doi.org/10.1145/3343031.3351034.

[8] S. Wang, Y. Yang, J. Yang, and J. Liu, "A dynamic collaborative recommendation method based on multimodal fusion," in *Advanced Intelligent Computing Technology and Applications - 20th International Conference, ICIC 2024, Tianjin, China, August 5-8, 2024, Proceedings, Part I (LNAI)*, D. Huang, X. Zhang, and Q. Zhang, Eds., ser. Lecture Notes in Computer Science, vol. 14875, Springer, 2024, pp. 3–14. DOI: 10.1007/978-981-97-5663-6_1. [Online]. Available: https://doi.org/10.1007/978-981-97-5663-6_1.

[9] F. Bottalico, *Introducing Multimodal Features in Recommender Systems based on Graph Convolutional Networks*. Masters degree course in Computer Science. Università degli Studi di Bari Aldo Moro, 2024.

[10] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelli-*

gence, ser. UAI '09, Montreal, Quebec, Canada: AUAI Press, 2009, pp. 452461, ISBN: 9780974903958. DOI: 10.5555/1795114.1795167.

[11] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *CoRR*, vol. abs/1903.02428, 2019. arXiv: 1903.02428. [Online]. Available: http://arxiv.org/abs/1903.02428.

[12] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *CoRR*, vol. abs/2002.10957, 2020. arXiv: 2002.10957. [Online]. Available: https://arxiv.org/abs/2002.10957.

[13] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/V1/N19-1423. [Online]. Available: https://doi.org/10.18653/v1/n19-1423.

[14] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020. arXiv: 2010.11929. [Online]. Available: https://arxiv.org/abs/2010.11929.

[16] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 6450–6459. DOI: 10.1109/CVPR.2018.00675. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Tran_A_Closer_Look_CVPR_2018_paper.html.

[17] S. Hershey, S. Chaudhuri, D. P. W. Ellis, *et al.*, "CNN architectures for large-scale audio classification," *CoRR*, vol. abs/1609.09430, 2016. arXiv: 1609.09430. [Online]. Available: http://arxiv.org/abs/1609.09430.

[18] J. Zhang, Y. Zhu, Q. Liu, S. Wu, S. Wang, and L. Wang, "Mining latent structures for multimedia recommendation," in *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021*, H. T. Shen, Y. Zhuang, J. R. Smith, *et al.*, Eds., ACM, 2021, pp. 3872–3880. DOI: 10.1145/3474085.3475259. [Online]. Available: https://doi.org/10.1145/3474085.3475259.

[19] R. He and J. J. McAuley, "VBPR: visual bayesian personalized ranking from implicit feedback," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, D. Schuurmans and M. P. Wellman, Eds., AAAI Press, 2016, pp. 144–150. DOI: 10.1609/AAAI.V30I1.9973. [Online]. Available: https://doi.org/10.1609/aaai.v30i1.9973.