MIA: Mutual Information Alignment for Side Information-Enhanced Recommendation with Multiple Views

Zhaowei Zhang

Faculty of Electrical Engineering, McGill University, Montreal

October, 2021

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Engineering

Abstract

The past decade has witnessed significant progress in recommender systems. With the development of information extraction technology, rich side information about the users and items has become available. This provides an opportunity to enhance the system's understanding of users' preferences. Several recent recommender systems use side information to enhance performance by treating it as an extra "view". The systems construct latent representations of users and items by relating information from multiple views. In other domains such as image classification, successful algorithms have demonstrated the usefulness of maximizing mutual information across views. This approach has not been explored in the setting of recommender systems with side information. In this thesis, we propose a new side information enhanced recommender system called MIA, which estimates **M**utual **I**nformation across multiple views for **A**lignment. We compare the performance with nine state-of-the-art methods across five datasets. We find that MIA achieves significant improvement over all models that ignore side information. This demonstrates the effectiveness of the proposed model in ingesting the additional information. However, the proposed model does not outperform some state-of-the-art baselines. Therefore, further analysis is conducted to understand the limitations of MIA and to provide a clearer explanation of the relationships with existing work. We explicitly identify the alignment mechanisms and loss functions employed in other systems. Experimental results show that maximizing mutual information across views has a similar effect as minimizing the cosine distance of the related representations once they have been transferred into a unified space.

Abrégé

Ces dix dernières années, les systèmes de recommandation ont fait des progrès considérables. Avec le développement de la technologie d'extraction d'informations, de riches informations secondaires sur les utilisateurs et les articles deviennent disponibles, ce qui permet au système de mieux comprendre les préférences des utilisateurs. Un paradigme des systèmes de recommandation améliorés par des informations secondaires existantes intègre des informations secondaires supplémentaires dans une vue additionnelle, puis extrait la représentation latente de l'utilisateur et de l'article en reliant les informations de plusieurs vues. Bien qu'il existe un grand nombre d'algorithmes réussis qui démontrent l'utilité de la maximisation de l'information mutuelle entre les vues, beaucoup d'entre eux sont proposés pour d'autres domaines d'application tels que la classification d'images, ou pour des systèmes de recommandation avec seulement des données d'interaction. Sur la base de cette observation, nous proposons un nouvel algorithme amélioré par l'information latérale appelé MIA qui estime l'information mutuelle entre plusieurs vues pour l'alignement. Nous comparons ses performances avec celles de neuf méthodes de pointe sur cinq ensembles de données. En particulier, nous constatons que le cadre MIA proposé obtient une amélioration significative par rapport au modèle ne prenant pas d'informations latérales en entrée, ce qui démontre l'efficacité du modèle proposé à ingérer les informations supplémentaires. Cependant, le modèle proposé n'est pas plus performant que certains des modèles de base. Par conséquent, une analyse plus approfondie est menée pour comprendre les limites du MIA. Les résultats expérimentaux montrent que la maximisation de l'information mutuelle entre les vues a un effet similaire

à la minimisation de la distance cosinusoïdale des représentations connexes transférées dans un espace unifié.

Acknowledgements

Throughout this work I have received tremendous help and support. I would like to start off by expressing my sincere gratitude to my supervisor Professor Mark Coates for his constant involvement and support during this research process. The guidance and insightful feedback he provided through those frustrating times can never be overestimated.

I wish to acknowledge my colleagues Yingxue Zhang (Research Engineer from Huawei), Florence Robert-Regol (PhD candidate), Chen Ma (PhD candidate) for all the scientific discussions that we had and all the effort I have received along the way. I am also thankful for every member of the McGill Computer Networks Research Lab for fostering a friendly and enriching environment to work in.

Conducting research especially during the pandemic is not easy. Thank you Erdon for always being there to listen, offer me advice and share your rich knowledge of machine learning with me. Your patience and caring has been a precious support throughout.

Most of all, I am eternally grateful to my mom for profoundly believing in me and always being a wise mentor and patient listener throughout my life. I could not have completed this work without you.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

With the rapid development of the Internet, the volume of online information being generated has increased in the past two decades, which has created an *information overload* problem for many internet users. Therefore, recommender systems have been used as an effective solution to search through this information and provide users with personalized contents and service. Whether they are employed to support media consumption or to increase sales, recommender systems are now ubiquitous in our daily experience. For example, Amazon recommends products based on our purchase history [4, 5]; YouTube recommends videos based on our viewing history [6]; Spotify recommends songs based on songs that we have listened to [7].

One dominant framework for recommender systems is the Collaborative Filtering (CF) technique which builds on the assumption that users with similar interaction patterns for a subset of items have similar preferences. These interactions can include the purchase or click history of users. The major task of a CF-based recommender system is to exploit user-item (UI) relations and predict potential interactions between users and items. Specifically, it focuses on learning latent representations for users and items such that UI interactions could be reconstructed.

Early matrix factorization (MF) methods tend to directly map user/item ID into an embedding space and model the UI interaction as an inner product of the corresponding user and item embedding pair [8]. With the development of deep learning, extensions are proposed to extend MF methods by adding in non-linear neural networks in the process of embedding generation or interaction modelling [9–11]. Despite the effectiveness of the matrix factorization technique for collaborative filtering, its performance is notoriously unstable and can be hindered by the simple choice of the interaction function. In other words, the collaborative signal is not properly captured by the aforementioned models. When modelling user-item interaction, it is natural to consider a graph representation which is well-known for its power to concisely represent the relational information within its structure [12]. As a result, many graph-based recommendation algorithms have been proposed and achieved promising progress by exploring the collaborative signal propagation in UI bipartite graphs under the graph neural network framework [1, 13–15].

One practical issue concerning the conventional CF-based recommender system is that ratings are inferred from the UI matrix and thus the system can struggle to draw any inferences for items with few or no interactions, also referred to as "cold items" [16–18]. Therefore, recommendations on these items are no better than random. One way to overcome this problem of learning representations for items with few interaction records is to use item attributes. The information residing in these attributes is referred to as *item side information*. In recent years, with the increasing availability and quality of item attributes, multiple strategies have been proposed that strive to learn more comprehensive item representations by integrating such side information [1, 2, 19, 20]. Intuitively, item side information can be considered as another view of an item, distinct from the user's perspective. With this viewpoint in mind, more recent work has focused on utilizing the multi-view representation learning method in side information enhanced recommendations [13, 21].

Despite the great success of these side information enhanced recommender systems, there lacks a unified framework to compare these methods from the perspective of *attribute-*

item association (AIA). For example, Wang et al. present their work in [1] following a graph representation learning framework where AIA is a combination of information propagation and information aggregation. Zhou et al. describe the AIA process in [21] as a self-supervised learning framework. In [20], Vasile et al. explain AIA from the perspective of matrix factorization. Therefore, the true difference of how category information is fused into item representations among these methods remains mysterious.

As such, the central goal of this research is to study and compare the factors that lead to success in side information enhanced recommendations. To achieve this goal, we first develop a unified framework that explicitly highlights the AIA diversity of various methods given equal notations. Specifically, we organize the various method based upon two key functions: the AIA function and its corresponding loss function.

In addition, inspiring by the recent success of mutual information maximization (MIM) application in the field of image classification [22, 23], we propose an item side information enhanced recommendation method which exploits the Mutual Information Alignment (MIA) technique in the multi-view representation framework.

1.2 Thesis Contributions and Organization

The major contributions in this thesis can be summarized as follows:

- We propose that many of the existing side information enhanced recommendation approaches can be organized within a unified framework from the item-attribute association perspective. Details are provided in Chapter 4.
- Motivated by the recent progress in the application of mutual information maximization in representation learning, we proceed to develop a mutual information alignment-based (MIA) multi-view representation learning algorithm for top-N recommendation task, as presented in Chapter 5.

The rest of this thesis is structured as follows:

- *Chapter 2 - Background*

  In this chapter, some detailed background materials essential to understanding the thesis research are provided. We start with a brief review of the theoretical foundation of graph theory and a high-level overview of node embedding learning methods and graph neural networks. Then, we continue to give a formal definition of the recommendation problem and survey key advances in this area. Subsequently, we present the mutual information principle and describe recent progress in its application to the image classification task. Lastly, we provide a comprehensive review of multi-view representation learning approaches from the alignment and fusion perspectives. This provides the necessary background for our proposed unified framework.

- *Chapter 3 - Related Work*

  This chapter focuses on reviewing recent progress in side information enhanced recommendation approaches. These are closely related to our work and inspire some of our architectural choices.

- *Chapter 4 - Side information Enhanced Methods in A Unified Framework*

  This chapter contains the first research contribution of the thesis. In this chapter, we develop a unified framework with which we can express many of the existing strategies for side information enhanced recommendation. The value of such a framework is that the similarities and differences between existing methods are immediately apparent. This makes it much easier to assess why one method might outperform another and can motivate future research directions. The framework consists of two components: an item-attribute association function which computes the association score between an item-attribute pair, and an association loss function which is used to train the item-attribute association function. We demonstrate how a selection of representative methods [1–3, 21] can be expressed in this framework.

We establish the equivalence between the expression of each method in the original paper and the expression in our proposed framework. We also introduce an adapted method, CLIP [3], which was originally proposed as a solution to the image classification problem. The Huawei machine learning researcher Yingxue Zhang first suggested adapting CLIP to be used in the recommendation scenario. Ph.D. student Haolun Wu conducted initial experiments to prove the effectiveness of this idea. I implemented the same model independently and tested its performance on another six different datasets. The development of the framework, the derivation of proofs and the experimental results related to CLIP presented throughout this thesis were my contribution. My supervisor, Prof. Mark Coates, provided guidance and feedback about the methods and results.

- *Chapter 5 - Mutual Information Alignment (MIA)*

  This chapter contains the second contribution of the thesis. We present a flexible mutual information alignment-based multi-view representation learning approach for the top-N recommendation task and report the experimental results. This approach models the UI interaction and item attributes as two views of items and aims to maximize the mutual information between related embeddings from these two views to improve the quality of item representations. Extensive experiments are conducted on six public real-world benchmark datasets from various domains. We compare six state-of-the-art baselines. The results demonstrate that the proposed algorithm can achieve comparable (and often slightly superior) performance to the best baselines while requiring significantly less training time. The algorithm development and implementation were my contributions, and I also conducted all experiments and subsequent analysis of results. My supervisor, Prof. Mark Coates, provided guidance and feedback about the methodology and the experimental results.

- *Chapter 6 - Conclusions*

In this chapter, the research contributions are summarized. We discuss the research findings and suggest some avenues for future work.

Chapter 2

Background

In this chapter, we provide necessary background material. We first provide a brief review of methods for learning from graphs and data on graphs. We introduce graph notation, provide an overview of graph neural networks, and present some of the state-of-the-art node embedding learning methods in Section 2.1. This is followed in Section 2.2 by an introduction to the problem of recommender systems. We identify categories of recommender systems, describe classic approaches, and briefly review the state-of-the-art. In Sections 2.3 and 2.4 we cover two major techniques used in our proposed model: mutual information maximization and multi-view representation learning. In Section 2.3, we first provide the definition of mutual information and then survey estimation strategies. Some recent applications in representation learning are also presented. In 2.4, we clarify the categorization of multi-view representation learning methods and provide a summary of recent works.

2.1 Learning on Graphs

The research in this thesis focuses on graph-based recommender systems, where interaction and side information data are usually viewed in the form of a bipartite graph or a knowledge graph. Therefore, in this section, we introduce terminology commonly used

in graph analysis. This is accompanied by a description of some of the important work in the field of graph representation learning.

2.1.1 Graphs: Definitions and Notation

Definition

Graphs are widely used to represent data that are associated with interacting or related entities. They have been employed in a wide range of machine learning applications, such as detecting molecular fingerprints from the biological structure [24], recommending friends via social networks [25], and classifying diseases from medical images [26]. The primary task associated with learning on graphs is to generate the graph structure representation which can be exploited by the machine learning models.

Formally, a graph can be described as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the sets of *vertices* and *edges* respectively. An *edge* is defined to be the component that connects a pair of nodes, *i.e.* , $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. For an edge, $e = (v, u)$, v and u are called the *endpoints* of the edge e. If $v = u$, this edge is also called a *self-loop*. The *neighborhood* of a node v is defined as $N(v) = \{u \in V | (v, u) \in E\}$. The $order$ and $size$ of a graph $\mathcal{G}$ are defined as the number of vertices and the number of edges of the graph, respectively.

An efficient way to describe the graph structure is through its *adjacency matrix* $A \in \mathbf{R}^{|V| \times |V|}$. The rows and columns are indexed by vertices and the elements are defined as:

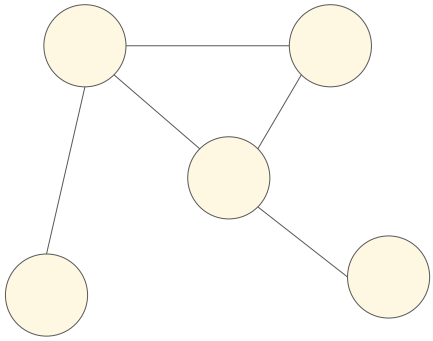
$$\mathbf{A}_{u,v} = \begin{cases} w_{e_{u,v}} > 0 \text{ if } e_{u,v} \in \mathcal{E} \,, \\ 0 \text{ otherwise} \,. \end{cases} \tag{2.1}$$

In this expression, $w_{u,v}$ is the edge weight and it is equal to 1 for unweighted graphs.

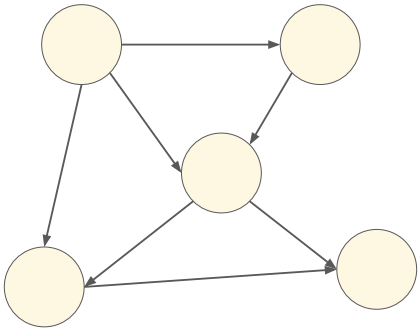
Another widely used matrix to analyze $\mathcal{G}$ is the *degree matrix* $D \in \mathbf{R}^{|V| \times |V|}$. This is a diagonal matrix in which each diagonal entry is equal to the degree of the corresponding vertex v, *i.e.* , $D_{v,v} = d(v)$.

Graph Types

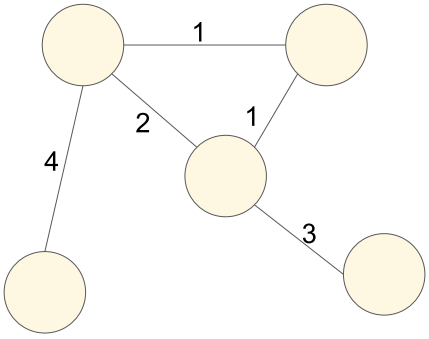
The simplest graph has homogeneous nodes and undirected edges. The representation capability of a simple graph can be extended by including directed edges, assigning weights to the edges, or imposing a specific topological structure. Some examples are shown in Figure 2.1.



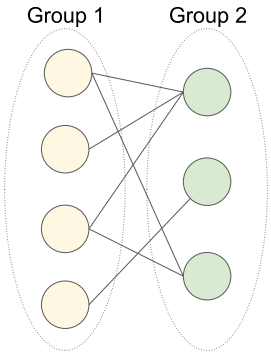
(a) Simple graph



(b) A directed graph



(c) Weighted graph



(d) Bi-partite graph

Figure 2.1: Examples of variants of graphs

In a *directed* graph an edge $e = (u, v)$ denotes the existence of a path from the head u to the tail v and $e = (u, v)$. This does not imply the existence of the edge $e = (v, u)$. Directed graphs are commonly used to construct knowledge graphs. For example, in [1], a model is proposed to propagate information on a knowledge graph with multiple kinds of edges. There is a directed edge from a user node to an item node if the user interacted

with the item. There is a directed edge from an item node to a category node if the item belongs to that category.

In a *weighted* graph, we assign positive, real-valued weights to each edge, and can denote the graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$. Alternatively, we can assign a categorical type to each edge, and represent the graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$. In the context of recommender systems, the set of user-item relationships is often modelled as a *bipartite* graph. In such a graph, there exist two groups of nodes and an edge can only exist if its endpoints belong to different groups. Each vertex in $\mathcal{V}$ corresponds to a user or an item in the dataset, and an edge in $\mathcal{E}$ represents an interaction between a user and an item.

In an *attributed* graph, a feature vector is often associated with each node. Much of the earlier work addressing the task of node embedding learning involved the application of statistical methods. The statistical methods extract feature information from node-level statistics such as the node degree, centrality or clustering coefficient. However, these types of methods have limitations: the hand-engineered features are inflexible and time-consuming to design. Therefore, the methods are difficult to adapt to a deep learning framework.

In the following two sections, we briefly survey an alternative approach to learn node embeddings: *graph representation learning*. Instead of extracting hand-engineered features, the objective of graph representation learning is to learn representations that encode information about the structure of each node in the graph. The representations can be adapted so that they can be incorporated into modern machine learning models. We discuss two major paradigms of graph representation learning approaches that are closely related to our research topic. First, in Section 2.1.2, we introduce the *shallow embedding* approaches [27] where a unique embedding for each node is learned. Then, in Section 2.1.3, we extend the discussion to *graph neural networks* (GNNs), which aim to generate node embeddings through message passing algorithms and neural layers.

2.1.2 Node Embedding Learning

The problem of node embedding can be formalized as follows: Given an undirected, (weighted) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, the objective is to learn a function $f : \mathcal{V} \rightarrow \mathbf{R}^d, d \ll |\mathcal{V}|$ that maps each node to a $d-$dimensional vector while preserving its structural properties. The output representations $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ can then be used in a variety of subsequent tasks such as node classification or link detection.

Many of the node embedding learning methods aim to learn an encoder *ENC* in an unsupervised way [28–30]. The encoder projects nodes into a continuous latent space with a relatively small dimension. The geometric relations in the latent space should reflect the original graph relationships, as illustrated in Figure 2.2.

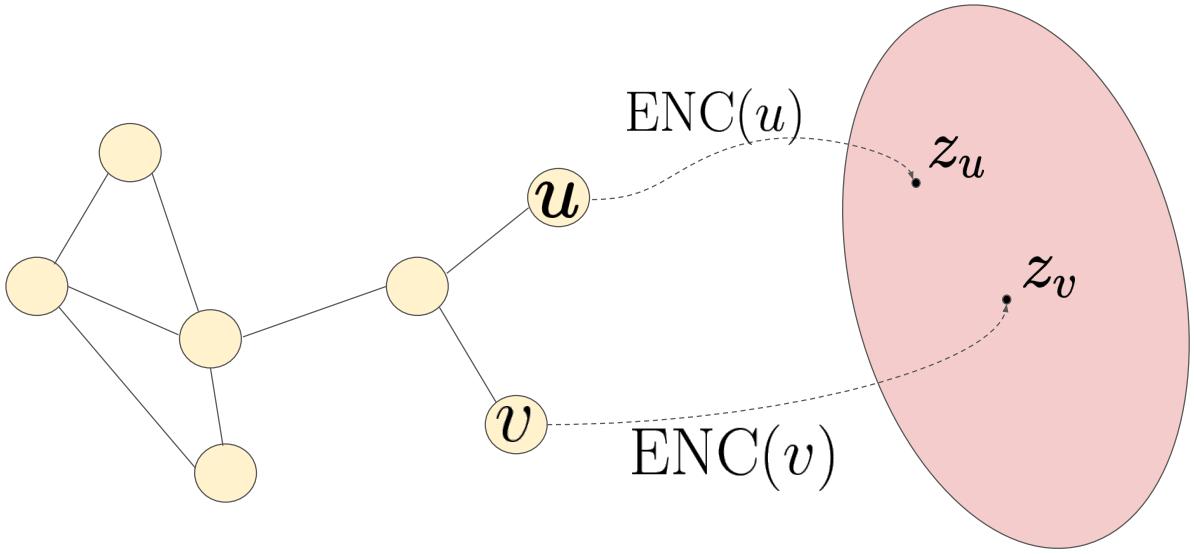


Figure 2.2: Illustration of the node embedding problem. Adapted from [31].

One successful type of node embedding method makes use of random walks on graphs to derive suitable embeddings [30, 32]. In general, these methods first sample a set of random walks with pre-defined length from each node and then apply the Skip-gram algo-

rithm to optimize the node embeddings such that nodes with similar embeddings tend to co-occur on the same walk.

In [31], Hamiltion et al. propose a generalized encoder-decoder framework for random-walk based algorithms where the decoder function and the loss function are defined as follows:

$$\text{DEC}(\mathbf{z}_u, \mathbf{z}_v) \triangleq \frac{e^{\mathbf{z}_u^T \mathbf{z}_v}}{\sum_{v_k \in V} e^{\mathbf{z}_u^T \mathbf{z}_v}} \approx p_{\mathcal{G},T}(v|u)\,, \tag{2.2}$$

$$\mathcal{L} = \sum_{(u,v)\in D} -\log(\text{DEC}(\mathbf{z}_u, \mathbf{z}_v))\,, \tag{2.3}$$

where $p_{\mathcal{G},T}(v|u)$ denotes the probability of visiting v on a random walk with length T starting from u, and D denotes the set of random walks starting from each node. In (2.2), proximity is no longer measured between every pair of nodes but only between adjacent nodes in a sampled walk. Moreover, the decoder output is defined in an asymmetric way which makes the learning more robust. Since the decoder output is approximately equal to a probability, it is natural to use cross-entropy loss to train the decoder, as defined in (2.3). However, directly evaluating (2.2) is computationally expensive, with time complexity $O(|D||V|)$. To efficiently optimize (2.3), certain approximations need to be made.

DeepWalk [32] employs a hierarchical softmax technique to compute the normalizing factor (i.e., the denominator) using a binary-tree structure. On the other hand, *node2vec* [30] approximates the normalizing factor in (2.3) using a noise contrastive approach:

$$\mathcal{L} = \sum_{(u,v)\in D} -\log(\sigma(\mathbf{z}_u^T \mathbf{z}_v)) - \beta \mathbb{E}_{v_n \sim P_n(V)}[\log(-\sigma(\mathbf{z}_u^T \mathbf{z}_v))] \tag{2.4}$$

where v_n denotes the negative sample, $P_n(V)$ is the distribution over the set of nodes V, where uniform distribution is normally used, and $\beta > 0$ is a hyperparameter used to control the contribution of negative samples.

In addition to *DeepWalk* and *node2vec*, Tang et al. propose another successful node embedding algorithm named *LINE* in [33] which is designed to preserve both the local and global structures in the graph. It is conceptually related to *DeepWalk* and *node2vec* in the way that they all use a probabilistic decoder and loss. But instead of measuring similarity from randomly sampled walks, *LINE* explicitly factorizes the first- and second-order proximities with a carefully designed objective. More specifically, the first objective of encoding first-order adjacency information can be defined as:

$$\text{DEC}(\mathbf{z}_u, \mathbf{z}_v) = \frac{1}{1 + e^{-\mathbf{z}_u \mathbf{z}_v}} \approx \mathbf{A}(u, v)\,. \tag{2.5}$$

The second objective of encoding two-hop neighborhood (*i.e.* , the information in $\mathbf{A}^2$) is very similar to (2.2):

$$\text{DEC}(\mathbf{z}_u, \mathbf{z}_v) \triangleq \frac{e^{\mathbf{z}_u^T \mathbf{z}_v}}{\sum_{v_k \in V} e^{\mathbf{z}_u^T \mathbf{z}_v}} \approx p_2(v|u)\,, \tag{2.6}$$

where $p_2(v|u)$ represents the conditional probability of v generated by u. To train (2.6), the goal is to minimize the distance between the estimated distribution $p_2(v|u)$ and the empirical distribution $\hat{p}_2(v|u) = \frac{w_{uv}}{d_u}$, where w_{uv} denotes the weight of the edge (u, v) and d_u is the out-degree of u. KL-divergence is used to measure the distance. The final representation for each node is a concatenation of the outputs from the first and the second objectives.

The node embedding approaches we discussed so far aim to generate a unique embedding vector for each node. It is challenging to apply such approaches to large datasets or dynamic graphs. To alleviate these limitations, researchers have experimented with replacing the shallow encoder with a more complicated encoder [29, 34–36]. This leads to the topic of discussion in the next section: graph neural networks (GNNs).

2.1.3 Graph Neural Networks

Another popular paradigm for learning low-dimensional embeddings of nodes in a graph is the *graph neural network* (GNN), first proposed in [37]. A GNN is a deep neural net-

work defined on graph data and it usually aims to learn the state embedding $\mathbf{h_v} \in \mathbb{R}^s$ for each node. The embedding contains information about a node's neighbourhood through the *message propagation* algorithm, which can be applied in a semi-supervised or self-supervised setting. Moreover, GNNs can directly perform a classification or regression task based on the extracted state embedding, i.e., they can be trained in an end-to-end fashion. This tends to improve efficiency and result in better performance for the target learning task.

Generally speaking, the objective of a GNN is to learn a trainable transition function f shared among all the nodes which takes the input node features and its neighborhood as input. The propagation rule and the output can be defined as follows:

$$\mathbf{h_v} = f(\mathbf{x}_v, \mathbf{x^e}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{x}_{ne[v]}) \,, \tag{2.7}$$

$$\mathbf{o}_v = g(\mathbf{h}_v, \mathbf{x}_v) \tag{2.8}$$

where g is the *local output function*.

In a compact form, the above equations can be expressed as:

$$\mathbf{H} = F(\mathbf{H}, \mathbf{X}) \,, \tag{2.9}$$

$$\mathbf{O} = G(\mathbf{H}, \mathbf{X}_N) \,, \tag{2.10}$$

where F, G, are the stacked versions of the transition functions, output functions, $\mathbf{H}$, $\mathbf{O}$ are the state embeddings and outputs produced and $\mathbf{X}_N$, $\mathbf{X}$ represent the node features and all the features (including node features and edge features), respectively.

The GNNs can also be modified to incorporate different types of graphs or mechanisms [1, 14, 36, 38]. These variants usually follow the same feed-forward neural network setting as in the output step but adopt a different propagation rule to aggregate and update the neighbourhood information. In the rest of this section, we will introduce a few variants related to the context of this thesis.

Graph Convolutional Networks

Graph Convolutional Networks (GCNs) generalize the convolution mechanism in the graph domain and can often be categorized into *spectral* methods [34, 39, 40] and *spatial* methods [36, 41–43]. Given the input features from all nodes in the local neighborhood of node i, the information of the local neighbourhood gets combined over the layers of convolutions, in a similar fashion to the way edge and higher level features are extracted when a classical CNN operates on images.

Spectral methods works with the spectral representation of the graphs. Convolution is defined in the Fourier domain and usually involves computation of the eigendecomposition of the graph Laplacian, $\mathbf{L} = \mathbf{D} - \mathbf{A}$, which contains a significant amount of information about the graph structure. More frequently, the symmetric normalized form of the Laplacian is used in the aggregation step as it ensures that the eigenvectors satisfy some useful properties:

$$\mathbf{L_{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}} \tag{2.11}$$

$$\mathbf{L_{sym}}_{ij} = \begin{cases} 1 \quad \text{if } i = j, \\ -\frac{1}{\sqrt{d_i d_j}} \quad \text{if} (i,j) \in \mathcal{E}, \\ 0, \text{otherwise} \end{cases} \tag{2.12}$$

However, the computation of $\mathbf{L}$ or $\mathbf{L_{sym}}$ is usually expensive with the computational complexity of $O(n^3)$. Moreover, the trained model is restricted to a specific structure defined by the eigenbasis and cannot generalize across graphs. Additionally, spectral-based methods are based on the assumption that the graph is undirected. Therefore, their application is limited.

Spatial methods On the other hand, spatial methods define graph convolutions based on a node's spatial relations. Analogous to convolution in computer vision, the idea of

spatial methods is to convolve the central node's information with its neighbours' information. By stacking multiple convolutional layers, node information is propagated along edges. Such methods enable parameter sharing across nodes and thus increase efficiency and can be used to generate embeddings for nodes that were not observed during training.

One of the challenges of spatial approaches is to define an aggregator which can handle nodes with different degrees while maintaining the weight sharing property. In [42], Duvenaud et al. propose to learn a specific weight matrix for each node degree. Atwood et al. suggest in [44] to consider the message passing process from a probabilistic perspective and assume that the information distribution can reach equilibrium after several rounds. It uses the transition matrix to define the diffusion process where information is transferred from one node to another with a certain probability. On the other hand, Niepert et al. propose in [36] to extract and normalize the neighbourhood to contain a fixed number of neighbours. Hamilton et al. introduce a neighbourhood sampling strategy in [35] to construct a fixed-size neighbourhood for each node.

Graph Recurrent Networks

Inspired by Recurrent Neural Networks (RNNs), another trend is to use the gate mechanism in the propagation step [45,46]. Adapting the GNN framework to employ recurrent units enables usage of the output of intermediate embeddings from the previous layers and thus encourages long-term propagation. A gate is a unit to control information flow and usually is composed of an activation function and element-wise multiplication. Popular gates include Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM). In general, the hidden state computation after inserting the gate mechanism into the graph convolutional layer can be expressed as follows:

$$\mathbf{H}^{(t)} = \sigma(CONV(\mathbf{X}^{(t)}, \mathbf{A}; \mathbf{W}) + CONV(\mathbf{H}^{(t-1)}, \mathbf{A}; \mathbf{U}) + \mathbf{b}), \tag{2.13}$$

where $\mathbf{X}^{(t)} \in \mathbb{R}^{n \times d}$ denotes the feature matrix for n nodes each with d features and $\mathbf{U}$ is the gate unit.

Graph Attention Networks

The attention mechanism assumes that the contributions of neighbouring nodes are neither identical nor pre-determined but learnable. It allows the network to focus on the most relevant part of the neighbourhood of its input and is extremely useful in sequence-based tasks. Variants of Graph Attention Networks (GATs) mainly differ in their definition of the attention scores which are assigned to each neighbour in order to identify the most important part [47–49]. In general, the final hidden state of a node can be expressed as:

$$\mathbf{h}_i = \sigma(\sum_{j \in N_i} \alpha_{ij} \mathbf{W} \mathbf{h_j}) \tag{2.14}$$

Velickovic et al. introduce a model named GAT in [50] which defines a single graph attention layer and stacks multiple such layers together to construct a graph attention network. It uses a multi-head mechanism where K independent attention mechanisms are applied to each node to compute hidden states. The computed states are then combined either through concatenation or element-wise sum to form the final representations. Compared to GAT, Zhang et al. also use the multi-head mechanism but heads are now assigned different weights which are learned through a self-attention mechanism [51].

2.2 Recommender Systems

In this section, we briefly review the relevant background material concerning recommender systems, mutual information maximization, and multi-view representation learning. We first identify a variety of different recommendation scenarios in Section 2.2.1. This is followed by an introduction of the two major categories of existing recommender systems and a discussion of their major differences in Section 2.2.2. We then summa-

rize different types of collaborative filtering techniques in Section 2.2.3. state-of-the-art graph-based recommender systems are presented in Section 2.2.3. Note that the models reviewed in this chapter will focus on implicit data only. However, we discuss models incorporating analysis of item side information in Chapter 3.

2.2.1 Recommendation Scenarios

Recommender systems (RS) have been playing an important role for enhancing efficiency in decision making, especially in this information-overloaded era [1, 6, 14, 15, 52]. Recently, a new paradigm of recommender system named *sequential recommendation* (SRS) has emerged which takes into consideration the temporal order of the interactions. Another paradigm of recommender system which is closely related to SRS and often confused with it is called *session-based recommendation* (SBRS).

The major difference between RS, SBRS and SRS is the input data. RS takes the entire set of interactions as input, while SBRS builts upon session data, and SRS on sequence data respectively. *Session* data systems slice the interactions into several sub-lists with clear boundaries (*e.g.* , interactions of a user within a year, purchased items within a single order). The chronological order of interactions within a sub-list (session) is ignored. By contrast, a *sequence* data approach considers all sequential dependencies of the interactions (*e.g.* , sorted in order of timestamp).

Although the general objective of all types of recommender systems is the same (generate a list of recommendations for a user), there are minor differences between the specification of this goal for the different types. An SRS aims to capture the short-term user interest and provide a timely recommendation according to the evolution of user preferences. An SBRS aims to predict either the unknown part of a session given the known part, *i.e.* , "bundle" prediction, or the next session given the historical sessions, *i.e.* , the next "basket" prediction. In this thesis, we focus on the traditional recommender system scenario where user preference is assumed to be static.

2.2.2 Content-Based Filtering and Collaborative Filtering Recommendations

Categorized by the input data, Collaborative Filtering (CF) and Content-Based Filtering (CB) are the two most basic and popular paradigms for recommender systems. They are demonstrated in Figure 2.3.

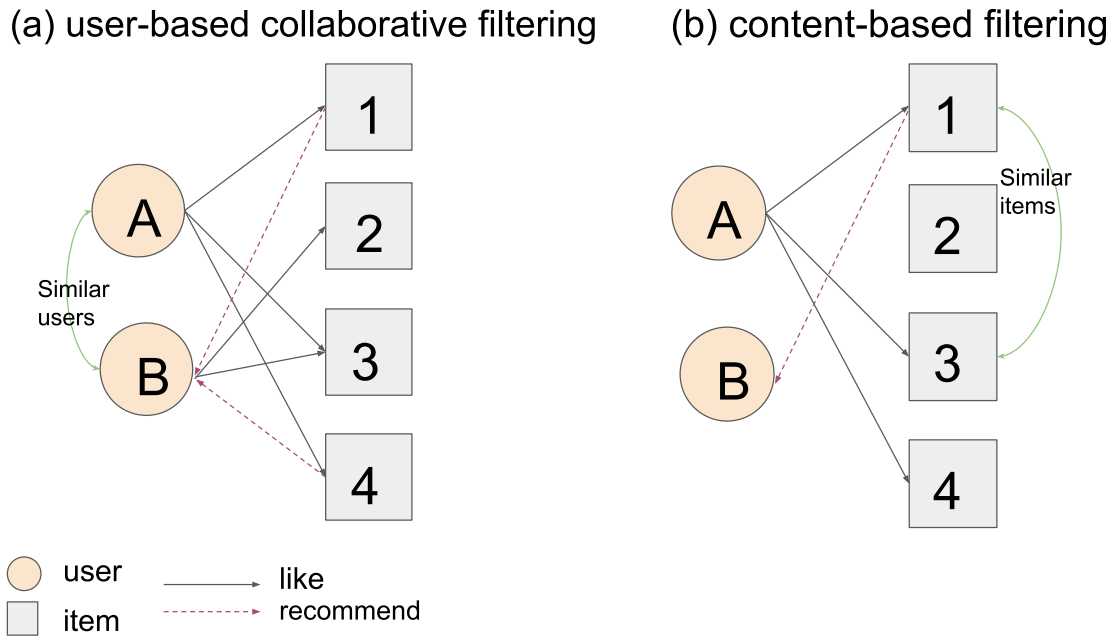


Figure 2.3: Demonstration of the two major recommendation algorithms.

The content-based filtering approach recommends items that are similar to what the user likes based on interaction history or explicit feedback [53–55]. Items are usually represented in a feature matrix, and the goal is to learn user representations in the same feature space. During recommendation, CB methods do not need any information about other users but only the item features and the learned user embedding. Therefore, CB methods are better at recommending cold items which not many users have interacted with. However, since the feature matrix is usually hand-engineered in conventional CB methods, the model's performance is limited by the quality of the engineered features.

Unlike content-based filtering systems which require informative item features, conventional CF-based methods are based on the assumption that users with similar tastes would like similar things. This allows collaborative filtering models to learn a user preference based on the interests of similar users. Consider an e-commerce scenario. A CF-based system usually takes as input the interaction history in the form of a user-item (UI) interaction matrix where each row represents a user and each column represents an item. The feedback about items can either be explicit, in which users explicitly specify how much they are satisfied with the purchase, or implicit, in which the system infers that the user is interested in an item if there is a purchase record. The key task of a conventional CF model is to learn the user and item embeddings and to reconstruct the historical interactions based on the embeddings, *i.e.* , given an item embedding, the goal of the system is to learn an embedding for a user to best explain his/her preference. Analogously, given a user embedding, the system learns the item embedding that best explains the UI matrix. As a result, the embeddings are learnt *collaboratively* from multiple viewpoints, which is how this type of approach is named. The distance between embeddings in the latent feature space is usually used to direct the learning process. In general, we expect that if an item and a user are close in the latent feature space, then it is likely that the user will rate that item highly. Similarly, if embeddings of users are close to each other in the latent feature space, then they are considered to have similar preferences on items.

2.2.3 Collaborative Filtering Techniques

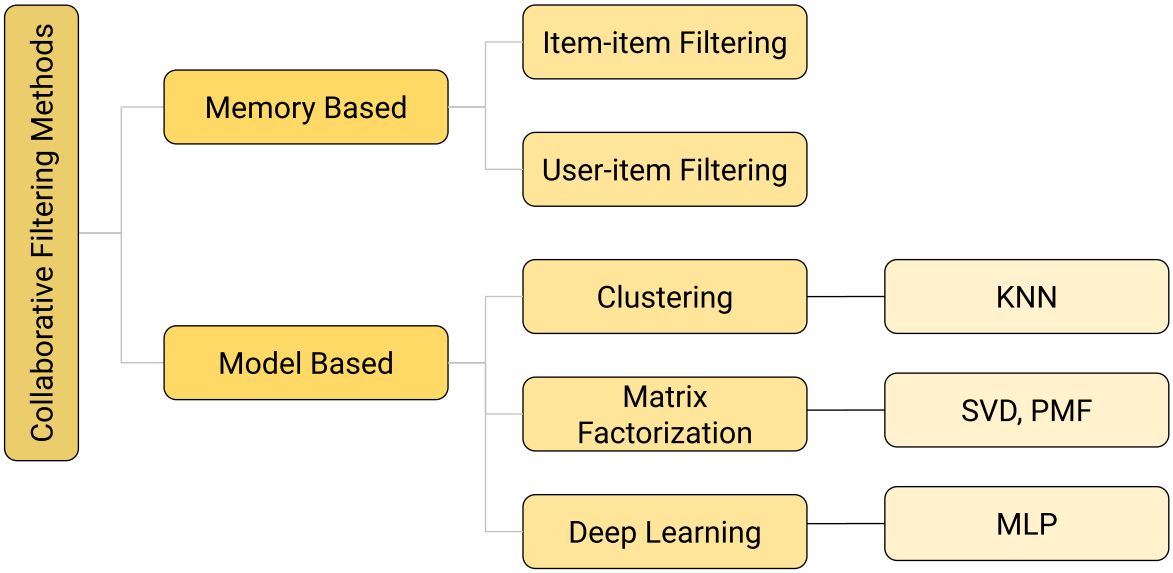


Figure 2.4: Categories of collaborative filtering recommendation models.

Figure 2.4 provides an overview of collaborative filtering techniques. In general, CF techniques can be divided in to two categories: *memory based* and *model based*.

Memory-based approaches formulate recommendations for a user by finding items that similar users like, either based on similarity of ratings or interaction history. Consider user-user similarity as an example. The memory-based approach treats each user's row in the UI matrix as a vector $\mathbf{r}$ and measures the cosine similarity between two users defined as follows:

$$similarity(u, u') = cos(\theta) = \frac{\mathbf{r}_u \cdot \mathbf{r}_{u'}}{\| \mathbf{r}_u \| \| \mathbf{r}_{u'} \|} \tag{2.15}$$

The system can then predict user u's rating of item i by taking the weighted sum of the target item's ratings from all other users and then normalizing it by the sum of the weight:

$$\hat{\mathbf{r}}_{ui} = \frac{\sum_{u'} similarity(u, u') \mathbf{r}_{u'i}}{\sum_{u'} \mid similarity(u, u') \mid} \tag{2.16}$$

The memory-based approach measures the similarity between users or items by pure arithmetic operations and does not learn any user or item embeddings. Therefore, it does not involve any training or optimization step. It is easy to implement and the results are highly explainable. However, when the interaction data is sparse, which is usually the case in the recommendation scenario, the performance of the memory-based approach drops significantly.

On the other hand, model-based recommender systems use machine learning algorithms and can be further categorized into three sub-categories: *clustering-based*, *matrix factorization based* and *deep learning-based*.

Matrix Factorization Methods

The matrix factorization (MF) approach is a popular technique based on the assumption that user preference is determined by a small number of hidden factors [2, 56–58]. In its natural form, matrix factorization characterizes items and users using vectors of factors inferred from item rating patterns, *i.e.* , it decomposes the rating matrix into the low-dimensional user and item latent factors. High correspondence between item and user factors leads to a recommendation. In a song recommendation scenario, for example, each factor in the user embedding measures how much the user likes songs that score high on the corresponding song factor. Formally, given the interaction matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$, the model objective is to factor $\mathbf{R}$ into the user and item embeddings, denoting as $\mathbf{P} \in \mathbb{R}^{|U| \times d}$ and $\mathbf{Q} \in \mathbb{R}^{|I| \times d}$ respectively, whose product is a good approximation of the interaction matrix $\mathbf{R}$:

$$\hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^T \approx \mathbf{R}\,. \tag{2.17}$$

In general, this factorization can be considered as an optimization problem:

$$\min_{\mathbf{P},\mathbf{Q}} \| \mathbf{R} - \mathbf{P}\mathbf{Q}^T \|^2\,, \tag{2.18}$$

where $\| \cdot \|$ denotes the Frobenius norm. In practice, in order to avoid over-fitting, it is common to add regularization terms in the objective function,

$$\min_{\mathbf{P},\mathbf{Q}}(\| \mathbf{R} - \mathbf{P}\mathbf{Q}^T \|^2 + \lambda(\| \mathbf{P} \|^2 + \| \mathbf{Q} \|^2)) \,. \tag{2.19}$$

Matrix factorization methods are highly flexible in dealing with various data as they do not require any hand-engineered feature inputs. However, their disadvantage is also obvious. Since the prediction is the dot product of the corresponding user and item embeddings, the system cannot generate a recommendation for an item if it is not seen before. This is called the *cold-start* issue in the recommendation scenario. To alleviate the cold-start issue, a variety of works have proposed to include side information from additional sources [1, 13, 21, 59].

In [60], Rendle et al. propose a general framework named the *Factorization Machine* (FM) which can model higher-order interactions to find stronger relationships between the latent representations of each feature. Unlike traditional MF models which directly factorize the user-item interaction matrix, the FM represents the interaction as binary vectors of user and item indicators such that in the matrix of training data, each row represents an interaction record with at least two non-zero entries corresponding to the given user-item pair. Moreover, this framework could easily integrate side information by concatenating the training matrix with the auxiliary information matrix. A two-way factorization machine considering up to the second-order interactions generates ratings as follows:

$$f(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \langle \mathbf{v_p}, \mathbf{v_q} \rangle x_p x_q \,, \tag{2.20}$$

where $\mathbf{x}$ is the d-dimensional vector in the (augmented) feature matrix, $\mathbf{w} \in \mathbf{R}^d$ is the weight vector, and $\langle \mathbf{v_p}, \mathbf{v_q} \rangle$ denotes the dot product of two latent feature vectors. He et al. further generalize the FM under a neural network framework in [2]. An advantage of doing so is to relax the linear assumption on the feature interaction which could significantly expand the model expressiveness especially when it comes to larger datasets.

Forbes et al. suggest in [61] to incorporate side information into the MF framework as a linear constraint. Given the content information for items, this method assumes that the item features depend explicitly on the content information:

$$\mathbf{Q} = \mathbf{X}\mathbf{\Phi}\,, \tag{2.21}$$

where $\mathbf{X} \in \mathbb{R}^{n \times k}$ is the item side information and $\Phi \in \mathbb{R}^{k \times d}$ denotes the feature embeddings.

In [62], Zhang et al. propose to project item side information into the same latent factor space as users and items. Each kind of side information is represented by a latent factor matrix. For example, consider a dataset that provides two extra sources of item side information, category and actors, covering n_c categories and n_a actors in total. In this case, a set of item features with two latent factor matrices is constructed: $\mathbf{F} = \mathbf{F_c}, \mathbf{F_a}$ where $\mathbf{F_c} \in \mathbb{R}^{n_c \times d}$ and $\mathbf{F_a} \in \mathbb{R}^{n_a \times d}$ denote the category and actor feature matrices, respectively. The final representation of an item is then a linear combination of its latent representation computed from the interaction matrix, as well as the related item features:

$$\hat{r_{ui}} = \mathbf{P}_u^T(\mathbf{Q}_i + \sum_{t=1}^{T} \frac{\sum_{a \in \mathbf{F}_t(i)} \mathbf{y}_t(a)}{|\mathbf{F}_t(i)|}) + b_{ui}\,, \tag{2.22}$$

where $|\mathbf{F}_t(i)|$ denotes the number of all possible attributes of feature $\mathbf{F_t}$ for item i and $\mathbf{y}_t(a)$ is the d-dimensional vector representing attribute a of feature $\mathbf{F_t}$.

Deep Neural Network Based Method

In contrast to linear models like matrix factorization methods, deep neural network (DNN) based methods to relax the linear assumption which tends to limit the model expressiveness. This is achieved by adding nonlinear activation functions such as *ReLU*, *Tanh*, and *sigmoid*. Due to the flexibility of the input layer design, a deep neural network can easily incorporate side information, which may help in user and item representation learning.

Depending on the architecture design and the mechanism being used, DNN based methods can be further categorized into Multi-Layer Perceptron (MLP) based recommendation, Auto-Encoder (AE) based recommendation, Convolutional Neural Network (CNN) based recommendation, Recurrent Neural Network (RNN) based recommendation, Generative Adversarial Networks (GAN) based recommendation. Hybrid models combine more than one of these deep learning techniques.

Multi-Layer Perceptron (MLP) Based Recommendation A set of methods combine the power of linear models along with MLP. In general, let $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ respectively denote the weight and bias term of the l-th layer. The output of each layer can then be defined as:

$$\mathbf{h}^{(l+1)} = f(\mathbf{W}^{(l)^T}\mathbf{h}^{(l)} + \mathbf{b}^{(l)}), \tag{2.23}$$

where $f(\cdot)$ is the activation function. The network can usually be trained using cross-entropy loss, defined as:

$$\mathcal{L} = \text{argmin} - \log \frac{e^{y_{ui}}}{\sum_j e^{y_{uj}}}, \tag{2.24}$$

where y_{ui} denotes the predicted preference score of user u to item i.

Cheng et al. describe in [63] a deep neural network for YouTube video recommendations where there are two major components: *deep learning* and *wide learning*. The deep learning component adopts MLP to generalize previously unseen feature interactions into low-dimensional embeddings. The wide learning component is a linear model (*e.g.* , a factorization machine) that is used to effectively memorize sparse feature interactions using cross-product feature transformations. Guo et al. further extend the model in [19] by seamlessly integrating a factorization machine with an MLP. It can model the high-order feature interactions via deep neural networks and low-order interactions with factorization machines without tedious feature engineering. In [2], He at al. introduce a bilinear-interaction pooling layer that allows a neural network model to learn more informative feature interactions at the lower level by converting a set of embedding vectors into one vector. This k-dimensional vector is then fed into a stack of fully connected lay-

ers to learn the higher-order interactions between features. Lian et al. further extend [19] by learning the explicit high order feature interactions through a compressed interaction network (CIN) [64].

Auto-Encoder Based Recommendation Another popular paradigm of recommender system design takes advantage of the superiority of auto-encoders in learning underlying feature representations [11, 65, 66]. An auto-encoder system usually consists of an encoder layer, a bottleneck layer and a decoder layer. In general, the encoded vector $\mathbf{z}$ at the bottleneck layer is a salient compressed low dimensional representation of the input data and the objective for an auto-encoder is to reconstruct its input data in the decoder layer. Almost all of the variants (denoting AE, variational AE, connective AE, and marginalized AE) can be applied to the recommendation task (*e.g.* , to fill in the blanks of the UI interaction matrix in the reconstruction layer).

Sedhain et al. extend the vanilla auto-encoder model in [11] to the collaborative filtering setting. It takes as input each row of the UI-interaction matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ as the partially observed user vector where $\mathbf{r}^{(u)} = (\mathbf{R}_{u1}, \ldots, \mathbf{R}_{un}) \in \mathbb{R}^n$, and the encoded vector $\mathbf{z}$ can be represented as:

$$\mathbf{z} = g(\mathbf{V}\mathbf{r}^{(u)} + \mu)\,, \tag{2.25}$$

where $g(\cdot)$ is the activation function. The objective is to solve the following problem:

$$\underset{\theta}{\operatorname{argmin}} \sum_{\mathbf{r}^{(u)} \in \mathbf{S}} \parallel \mathbf{r}^{(u)} - h(\mathbf{r}^{(u)}, \theta) \parallel_2^2 \tag{2.26}$$

where $h(\cdot)$ is a single layer auto-associative neural network. A similar method can also be applied to the column of the interaction matrix to form an item-based learning model.

In [10], Liang et al. propose a model named *MultiVAE* which adapts the variational auto-encoder (VAE) idea to the collaborative filtering setting. The difference between VAE and the vanilla AE is that the encoded input at the bottleneck layer is now a probability distribution of the input data instead of a point estimate. *MultiVAE* replaces the likelihood

function with Bayesian inference; it takes as input the click history $\mathbf{x}_u$ and outputs the corresponding variational parameters μ_u, σ_u^2.

Handling temporal information is crucial for improving the accuracy of VAEs. Sachdeva et al. propose in [67] a recurrent version of *MultiVAE* that passes the consumption sequence subset through a recurrent neural network.

2.2.4 Graph-based Recommendation

In the graph setting, interactions among users and items can naturally be modeled as edges in a bipartite graph. In the past decade, GNNs have been intensively studied in the field of recommender system. As introduced in Section 2.1.3, the GNN framework often consists of two stages: 1) graph feature extraction; and 2) link prediction. In recommender system settings, the stages can be translated into: 1) item and user representation learning; 2) user-item relationship inference/prediction. In the rest of this section, we review and compare [68], [38], [69], [14], and [15] from the following five aspects:

- item and user representation learning
  - message construction
  - message aggregation
  - higher-order message propagation
  - final representation generation
- user-item relationship prediction

In a GNN, the hidden representation of a node $\mathbf{h_u^{(k+1)}}$ is an aggregation of a node's embedding as well as the message $\mathbf{m}_{N(u)}^{(k)}$ aggregated from the node's local neighbours. These are also based on information aggregated from their respective neighbourhoods,

as illustrated in Figure 2.5. Hamilton et al. generalize such a process in [31] as:

$$\begin{aligned}\mathbf{h_u^{(k+1)}} &= \text{UPDATE}^{(k)}(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}(h_v^{(k)}, \forall v \in N(u))) \\ &= \text{UPDATE}^{(k)}(\mathbf{h}_u^{(k)}, \mathbf{m}_{N(u)}^{(k)})\end{aligned} \tag{2.27}$$

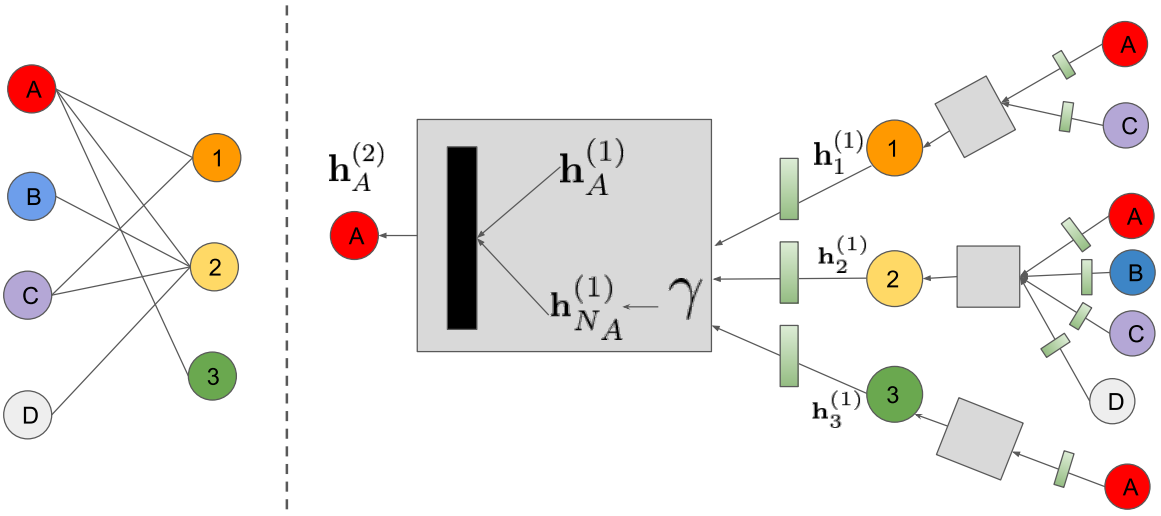


Figure 2.5: An illustration of the message passing framework. Left: An example user-item interaction bipartite graph. Right: A two-layer neural message passing model that computes the embedding of node A. Diagram adapted from [38]

Message Construction Let $\mathbf{h_i}$ and $\mathbf{m}_{u \leftarrow i}$ denote the initial feature vector of node i and the message passed from node i to node u respectively. There are two main message construction approaches: linear transformation and non-linear transformation. Berg et al. propose a user rating specific message construction method in [68]:

$$\mathbf{m}_{u \leftarrow i,r} = \frac{1}{\sqrt{|N_i||N_u|}} \mathbf{W_r h_i}\,, \tag{2.28}$$

where $\frac{1}{|N_i||N_u|}$ is the normalization constant and $|N_i|$ denotes the number of neighbors of node i. $\mathbf{W_r}$ is the rating specific parameter matrix which linearly transforms the initial embedding.

Comparing to [68], Wang et al. further encode the interaction between nodes in [69] via the element-wise product within the message:

$$\mathbf{m}_{u\leftarrow i} = \frac{1}{\sqrt{|N_i||N_u|}}(\mathbf{W_1}\mathbf{h_i} + \mathbf{W_2}(\mathbf{e}_i \odot \mathbf{e}_u)) \,, \tag{2.29}$$

where $\mathbf{W_1}, \mathbf{W_2}$ are the trainable weight matrices to encode useful information within the original feature vector.

Ying et al. apply a dense neural network in [38] to construct the message in an non-linear way:

$$\mathbf{m}_{u\leftarrow i} = (\mathbf{Q}\mathbf{h}_i + \mathbf{q}) \,, \tag{2.30}$$

where Q is a weight matrix shared among all nodes.

He et al. suggest in [14] that the two popular components in GCN: feature transformation and non-linear activation can be removed since, in the recommendation scenario, the initial feature vectors are either one-hot IDs or randomly initialized. Sun et al. argue in [15] that message transformation should be dependent on the type of entity:

$$\mathbf{m}_{u\leftarrow i} = (\mathbf{Q_v}\mathbf{h}_i + \mathbf{q_v}) \tag{2.31}$$

$$\mathbf{m}_{i\leftarrow u} = (\mathbf{Q_u}\mathbf{h}_u + \mathbf{q_u}) \tag{2.32}$$

Besides the messages contained in the bipartite graph, Sun et al. also propose in [15] that two other types of messages should be passed during the propagation phase: the collaborative signals among the same type of entity $\mathbf{m}_{uu} = (\mathbf{M_u}\mathbf{h}_u)$, $\mathbf{m}_{ii} = (\mathbf{M_i}\mathbf{h}_i)$, and the initial node features h_i.

Message Aggregation After the message is computed, the next step is to generate an aggregated neighborhood information $\mathbf{m}_{N(u)}^{(k)}$ to update each node's embedding. The su-

perscript distinguishes between embeddings at different layers of the GNN. Two types of methods are commonly used: concatenation and summation. The summation can be further categorized into simple element-wise sum and weighted sum. For weighted sum, the importance of each message is determined by a normalization factor, *e.g.* , $\alpha_{ij} = \frac{1}{\sqrt{|N_i||N_u|}}$, or $\alpha_{ij} = \frac{1}{\sqrt{|N_i|}}$ (left normalization). More complex attention mechanisms are also incorporated in several proposed systems, e.g., [15, 70]. The simple element-wise sum can be considered as a special case of weighted sum where the normalization factor is equal to 1. An activation function and simple transformation are usually applied afterwards to make the aggregated messages more robust. Considering the embedding after the k-th ($k \geq 1$) propagation layer, $\mathbf{h}_i^{(k)}$, as an example, Berg et al. simply accumulate (either concatenate or sum depending on the dataset) all the neighborhood messages [68]:

$$\mathbf{h}_i^{(k)} = \mathrm{ReLU}(\mathbf{W} \cdot \mathrm{ReLU}[\mathrm{ACCUM}(\sum_{j \in N(i,1)} \mathbf{m}_{i \leftarrow j,1}, \ldots, \sum_{j \in N(i,R)} \mathbf{m}_{i \leftarrow j,R})]) \,, \tag{2.33}$$

where $\mathbf{W}$ denotes the transformation matrix shared with all nodes.

Compared to [68], Sun et al. and Ying et al. additionally encode the embedding of the node at the previous layer along with the neighborhood information in [15] and [38] as follows:

$$\mathbf{h}_i^{(k)} = \mathrm{ReLU}(\mathbf{W} \cdot \sigma[\mathrm{CONCAT}(\mathbf{h}_i^{k-1}, \sum_{j \in N(i)} \mathbf{m}_{i \leftarrow j})] + \mathbf{q}) \,. \tag{2.34}$$

He et al., as described in [14], directly take the weighted sum of the neighborhood messages without any transformation or non-linear activation:

$$\begin{aligned} \mathbf{h}_i^{(k)} &= \sum_{j \in N(i)} \frac{1}{|N_i||N_j|} \mathbf{m}_{i \leftarrow j} \\ &= \sum_{j \in N(i)} \frac{1}{|N_i||N_j|} \mathbf{h}_j^{(k-1)} \end{aligned} \tag{2.35}$$

Instead of directly incorporating a node's embedding from the previous layer in the accumulation, in [69], Wang et al. first transform it:

$$\begin{aligned}\mathbf{h}_i^{(k)} &= \text{LeakyReLU}(\mathbf{m}_{i\leftarrow i} + \sum_{j\in N(i)} \mathbf{m}_{i\leftarrow j}) \\ &= \text{LeakyReLU}(\mathbf{W}_1\mathbf{h}_i^{(k-1)} + \sum_{j\in N(i)} \mathbf{m}_{i\leftarrow j})\,.\end{aligned} \tag{2.36}$$

Higher-Order Message Propagation Higher-order messages can be very informative when it comes to understanding user preference. Such connectivity can be modelled in a GNN by stacking embedding propagation layers. By stacking L layers, a node is capable of receiving messages from its L-hop neighbours. Intuitively, embeddings from different layers capture different information. For example, embeddings from the first layer smooth users and items with direct interaction. Embeddings on the second layer enforce smoothness among users and items that have overlapping interactions. Except for [68], all other papers we have reviewed in this section stack at least two layers to obtain high-order connectivity semantics. Note that stacking too many layers can be harmful to the system's performance. A layer size of two or three is most commonly reported to have the best performance.

Final Representation Generation The final representation of each node is usually obtained by either using the output directly from the last propagation layer or by combining outputs from each layer.

In [38], Ying et al. form the final embedding by feeding the last layer output through a fully connected layer:

$$\mathbf{h}_i^* = \mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1\mathbf{h}_i^{(L)} + \mathbf{q})\,. \tag{2.37}$$

On the other hand, Wang et al. propose in [69] to concatenate the representation learned by different layers to get the final representation:

$$\mathbf{h}_i^* = \mathbf{h}_i^{(0)} \parallel \ldots \parallel \mathbf{h}_i^{(L)}\,. \tag{2.38}$$

Moreover, He et al. describe in [14] to generate the final embedding by adding the embeddings from each layer:

$$\mathbf{h}_i^* = \sum_{l=0}^{L} \alpha_l \mathbf{h}_i^{(l)} , \tag{2.39}$$

where α_l is the importance score of the l-th layer embedding. Specifically, $\alpha_l = \frac{1}{L+1}$ is reported to yield the best performance in [14].

In [15], the final representation is derived by combining multiple sources of information:

$$\mathbf{h}_i^* = \text{ACCUM}[\mathbf{h}_i^{(L)}, \mathbf{s}_i, \mathbf{z}_i] , \tag{2.40}$$

where $\mathbf{s}_i = \sigma(\mathbf{W} \cdot \mathbf{h}_i^{(0)})$ preserves most of the information contained in the initial embedding and $\mathbf{z}_i$ contains the collaborative information from the same type of entity.

User-Item Relationship Prediction The inner product between the final representations of the target user and an item is the most popular method for estimating a user's preference on item i:

$$\hat{y}_{ui} = \mathbf{h}_u^{*T} \mathbf{h}_i^* . \tag{2.41}$$

Items with the highest (K) preference scores are then recommended.

2.3 Mutual Information

Mutual Information Maximization (MIM) is a key concept closely related to the proposed method in this thesis and it has been intensively studied in the field of feature selection, *i.e.* , representation learning. Therefore, in this section, we first give a brief introduction to mutual information, as well as the intuition behind MIM. We then proceed to introduce state-of-the-art methods to estimate mutual information.

2.3.1 Introduction

Shannon information To start with, Mutual information (MI) is closely related to the concept of *Shannon information*. In information theory, the amount of information contained in an event can be calculated using the occurrence probability of such event, which is called the *Shannon information*, aka. "*self-information*" of an event. Formally, given a discrete random variable X, with possible outcome events $A_X = \{x_1, \cdots, x_n\}$, and its corresponding probability $p(\mathbf{x}) = \{p(x_1), \cdots, p(x_n\})$, Shannon information of an outcome x_i is defined to be [71]:

$$h(x_i) = -\log p(x_i) \tag{2.42}$$

when a base-2 logarithm is used, this can simply be interpreted as the number of bits required to represent this event. Since probability of an event occurs is always less than or equals to 1, $p(x_i) \leq 1$, $h(x)$ is always positive or zero.

entropy Another important concept used in the definition of mutual information is *entropy*. Entropy compute the average amount of Shannon information within X, and can be also interpreted as the uncertainty of X, denoted as $H(X)$. Its definition is given as [71] :

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{2.43}$$

where the sum is applied over X's possibility space.

Conditional entropy *Conditional entropy* quantifies the amount of information needed to describe an event given the information of another event. Given two discrete random variables X and Y, the entropy of X conditioned on Y is denoted as $H(X|Y)$ and can be computed as:

$$H(X|Y) = -\sum_{x \in A_X, y \in A_Y} p(x, y) \log \frac{p(x, y)}{p(\mathbf{x})} \tag{2.44}$$

where A_X, A_Y are the set of possible events of X and Y.

Kullback-Leibler (KL-) divergence Also known as the *Relative entropy*, the *KL-divergence*, D_{KL}, is a metric measuring the discrepancy between two probability distributions. Specif-

ically, given two probability distributions P and Q defined on a probability space Ω, and know that P is discrete with respect to Q, the KL-divergence between the two distributions P and Q can be defined as:

$$D_{KL}(P||Q) := \sum_{x \in \Omega} p(\mathbf{x}) \log(\frac{p(\mathbf{x})}{Q(x)}) = \mathbb{E}_P[\log \frac{p(\mathbf{x})}{Q(x)}] \,, \tag{2.45}$$

where $\mathbb{E}_P[\cdot]$ is the expectation with respect to P. A KL-divergence of 0 indicates that the given pair of distributions share the same amount of information, *i.e.* $P = Q$. Also note that KL-divergence is an asymmetric metric, interchanging P and Q in general gives a different result: $D_{KL}(P||Q) \neq D_{KL}(Q||P)$. Therefore, although KL-divergence is sometimes called the " KL-distance", it is not strictly a distance.

Mutual Information In the case when we care more than just the information contained in a single variable but also the amount of shared information between two random variables, mutual information (MI) comes in handy. MI measures how much uncertainty about a random variable is reduced, given knowledge about the other variable. Formally, let X and Y be two random variables defined in the same space $\Omega \in \mathbb{R}^d$, MI between X and Y, $I(X;Y)$, can then be defined as [72]:

$$I(X;Y) := H(X) - H(X|Y) \tag{2.46}$$

$I(X;Y)$ can also be interpreted as KL-divergence between the joint distribution, $p(x,y)$, and the product of the marginal distributions, $p(\mathbf{x})p(\mathbf{y})$, considering $x \in X, y \in Y$ are the samples from each random variables. Let $p(\mathbf{x}|\mathbf{y})$ denote the conditional distribution of x given y. Mutual information between x and y is:

$$\begin{aligned} I(x;y) &= D_{KL}(p(x,y)||p(\mathbf{x})p(\mathbf{y})) \\ &= \sum_{x \in X} p(x,y) \log \frac{p(\mathbf{x}|\mathbf{y})}{p(\mathbf{x})} \end{aligned} \tag{2.47}$$

From the probability theory, we know that if two random variables are independent, then $P_{XY} = P_X P_Y$. Therefore, if two random variables are independent, the KL-divergence between their joint and their product of the marginals is zero [72], *i.e.* mutual information between these two variables is zero.

Mutual Information Maximization When applied in representation learning, the intuition behind maximizing mutual information (MIM) is clear: given a pair of encoded representations, if their mutual information is maximized, there is then a guarantee on the amount of shared information being encoded in both representations. Depending on the final goal, this maximization can be done over representations extracted from different layers of the same neural network [22], or output of different networks as long as it is describing the same object [21].

2.3.2 Mutual Information Estimation

Unfortunately, MI estimation in high-dimensional spaces is intractable [73]. Therefore, one often resorts to maximizing a tractable lower bound of the mutual information estimator in practice. Three estimators are commonly used [21, 22, 73, 74]: the Donsker-Varadhan (DV) estimator [75], the Jensen-Shannon Divergence (JSD) estimator [76], and the Information Noise Contrastive Estimator (InfoNCE) [77].

We first outline the general setting and notations for maximizing mutual information between two representations. Let $\mathbf{x}$ and $\mathbf{y}$ be the target representation vectors; we aim to maximize the mutual information between them. $p(\mathbf{x})$, $p(\mathbf{y})$ and $p(\mathbf{x}, \mathbf{y})$ are then the corresponding marginal and joint distributions. $p(\mathbf{x}|\mathbf{y})$ is the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$.

The Donsker-Varadhan Estimator [75]: This estimator follows a variational formulation named the *Donsker-Varadhan(DV) representation* which provides a dual representation of the KL-divergence as a supremum over a set of functions. Formally, the Donsker-

Varadhan (DV) representation of the KL-divergence is defined as follows:

$$D_{KL}(p(\mathbf{x})||p(\mathbf{y})) = \underset{T:\Omega\to\mathbb{R}}{\text{SUP}} \mathbb{E}_{p(\mathbf{x})}[T] - \log \mathbb{E}_{p(\mathbf{y})}[e^T]\,, \tag{2.48}$$

where the supremum is taken over all functions $T : \Omega \to \mathbb{R}$ such that the two expectations are finite. A straightforward consequence of this dual representation is that a lower bound of the KL-divergence can be derived:

$$D_{KL}(p(\mathbf{x})||p(\mathbf{y})) \geq \mathbb{E}_{p(\mathbf{x})}[T] - \log \mathbb{E}_{p(\mathbf{y})}[e^T]\,. \tag{2.49}$$

From (2.47) and (2.49), the lower bound on the MI between $\mathbf{x}$ and $\mathbf{y}$, obtained from the DV representation, is:

$$\begin{aligned} I(\mathbf{x};\mathbf{y}) &:= D_{KL}((p(\mathbf{x},\mathbf{y})||p(\mathbf{x})p(\mathbf{y})) \\ &\geq I(\mathbf{x};\mathbf{y})^{DV} = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[T(\mathbf{x},\mathbf{y})] - \log(\mathbb{E}_{p(\mathbf{x})p(\mathbf{y})}[e^{T(\mathbf{x},\mathbf{y})}])\,. \end{aligned} \tag{2.50}$$

The Jensen-Shannon Estimator [76]: An alternative approach is to estimate MI based on the Jensen-Shannon divergence (JSD), which is a symmetrized and smoothed version of the KL-divergence. Formally, the JSD between two distributions $p(\mathbf{x})$, $q(\mathbf{x})$ is defined as:

$$D_{JSD}(p(\mathbf{x})||p(\mathbf{y})) = \frac{1}{2}D_{KL}(p(\mathbf{x})||m(\mathbf{x},\mathbf{y})) + \frac{1}{2}D_{KL}(p(\mathbf{y})||m(\mathbf{x},\mathbf{y})) \tag{2.51}$$

where $m(\mathbf{x},\mathbf{y}) = \frac{1}{2}(p(\mathbf{x}) + p(\mathbf{y}))$. In [76], Novozin et al. formulated a MI estimator based on the JSD:

$$I(\mathbf{x};\mathbf{y}) \geq \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[-\text{sp}(-T(\mathbf{x},\mathbf{y}))] - \mathbb{E}_{p(\mathbf{x})p(\mathbf{y})}[\text{sp}(T(\mathbf{x},\mathbf{y}))]\,, \tag{2.52}$$

where $\text{sp}(x) = \log(1 + e^x)$ is the soft-plus function. In [22], Hjelm et al. show that the JSD between the joint distribution and the product of the marginals has a monotonic relationship with the KL-divergence. The distributions with the highest MI also have the highest JSD.

The InfoNCE Estimator [78] InfoNCE is a contrastive loss function used for self-supervised learning. NCE is the initialization of Noise Contrastive Estimation [76]. NCE was first used as a lower bound of MI in [78]. The basic idea of InfoNCE is to use the mutual information between a pair of representations (*e.g.* , x and y) as the training signal and maximize this MI to achieve the representation learning objective. Since we can not directly control the joint distribution $p(\mathbf{x}, \mathbf{y})$, the objective is to maximize the ratio:

$$f(x, y) \propto \frac{p(\mathbf{x}|\mathbf{y})}{p(\mathbf{x})}\,, \tag{2.53}$$

where $\propto$ (proportional to) is used since the ratio is an unbounded value. In [78], this ratio is approximated by the following function:

$$f(\mathbf{x}, \mathbf{y}) = \exp(T(x, y))\,, \tag{2.54}$$

where $T(x, y) : \Omega \to \mathbb{R}$ is a suitable function. For example, we can set $T(x, y) = \mathbf{x}^T\mathbf{W}\mathbf{y}$, with $\mathbf{W}$ being a trainable transformation matrix. The InfoNCE loss is then defined as:

$$\begin{aligned} I(\mathbf{x}, \mathbf{y})^{(\text{NCE})} &= \mathbb{E}_{p(\mathbf{x})}[\log \frac{f(\mathbf{x}, \mathbf{y})}{\sum_{\bar{\mathbf{x}}} f(\bar{\mathbf{x}}, \mathbf{y})}] \\ &= \mathbb{E}_{p(\mathbf{x})}[T(x, y) - \mathbb{E}_{p(\bar{x})}[\log \sum_{\bar{x}} e^{T(\bar{x}, y)}]]\,, \end{aligned} \tag{2.55}$$

where $\bar{x}$ is a negative sample drawn from the same space as x. For example, in the image classification task, if x is a representation of a *cat* image, then a possible negative sample $\bar{x}$ can be the representation of an image from a different class (*e.g. dog*).

From the results reported in [22], the JSD estimator is found to be the most stable, while the InfoNCE estimator can potentially provide a better outcome if many negative samples are provided. However, the performance difference between these two estimators becomes smaller in large datasets. Overall, both the JSD and InfoNCE estimators can be used when computing the precise value of the mutual information is not the objective.

2.3.3 Mutual Information Maximization Solutions in Representation Learning

In the past five years, mutual information maximization has been established as one of the state-of-the-art strategies for representation learning in multiple domains, including computer vision and recommender systems. These methods usually maximize the mutual information between pairs of embeddings. The pair can consist of embeddings extracted from different layers of a neural network. Alternatively, the pair can comprise one real and one corrupted embeddings. Another option is for the embeddings to be derived from the same context but with different views.

Formally, given an object $\mathbf{X}$ and its two different views $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, the objective can be expressed as follows:

$$\underset{g_1, g_2}{\operatorname{argmax}} \hat{I}(g_1(\mathbf{X}^{(1)}), g_2(\mathbf{X}^{(2)}))\,, \tag{2.56}$$

where $\hat{I}(\cdot)$ is a sample-based MI estimator, and g_1, g_2 are the encoders used to extract features from the specified perspective.

In the case when the objective is to maximize the MI instead of knowing its precise value [21–23, 79], a JSD-based estimator can be used which may offer some favourable trade-off.

In the application of image classification, the basic version of [22] specifically sets $X^{(1)}$ to be the entire image, $X^{(2)}$ to be an extracted image patch, and g_1 and g_2 to be the outputs from different layers of the same convolutional network. The objective can thus be interpreted as maximizing the mutual information between local (patch) features and global (entire image) features. Following a similar setup, [79] enhances the method of [22] by using different augmentations of the same image as $X^{(1)}$, $X^{(2)}$. In a further extension, [78] passes a sequence of image patches with a fixed order to an encoder and takes the aggregated representations from the first t patches as $X^{(1)}$ and the representation from the $t+k$-th patch as $X^{(2)}$. This leads to a location-sensitive prediction. In [80], Tian et al. gen-

eralize and extend [78] to a multi-view setting (*e.g.* , multiple image channels) where [78] can be considered as a special case with only two views.

Mutual information also proves useful when assessing the correlation between different views of data in the recommender system setting. Zhou et al. propose a pre-training method in [21] for sequential recommendation in which mutual information is maximized between two views of the item, one being derived from the item attributes and one from the sequence of items. Sankar et al. present a model in [81] for group recommendation where MI is measured between group members and the group. The estimated MI serves as the weight to control the contribution of a user's preference to the group preference. In [82], Cao et al. argue that existing bipartite graph representation learning methods mainly focus on modelling local structural properties and ignore the importance of encoding global properties. To alleviate this issue, the authors propose to generate global views of the graph by aggregating embeddings of homogeneous nodes, *i.e.* , the user nodes and item nodes. An InfoMax-based loss function [22] is to maximize the mutual information between the global view and the local view which is the representation generated by aggregating information from k-hops neighbours.

2.4 Multi-View Representation Learning

The top-K recommendation problems focus on analyzing user preferences hidden in interaction history, explicit feedback, user profiles, and item attributes. These data can be considered as multiple distinct views with complementary information. Intuitively, combining information from these views in an effective manner should improve the quality of the learned representations for the target objects, which is crucial to system performance. Therefore, it is natural to organize input data into multiple views and apply multi-view representation learning techniques to improve the overall performance. In fact, with the increasing availability of multi-modal data collected from different sources, multi-view

representation learning has been extensively studied for a variety of machine learning applications, including in the recommender system domain.

2.4.1 Preliminaries

In general, multi-view representation learning aims to learn from data from distinct views which do not share a common feature space. A modality or a view is defined as data coming from heterogeneous sources or captured by different techniques. For example, for a piece of music, audio signals, lyrics and other side information such as the performer, the producer etc, are considered three views or three modalities. These multi-view data usually have different statistical properties which make it difficult to incorporate all of the information in a unified framework.

Canonical Correlation Analysis (CCA) One of the earliest techniques that studies relationships between two views is CCA, first introduced in [83]. Given two views of the data $\mathbf{X} = \mathbf{X}, \ldots, \mathbf{x_n}$, $\mathbf{Y} = \mathbf{y_1}, \ldots, \mathbf{y_n}$, CCA strives to find two linear transformations $\mathbf{W_X}$, $\mathbf{W_Y}$ such that the correlations between the corresponding transformed variables are maximized. The correlation coefficient is defined as:

$$\begin{aligned} \rho &= \text{corr}(\mathbf{W_X}^T\mathbf{X}, \mathbf{W_Y}^T\mathbf{Y}) \,, \\ &= \frac{\mathbf{W_X}^T\mathbf{C_{XY}}\mathbf{W_Y}}{\sqrt{(\mathbf{W_X}^T\mathbf{C_{XX}}\mathbf{W_X})(\mathbf{W_Y}^T\mathbf{C_{YY}}\mathbf{W_Y})}} \,, \end{aligned} \tag{2.57}$$

where $\mathbf{C_{XY}}$ denotes the covariance matrix defined by:

$$\mathbf{C_{XY}} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x_i} - \mu_\mathbf{x})(\mathbf{y_i} - \mu_\mathbf{y})^T \,. \tag{2.58}$$

Here $\mu_\mathbf{x} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x_i}$, $\mu_\mathbf{y} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y_i}$ are the means of the two views, respectively. Despite its great success, the CCA method does not take into consideration the nonlinearities of multi-view data, *i.e.* it can not model the high order association among the multi-view data.

Inspired by the recent progress in deep neural network techniques, numerous works have proposed extensions of CCA [84], and endeavoured to define better association methods beyond CCA [85]. In [86], Li et al. classify existing methods into two categories: multi-view representation alignment and multi-view representation fusion. In the rest of this section, we briefly introduce these two categories and review a collection of applications in the recommender system setting.

2.4.2 Multi-View Representation Fusion

The multi-view representation learning models presented in this section are mainly based on the *complementary* principle. This principle states that there exists some view-exclusive knowledge underlying multiple views which can be exploited to improve the representation quality [87]. These methods aim to fuse representations learned from separate views into a single compact vector. Suppose we have two views $\mathbf{X}$ and $\mathbf{Y}$. The objective is to find a compact representation $\mathbf{h}$ such that complementary knowledge in multiple views is fully captured [86]. We define

$$\mathbf{h} = \phi(\mathbf{x}, \mathbf{y}) \,, \tag{2.59}$$

where ϕ is the feature fusion method. Taking two-view fusion for example, suppose that $\mathbf{x}$ and $\mathbf{y}$ are the two mid-level features learned from views $\mathbf{X}$ and $\mathbf{Y}$, respectively. Three simple yet widely used fusion operations are: taking the maximum value (*i.e.* , $\mathbf{h} = \max\{\mathbf{x}, \mathbf{y}\}$), calculating the average (*i.e.* , $\mathbf{h} = \frac{1}{2}(\mathbf{x} + \mathbf{y})$) and forming the concatenation (*i.e.* , $\mathbf{h} = [\mathbf{x}; \mathbf{y}]$).

In [88], Zhang et al. construct three views based on heterogeneous information from three sources: textual review, visual image, and numerical rating. The user and item representations obtained from each view are concatenated and a pair-wise contrastive ranking loss is applied on the merged representations to train the model. Guan et al. apply an additional transformation layer in [89] on the concatenated representation to reduce its dimension. To reduce the impact of user bias, a user-specific trainable vector is also con-

catenated with the transformed content representation to form the final comprehensive item representation. In [90], Liang et al. argue that the aforementioned fusion methods have neglected the fact that some views may contain more information than others. The authors propose a solution by performing weighted summation over the views, where the attention scores are computed by passing the summation of multi-view features through an MLP.

2.4.3 Multi-View Representation Alignment

In contrast to multi-view representation fusion methods, multi-view representation alignment methods follow the *consensus* principle [87] and perform alignment between corresponding representations learned from each view such that maximum agreement is achieved on distinct views. Given two views $\mathbf{X}$ and $\mathbf{Y}$ of the dataset, and let $\mathbf{x_i}$, $\mathbf{y_i}$ represent the corresponding pair of representations of an object i extracted from the two views. The general goal can then be formulated as:

$$\operatorname{argmin} \sum_{i=1}^{n} M(\mathbf{x_i}, \mathbf{y_i}) \,, \tag{2.60}$$

where $M(\cdot)$ denotes the chosen metric measuring the distance (or dissimilarity) between the given pair of inputs. For example, the aforementioned CCA method chooses $M(\cdot)$ to be the correlation measurement. Li et al. define a distance-based metric involving transformation matrices in [91]. The alignment method aims to find linear transformation matrices $\mathbf{W_x}$, $\mathbf{W_y}$ such that the distance between the two representations, after transforming to the same space, is minimized. The dissimilarity metric is:

$$M(\mathbf{x_i}, \mathbf{y_i}) = ||\mathbf{x_i}^T \mathbf{W_x} - \mathbf{y_i}^T \mathbf{W_y}||_2^2 \tag{2.61}$$

From the multi-view representation learning perspective, the major difference across distinct methods are the view construction methods and their choice of the relevance measurement metric. Common choices are cosine similarity, Euclidean distance, and correlation-based metrics such as MI. In particular, Elkahky et al. introduce in [92] a content-based recommender system in the web search scenario with one user view and three item views. The user view contains a collection of user features generated from users' search queries and clicked URLs. Each of the three item views has features of apps, news, movies/TVs that the users have viewed/interacted with, respectively. A user-item view pair is then matched through the Microsoft user IDs to leverage the information across domains and achieve a better user representation. The goal is to maximize the overall cosine similarity between features from the user view and each of the item views.

One challenge of multi-view representation learning is that one of the views may have low-quality data. In [93], Cai et al. develop an active learning algorithm to progressively train a mapping function that transforms the rich visual features of each video to the more informative but often missing text features. The transformed text feature is then used to predict the user preference score and the objective is to minimize the difference between the score computed from the transformed text feature and the original text feature. Furthermore, in [21], Zhou et al. quantify the correlations among different views by mutual information.

2.5 Summary

In this chapter, a thorough background review is presented to cover the four topics that are most relevant to this thesis. As described later in Chapter 5, our proposed method is a **recommender system** solution that incorporates item side information following a **multi-view** representation learning framework. The collaborative signal between users and items is explored using a state-of-the-art **graph neural network** method and **mutual information** is used as an objective during training. Therefore, each of the four sections

in this chapter provides fundamental material corresponding to one of the keywords associated with the proposed method. In the following chapter, we review how side information is injected into existing recommender system algorithms to improve recommendation accuracy.

Chapter 3

Related Work

The sparsity of datasets has been a major challenge for real-world recommender systems. In this chapter, we review recent methods which strive to alleviate this issue by using side information. More specifically, these methods aim to encode side information into the users' and items' embeddings. There are many solutions to incorporate side information. These solutions vary in how they represent side information, how they model feature interactions, and how they incorporate the side information in the user/item representations.

We first provide a detailed description of algorithms that are highly related to this thesis. These are used as baselines in our subsequent experiments. We then discuss how these methods design the interactions between side information and items.

The notations used throughout this chapter are listed in Table 3.1.

Table 3.1: Notation Used in Chapter.3

Notation	Description
$\mathcal{U}$	the set of users
$\mathcal{I}$	the set of items
$\mathcal{C}$	the set of attributes
$\mathbf{U} \in \mathbb{R}^{\lvert\mathcal{U}\rvert \times d}$	matrix of latent user representations
$\mathbf{I} \in \mathbb{R}^{\lvert\mathcal{I}\rvert \times d}$	matrix of latent item representations
$\mathbf{C} \in \mathbb{R}^{\lvert\mathcal{C}\rvert \times d}$	matrix of latent attribute representations
$\mathbf{R} \in \mathbb{R}^{\lvert\mathcal{U}\rvert \times \lvert\mathcal{I}\rvert}$	binary user item interaction matrix
$\mathbf{S} \in \mathbb{R}^{\lvert\mathcal{I}\rvert \times \lvert\mathcal{C}\rvert}$	binary item attribute indicator matrix [1]
$\mathbf{V} \in \mathbb{R}^{\lvert\mathcal{I}\rvert \times d}$	the aggregated attribute embedding matrix
$\mathbf{e}_u$	latent representation of user u; the u-th row of $\mathbf{U}$
$\mathbf{e}_i$	latent representation of item i; the i-th row of $\mathbf{I}$
$\mathbf{e}_c$	latent representation of attribute c; the c-th row of $\mathbf{C}$

[1] Each row is a multi-hot vector with 1 indicating the attribute an item is associated with, and 0 otherwise.

3.1 Baseline Models

MultiGraph Convolution Collaborative Filtering (MGCCF) [15]: MGCCF is a state-of-the-art model that explores the user-item correlations through graphs. Overall, this model contains three sub-graphs: a user-item (UI) bipartite graph representing user-item interactions, a user-user and an item-item graph capturing proximity among users and items. The overall architecture is illustrated in Fig. 3.1.

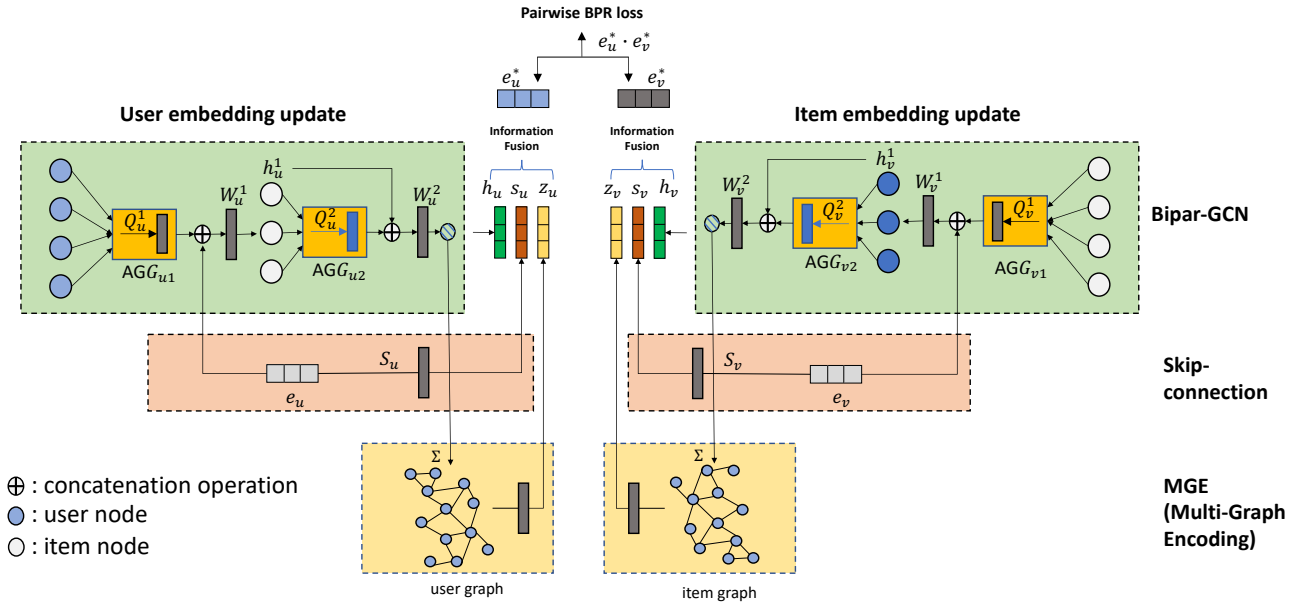


Figure 3.1: Illustration of the MGCCF framework, diagram adapted from [15].

The most basic graph is the UI bipartite graph. As introduced in Section 2.1, there are two types of nodes in a bipartite graph (in this case user nodes and item nodes), and an edge can only exist between nodes of different types. The UI graph contains an edge between a user node and an item node if there was an interaction between them. To explore the UI bipartite graph, Sun et al. propose a *Bipartite Graph Convolutional Neural Network (Bipar-GCN)* that iteratively aggregates the K-hop neighbourhood information of a user or item node through graph convolution [15]. Specifically, given the initial user embedding $\mathbf{e}_u$ (or item embedding $\mathbf{e}_i$) extracted from the maintained user embedding matrix $\mathbf{U}$ (or an item embedding matrix $\mathbf{I}$), the following information propagation rule is applied:

$$\mathbf{h}_u^k = \sigma(\mathbf{W}_u^k \cdot [\mathbf{h}_u^{k-1}; \mathbf{h}_{N(u)}^{k-1}]), \mathbf{h}_u^0 = \mathbf{e}_u \,, \tag{3.1}$$

where $\mathbf{h}_u^k$ denotes the k-th layer embedding of the target user u, $[;]$ represents the concatenation operation, $\sigma(\cdot)$ is the *tanh* activation function, $\mathbf{W}_u^k$ denotes the transformation

weight matrix in k-th layer of the Bipar-GCN, and $\mathbf{h}_{N(u)}^{k-1}$ is the aggregated neighborhood information matrix of $k-1$-hop neighbors. To ensure the neighborhood information aggregation process is permutation invariant, an element-wise weighted mean aggregator is applied:

$$\mathbf{h}_{N(u)}^{k-1} = \sigma(\text{MEAN}(\mathbf{h}_i^{k-1} \cdot \mathbf{Q}_u^k, i \in N(u))), \tag{3.2}$$

where MEAN denotes the element-wise mean operation and $\mathbf{Q}_u^k$ is the (user) aggregator weight matrix at the $k-$th layer. Note that both $\mathbf{W}_u^k$ and $\mathbf{Q}_u^k$ in the Bipar-GCN are shared across all user nodes at layer k. Analogously, item embeddings can be generated following the same propagation and aggregation rule.

To fully investigate the given user-item interactions and to capture information left out by the Bipar-GCN, Sun et al. propose to construct a user-user graph and an item-item graph from the UI interaction history and explore them using the proposed *Multi-Graph Encoding (MGE)* layer. In particular, the user-user graph and the item-item graph are first constructed by computing pairwise cosine similarities on the rows or columns of the UI interaction matrix, $\mathbf{R}$. For each node, the 10 most similar users or items are then considered as neighbours. The neighbourhood information is then aggregated through a simple graph convolution layer. For a target user u, the embedding generated from the MGE layer is computed by aggregating information from the 1-hop neighbours through a weighted sum:

$$\mathbf{z}_u = \sigma(\sum_j \in N'(u) e_j \cdot \mathbf{M}_u). \tag{3.3}$$

Here $N'(u)$ denotes the 1-hop neighbours of user u, e_j is the initial user embedding from the maintained user embedding matrix $\mathbf{U}$, and $\mathbf{M}_u$ is the learnable weight matrix shared across all users. Item embeddings can be generated from the item-item graph through MGE following the same propagation and aggregation rule.

Moreover, Sun et al. argue that information contained in the initial user and item embeddings are diluted during propagation. To alleviate this issue, a skip-connection mech-

anism is added which generates another set of user (or item embeddings) by passing the initial embedding $\mathbf{e}_u$ (or $\mathbf{e}_i$) through a single fully-connected layer.

The final embedding is generated by fusing the corresponding embeddings from these three components through element-wise sum, concatenation or an attention-based summation operation, out of which the element-wise sum is reported to achieve the best performance in [15].

This model exploits multiple graphs to explicitly explore user-item, user-user and item-item relationships and can be easily adapted to incorporate item attributes by changing the item-item graph construction method. For example, instead of computing cosine similarity on the columns of the UI interaction matrix, we can compute cosine similarity on the rows of the item-attribute indicator matrix $\mathbf{S}$. Attribute information can thus be naturally injected into the system, *i.e.* , neighbours in the item-item graph are now items that share similar attributes.

KGAT [1]: Instead of explicitly constructing multiple graphs to represent different relations between entities (users, items, attributes), another strategy is to encode all the information into a unified graph. In [1], Wang et al. proposed to construct a directed graph with labelled edges where each user, item, and attribute is modelled as an entity node and an edge connecting a user and an item represents an existing interaction while an edge between an item and an attribute indicates that the attribute is related to the item. Different relations, such as "directed by", "genre", "acted by", can be identified through the label given to the edge. Let $\mathcal{U}$, $\mathcal{I}$, $\mathcal{C}$ denote the set of users, the set of items and the set of attributes, respectively. The collaborative knowledge graph (CKG) $\mathcal{G}$ can then be represented as $\mathcal{G} = \{(h, r, t)|(h, t) \in \mathcal{E}, r \in \mathcal{R}'\}$ where $\mathcal{E} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{C}$ is the set of all entities and $\mathcal{R}'$ is the unified set of all the relations, including user-item interactions and item-attribute associations. The overall structure of KGAT is given in Figure.3.2

First, a *CKG embedding layer* is applied to specifically train the node embeddings, $\mathbf{U}$, $\mathbf{I}$ and $\mathbf{C}$. Let $\mathbf{e}_h$, $\mathbf{e}_t$ denote the **h**ead and **t**ail embeddings of a given triplet (h, r, t). The aim of the embedding layer is to optimize the node embeddings such that the translation

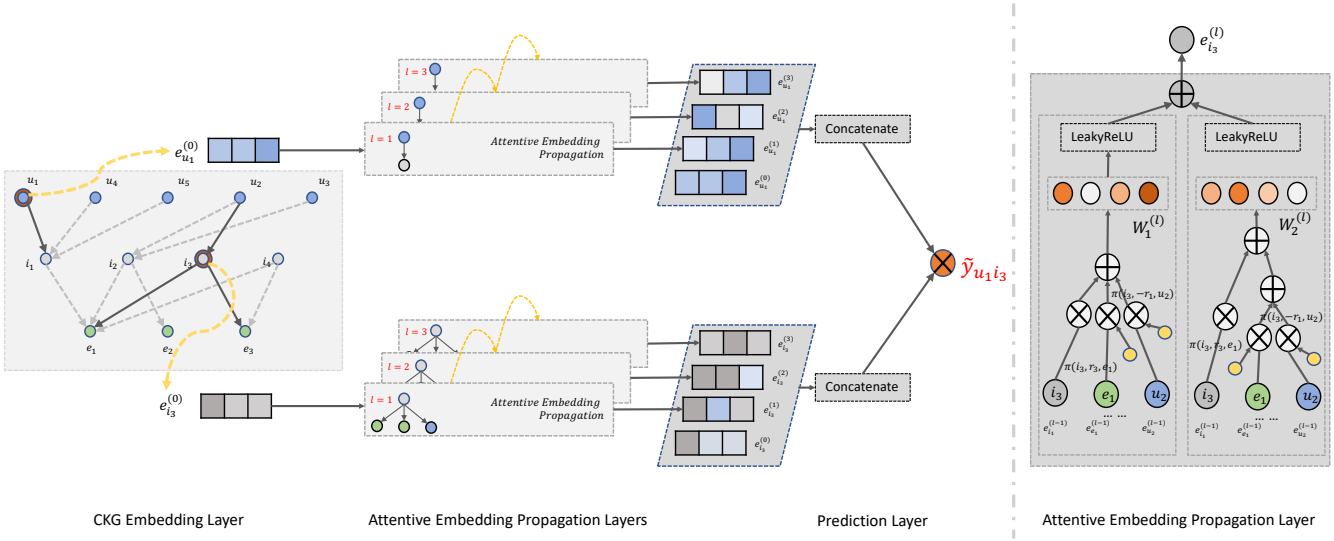


Figure 3.2: Illustration of the KGAT framework (diagram adapted from [1]). Left: The overall architecture of KGAT; Right: Detailed illustration of the attentive embedding propagation layer.

relation in the graph is preserved: $\mathbf{e}_h^r + \mathbf{e}_r \approx \mathbf{e}_t^r$, where $\mathbf{e}_h^r = \mathbf{W}_r\mathbf{e}_h$, $\mathbf{e}_t^r = \mathbf{W}_r\mathbf{e}_t$ are the head and tail embeddings projected into the relation r's space and $\mathbf{W}_r$ is the projection matrix for relation r. Following this translation principle, the plausibility score for each triplet (h, r, t) can be computed as:

$$g(h, r, t) = ||\mathbf{W}_r\mathbf{e}_h + \mathbf{e}_r - \mathbf{W}_r\mathbf{e}_t||_2^2 . \tag{3.4}$$

The first objective of KGAT is then to minimize the plausibility scores for triplets that truly exist in $\mathcal{G}$ while maximizing the scores for the "broken" ones that do not exist in $\mathcal{G}$. The loss is thus:

$$L_{\text{KG}} = \sum_{(h,r,t,t')\in\mathcal{T}} -\ln \sigma(g(h, r, t') - g(h, r, t)) , \tag{3.5}$$

where $\mathcal{T} = \{(h, r, t, t')|(h, r, t) \in \mathcal{G}, (h, r, t') \notin \mathcal{G}\}$. Each broken triplet is constructed by replacing the tail entity in an existing triplet (h, r, t) with t' sampled randomly from the entity set.

The first objective ensures that the direct relations in the knowledge graph are encoded in the node embeddings. A natural goal that comes next is to propagate information in each node along the edges. KGAT achieves this through the *attentive embedding propagation layers*. Specifically, KGAT employs a new aggregator, f, to explicitly explore two types of interactions between each node and its neighbourhood representations. The aggregator is:

$$f(\mathbf{e}_h, \mathbf{e}_{N(h)}) = \text{LeakyReLU}(\mathbf{W}_1(\mathbf{e}_h + \mathbf{e}_{N(h)})) + \text{LeakyReLU}(\mathbf{W}_2(\mathbf{e}_h \odot \mathbf{e}_{N(h)}))\,, \tag{3.6}$$

where $\mathbf{e}_h$ and $\mathbf{e}_{N(h)}$ represent the embedding and the aggregated neighborhood embedding of the target entity h respectively. $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable weight matrices and $\odot$ denotes the element-wise product operation. The neighborhood message of node h, $\mathbf{e}_{N(h)}$, is computed through a weighted summation of its neighbors' embeddings, where the weight is dependent on the distance between the two nodes in the relation space, *i.e.* , closer entities contribute more in the neighborhood aggregation stage.

Multiple propagation layers can be stacked to explore higher-order connectivity information. For example, if the target entity is a user node of user u, then its representation generated from the $l-$th step, $\mathbf{e}_h^{(l)}$, is represented as:

$$\mathbf{e}_h^{(l)} = f(\mathbf{e}_h^{(l-1)}, \mathbf{e}_{N(h)^{(l-1)}}), \quad \mathbf{e}_h^{(0)} = \mathbf{e}_u\,. \tag{3.7}$$

Finally, representations from all L layers are concatenated to formulate the final representation of a target node, denoted as $\mathbf{e}_u*$ for user node u and $\mathbf{e}_i*$ for item node i. The preference score of user u for item i is computed through an inner product:

$$y(u, i) = \mathbf{e}_u * \cdot \mathbf{e}_i * \ . \tag{3.8}$$

To train this system such that a higher preference score is assigned to an observed user-item pair, the BPR [57] loss is used, which leads to KGAT's second objective:

$$L_{CF} = \sum_{(u,i,j)\in\mathcal{O}} -\log\sigma(y(u,i) - y(u,j))\,, \tag{3.9}$$

where $\mathcal{O} = \{(u,i,j)|(u,i) \in \mathcal{R}^+, (u,j) \in \mathcal{R}^-\}$ is the training set, with $\mathcal{R}^+$ denoting the positive (observed) interactions, and $\mathcal{R}^-$ denoting the sampled negative (unobserved) interactions.

Combining the aforementioned two objectives, the final objective of KGAT is:

$$L = L_{\text{KG}} + L_{\text{CF}} + \lambda||\Theta||_2^2 \tag{3.10}$$

NFM [2]: In neural factorization machines (NFMs), the user interaction history and item knowledge are encoded in a sparse feature matrix. Given the user interaction history consisting of a total of m interaction records from $|U|$ users over $|I|$ items, and the item knowledge indicating which attributes each item is associated with, for $|I|$ items over $|A|$ attributes, the feature matrix is defined as: $\mathbf{X} \in \mathbb{R}^{m\times n}$ where $n = |U| + |I| + |A|$. Each row, $\mathbf{x}$, in the feature matrix is a sparse binary vector representing an interaction record, with $\mathbf{x}_i = 1$ indicating the existence of the i-th feature. The feature set $V = \{\underbrace{\mathbf{e}_u^1, \cdots, \mathbf{e}_u^{|U|}}_{\text{user features}}, \underbrace{\mathbf{e}_i^1, \cdots, \mathbf{e}_i^{|I|}}_{\text{item features}}, \underbrace{\mathbf{e}_a^1, \cdots, \mathbf{e}_a^{|A|}}_{\text{attribute features}}\}$ is a combination of all the features associated with users, items and attributes. Given a sparse vector $\mathbf{x} = [x_1, \cdots, x_n]$ as input, a set of feature embedding vectors can then be constructed as $\mathbf{V}_x = \{x_1\mathbf{v}_1, \cdots, x_n\mathbf{v}_n\}$. Based on $\mathbf{V}_x$, the NFM outputs a prediction score $y_{\text{NFM}(\mathbf{x})}$ and the goal is to train NFM such that the prediction score is high if $\mathbf{x} \in \mathbf{X}$. In [2], NFM's predictive model is formulated as:

$$\hat{y}_{\text{NFM}}(\mathbf{x}) = w_0 + \sum_1^n w_i x_i + \mathbf{h}^T\sigma_L(\mathbf{W}_L(\cdots\sigma_1(\mathbf{W}_1 f_{\text{BI}}(\mathbf{V}_\mathbf{x}) + \mathbf{b_1})\cdots) + \mathbf{b}_L)\,, \tag{3.11}$$

where $W = \{w_0, \cdots, w_n\}$ is a set of learnable weights. In particular, the first term is the global bias and the second term models the weight of each feature. $\mathbf{h}$ in the third term are the neuron weights of the prediction layer and $\{\mathbf{W}_1, \mathbf{b}_1, \cdots, \mathbf{W}_L, \mathbf{b}_L\}$ are the weights and biases of the hidden layers. The bi-interaction layer, f_{BI}, is a key contribution of NFM. It was introduced to model the second order feature interactions, including attribute-item interactions. Specifically, the bi-interaction layer is defined as follows:

$$f_{\text{BI}}(\mathbf{V}_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j \,, \tag{3.12}$$

where $\odot$ is the element-wise product operation.

The objective of NFM when used in the ranking task is:

$$L = -\ln \sigma(\hat{y}_{\text{NFM}}(\bar{\mathbf{x}}) - \hat{y}_{\text{NFM}}(\mathbf{x})) = -\ln \frac{1}{1 + e^{\hat{y}_{\text{NFM}}(\mathbf{x}) - \hat{y}_{\text{NFM}}(\bar{\mathbf{x}})}} \tag{3.13}$$

cVAE [94]: As both user ratings and item attributes are associated with items, Chen et al. propose that a variational autoencoder (VAE) can be trained to recover both the interaction matrix $\mathbf{R}$ and the item-attribute indicator matrix $\mathbf{S}$ *collectively*. The approach is thus named CVAE. In short, a variational autoencoder is just an autoencoder, make up of both an encoder and a decoder whose objective is to minimize the reconstruction error between the encoded-decoded data and the initial data, but with some regularisation on the latent space to avoid overfitting. Therefore, instead of encoding an input as a single representation vector, the goal is now to encode it as a distribution over the latent space. Specifically, the encoder in VAE is referred to as the *inference network* and the decoder is referred to as the *generation network*.

Let f_ϕ and f_θ be the inference network and the generation network of a VAE parametrized by ϕ and θ, respectively. The goal of CVAE is to recover $\mathbf{R}$ and $\mathbf{S}$ one-by-one through the same VAE network:

$$\begin{aligned} &\mathbf{M} \sim f_\phi(\mathbf{R}), \mathbf{R} \sim f_\phi(\mathbf{M}), \\ &\mathbf{Z} \sim f_\phi(\mathbf{S}), \mathbf{S} \sim f_\phi(\mathbf{Z}), \end{aligned} \tag{3.14}$$

where $\mathbf{M}$, $\mathbf{Z}$ are the distributions over the latent space of $\mathbf{R}$ and $\mathbf{S}$ generated by the inference network. The objective is to train the inference network such that the distribution can be approximated.

In order to do so, Chen et al. first assume that the latent variables in $\mathbf{M}$, $\mathbf{Z}$, denoted as $\mathbf{m}$ and $\mathbf{z}$, follow a Gaussian distribution. Next, the interaction history of user j over all items is assumed to follow a Bernoulli distribution:

$$\begin{aligned} &\mathbf{r}_j|\mathbf{m}_j \sim Bernoulli(\sigma(f_\theta(\mathbf{m}_j)))\,, \\ &\log p_\theta(\mathbf{r}_j|\mathbf{m}_j) = \sum_{i\in\mathcal{I}} r_{ji}\log\sigma(f_{ji}) + (1-r_{ji})\log(1-\sigma(f_{ji}))\,, \end{aligned} \tag{3.15}$$

where $\sigma(\cdot)$ is the sigmoid function, f_{ji} denotes the $i-$th element of vector $f_\theta(\mathbf{m}_j)$ and r_{ji} is the interaction history of user j to item i. Note that $\sigma(f_{ji})$ is within $(0, 1)$.

To distinguish between $\mathbf{R}$ and $\mathbf{S}$, for side information, Chen et al. assume the feature indicator of item j over all attributes follows a Gaussian distribution:

$$\begin{aligned} &\mathbf{s}_j|\mathbf{z}_j \sim N(f_\theta(\mathbf{z}_j), \mathbf{I})\,, \\ &\log p_\theta(\mathbf{s}_j|\mathbf{z}_j) = \sum_{i\in\mathcal{I}} -\frac{1}{2}(s_{ji} - f_{ji})^2\,, \end{aligned} \tag{3.16}$$

where f_{ji} denotes the $i-$th element of vector $f_\theta(\mathbf{s}_j)$, and $\mathbf{I}$ is the identity matrix.

To generate recommendations for a target user u, a latent variable is first computed as $\mathbf{m}_j = f_\phi(\mathbf{r}_j)$. This is then decoded through the inference network as $f_\theta(\mathbf{m}_j) \in \mathbb{R}^{\mathcal{I}}$. Items are then ranked in descending order of the $f_\theta(\mathbf{m}_j)$ values for the recommendation.

To train this network, the cVAE is first pre-trained by feeding it item attributes and then refined by feeding the user ratings. When item attributes are fed, the objective is to maximize the log-likelihood:

$$L = \sum_{j\in\mathcal{I}} (\log p_\theta(\mathbf{s}_j|\mathbf{z}_j)) \tag{3.17}$$

Similarly, when user ratings are fed, the objective then becomes:

$$L = \sum_{j \in \mathcal{U}} (\log p_\theta(\mathbf{r}_j | \mathbf{m}_j)) \tag{3.18}$$

In the next section, we provide a discussion of the side information exploration process in the aforementioned algorithms along with other highly related non-baseline methods from the perspective of how side information is incorporated into the system.

3.2 Discussion

In this section, we identify three major challenges in side information incorporation and discuss how each algorithm tackles these challenges one by one.

***Side information Representation*:** The first challenge is to effectively represent the side information. A simple way is to use an attribute indicator matrix, $\mathbf{S}$, where each row is a multi-hot vector with 1 indicating an attribute an item is associated with, and 0 otherwise. Some approaches choose to use the binary indicator matrix $\mathbf{S}$ directly [2, 94, 95]. In [96], Zheng et al. propose to learn a weighted version of $\mathbf{S}$ where the contributions from each feature factor can be varied. However, the nature of $\mathbf{S}$ makes it generally sparse. A more common way to represent side information is to use latent feature representations [3, 21, 59]. For example, Chen et al. propose to maintain a randomly initialized latent feature representation matrix $\mathbf{C}$ which projects the high-dimensional multi- or one-hot vectors of item attributes to low-dimensional dense representations [59]. $\mathbf{C}$ is then trained using a top-n recommendation objective in cross-entropy form. In [1], each feature in the side information is represented as a node in a knowledge graph, and $\mathbf{S}$ is used to create links between these feature nodes and the item nodes. Radford et al. choose to use a transformer to encode the multiple feature representations into a single vector [3] .

***Side Information Interaction*:** The second challenge in incorporating side information is to handle the feature interaction. Specifically, for items with more than one category

tag, it must be determined how the multiple features can be combined. Early methods mostly consider linear interactions only. In [96], Zheng et al. sum the weights associated with each feature. Similarly, Chen et al. propose in [59] to conduct an element-wise summation over the related feature representations. Recently, some researchers have argued that there exist higher-order interactions between the features that cannot be captured by linear operations. In [2], He et al. suggest to replace the inner product with an non-linear neural network. Specifically, they develop a "bi-interaction" pooling layer which converts a set of embedding vectors into a single vector by computing an element-wise product between each pair of user/item/feature representations and summing over the results. Higher-order interactions can then be achieved by stacking multiple propagation layers.

***Side Information Incorporation*:** The third challenge is to infuse side information into user or/and item representations. In some works, researchers simply aggregate the two representations learned from user ratings/interactions and side information, either by summation [59, 97] or concatenation [35]. A preference score is then calculated based on the dot product of the aggregated user and the item embeddings. The preference score features in the cross-entropy or the pair-wise rank loss for optimization.

However, these types of methods tend to ignore the data heterogeneity between user ratings and side information. To cater for the heterogeneity of ratings and side information, Chen et al. connect side information with the interaction data by learning both representations using the same neural network [94]. A variational autoencoder (VAE) is first pretrained to reconstruct the attribute indicator matrix $\mathbf{S}$. The same autoencoder is then fine-tuned with an objective that focuses on reconstructing the user-item interaction matrix $\mathbf{R}$.

On the other hand, Chen et al. proposed to alleviate this issue by adding an additional objective specifically for exploring item-attribute interactions. This objective is based on the item tagging task [59], which aims to predict correct tags (or attributes) for items. This is analogous to how traditional recommender systems aim to predict potential items

for users. In particular, [59] is trained to compute the probability of each attribute being associated with an item by passing the attribute representation through a two-layer fully connected neural network.

The bi-linear layer proposed in [2] takes into account the interactions between users, items and side information through an element-wise product operation.

Besides the model-based methods, graph-based methods are also gaining increasing attention. As described in [21], Zhou et al. propose to incorporate a multi-head self-attention block that takes all the item and item attribute embeddings as input. The block is designed to extract a selective combination of information from different representation spaces. In [1], Wang et al. introduce a bi-interaction aggregator to connect each item and its aggregated attribute information, as reviewed in section 3.1. By considering the dimension-wise feature interaction between each node embedding and its corresponding neighbourhood representation, this aggregator can effectively propagate more information from similar entities.

3.3 Summary

In this chapter, we presented the works most relevant to the topic of this thesis. We identified three key stages in side information enhanced recommender systems and discuss different types of solutions in terms of their design in each stage. From the literature review, we see that state-of-the-art methods usually build upon different frameworks (*e.g.* GNN-based, autoencoder-based, matrix factorization-based, *etc*) to explore the correlation between items and their side information. These various formulations make it difficult to compare the intrinsic difference in terms of how side information is being absorbed into the system. Moreover, to the extent of our knowledge, major research focus has been put on developing more accurate recommender systems based upon existing state-of-the-art models but little effort has been given to truly understand the side information

exploration process hidden within the basic framework. For these reasons, in the next chapter, we propose a solution to provide a unified description of these methods.

Chapter 4

Side Information Enhanced Methods in A Unified Framework

Over the past decade, an increasing amount of new information beyond implicit user feedback has become available as a result of the development of data collection technologies. The information can be categorized into two types: rich side information related to users and items, and information associated with interactions between users and items. Incorporating side information in CF algorithms is emerging as a promising direction in recommender system research [13, 20, 21, 94, 95, 98]. In Chapter 3, we provided a review of a series of key algorithms developed to exploit side information, including matrix factorization-based methods [2, 99], auto-encoder based approaches [94, 96], and graph neural network-based algorithms [1, 95]. However, these methods are developed based on different base models, motivated from different perspectives and usually employ diverse notations. Therefore, it is difficult to compare the characteristics of existing successful side information enhanced methods. For example, [94] is developed on top of an auto-encoder framework, whereas the key components in [1, 95] are graph neural networks (GNNs). [1] is developed to analyse the user-item-attribute knowledge graph, whereas [13] is presented from a multi-view perspective. To better analyze and compare these various side information enhanced approaches, we propose a framework which de-

scribes these methods from a unified perspective: the *multi-view alignment* perspective and makes it considerably easier to perceive the similarities and differences between the approaches.

In this chapter, we first present the novel framework named ***a**ttribute-**i**tem **a**lignment* (AIA) developed for item side information enhanced algorithms. Subsequently, we show how some of the prominent algorithms fit into the introduced framework. Note that in this chapter, the term "side information" is used to describe side information in general, including side information about users and items; "item side information" and "attribute" are used interchangeably to refer to side information describing features of items. In special cases where information from other sources is used, we specifically identify the information source.

4.1 Overview of the AIA Framework

Our framework is motivated by the intuition that item side information and UI interaction history can be considered as two views of items that capture different aspects of the items. The former focuses on describing the items' attributes while the latter focuses on how users perceive and interact with the items. If we can learn to encode the complementary information from the multi-view data in a low-dimensional embedding, then in principle, the resultant item embeddings should contain more information that is helpful for downstream recommendation tasks.

Recommender system incorporating item side information can be considered as having two blocks in their algorithms: (i) a block for handling user-item interactions, where in general a user-item preference score is computed and the score for the correct pair is optimized; (ii) a block for exploring the item-attribute relations. The AIA framework is designed to analyze the second block.

As illustrated in Figure 4.1, there are primary two key components in the AIA framework:

- an **association measure function**, $f : \mathcal{R}^d \times \mathcal{R}^d \rightarrow \mathcal{R}$, that measures the association between an item and its corresponding attribute(s);
- an **association loss function**, $L : \mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}$, that evaluates the quality of the association function in order to train the model.

Depending on the system design, there can also be an **attribute aggregator** that is designed to aggregate all of the attributes associated with an item if it has multiple attributes. However, this component is optional. In the rest of this chapter, we explain each component in detail.

4.1.1 The Association Measure Function

In principle, the association function aims to predict the association score of an attribute-item pair given an item embedding $\mathbf{e}_i$ as well as an attribute embedding $\mathbf{e}_c$. The goal is to output a higher score for a "positive" pair, i.e., where the item is indeed related to the given attribute while doing the opposite for a "negative" pair. For instance, the association score for the {*Dove body wash*, *Skin Care*} pair should be higher than the {*Dove body wash*, *Makeup Tools*} pair.

The association measure function is defined to have the following signature:

$$s = f_\theta(i, c) \tag{4.1}$$

where s is the association score, i is the item and c is the attribute. Depending on the design of the model, an association function may or may not contain trainable parameters θ.

4.1.2 The Attribute Aggregator (Optional)

In many cases, an item is related to more than one attribute. Instead of computing association score for each attribute-item pair, another option is to pass in a set of related

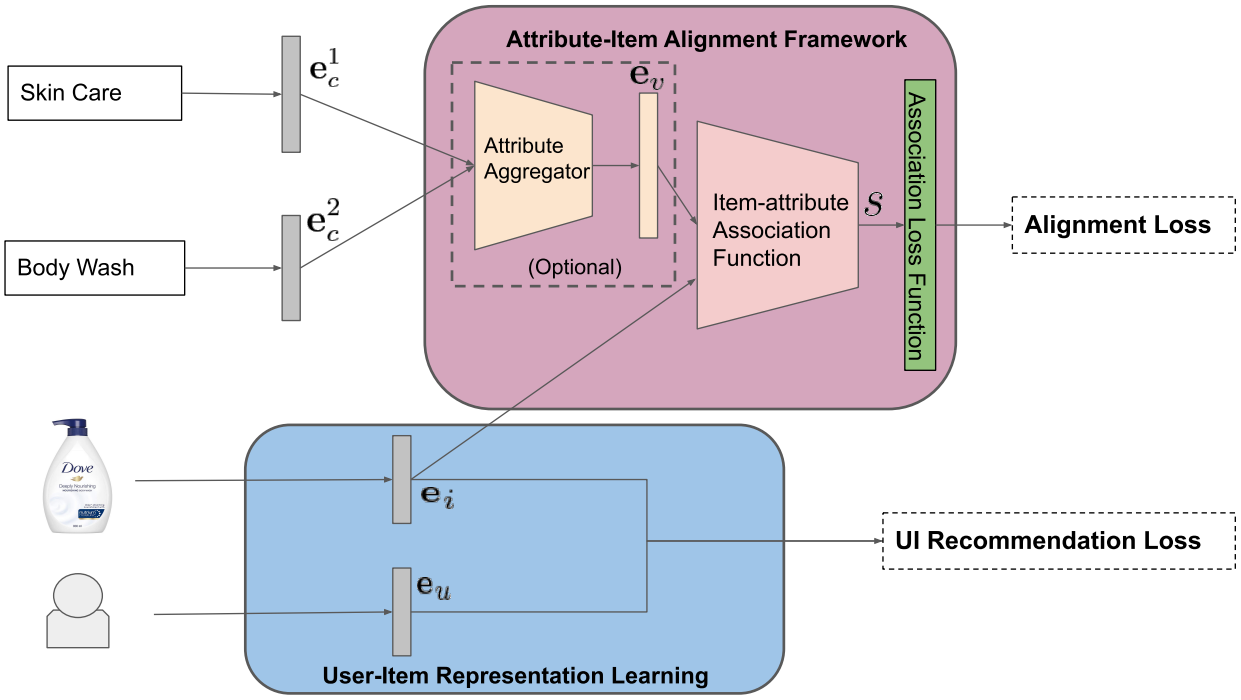


Figure 4.1: Overall structure of the AIA framework. The purple block denotes the AIA framework and the blue block denotes the regular UI recommendation block. Item embeddings are shared in both blocks. *First*, an attribute aggregator generates the aggregated attribute representation of an item, $\mathbf{e}_v$, by aggregating this item's attributes $\mathbf{e}_c^i$ if required. *Next*, the item-attribute association function is applied on the item representation $\mathbf{e}_i$ and either $\mathbf{e}_c^i$ or $\mathbf{e}_v$, depending on the system design, to measure the item-attribute association S. *Finally*, the association loss function is evaluated to compare S to the score of an irrelevant item-attribute pair $\bar{S}$ to train the model.

attributes $\mathcal{V} = \{c_1, \ldots, c_k\}$, and aggregate them into a single embedding vector, denoted as $\mathbf{e}_v$:

$$\mathbf{e}_v = \text{AGGREGATOR}(\{\mathbf{e}_c, c \in \mathcal{V}\}) \tag{4.2}$$

The association score is then computed between the item representation and the final attribute representation:

$$s = f_\theta(i, \mathcal{V})\,. \tag{4.3}$$

4.1.3 The Association Loss Function

The association loss function evaluates the accuracy of the pairwise association score. Instead of evaluating the association score computed for a single item-attribute pair (i, k), (i, k, l) will be considered as training data where k is the true relevant attribute associated with item i and l is an attribute that is assumed to be not relevant for i. The item l is sampled from the rest of the attributes excluding the true relevant attributes. Optimization would be performed based on the rank of these item-attribute pairs: (i, k) and (i, l). Specifically, item i is assumed to be more relevant to k over l. A contrastive loss is thus used in order to optimize the association measure function in an unsupervised manner [1, 2, 21]:

$$L = \sum_{(i \in \mathcal{I})} l(s, \bar{s}) \tag{4.4}$$

where L is the total association loss, and l is the loss function taking in the positive association score, and the negative association score $\bar{s}$.

4.2 Item side Information Algorithms in AIA Framework

In this section, we present three state-of-the-art side information enhanced recommendation models from distinct backgrounds and explain how they can be expressed in the AIA framework introduced in Section 4.1. Tables 4.1 and 4.2 provide a summary of the association measures and association loss functions for the analyzed algorithms.

For each model, we first review the component of the model where the attribute-item association is considered. Subsequently, we explain how the methodology can be reformulated in order to fit the association-based model in our proposed AIA framework. In this section, we continue to employ the notation defined in Table 3.1. Model-specific notation is introduced and explained only when it is necessary.

[1]The global bias w_0 and the weight of each entity $w_j, j \in \{u, i, C\}$ are omitted for clarity.
[2]The normalization factor τ is neglected.

Table 4.1: A summary of association measures employed in state-of-the-art baseline algorithms. $\mathbf{e}_i$ and $\mathbf{e}_c$ denote the latent representations for item i and attribute c, respectively. $\mathcal{V}_i = \{\mathbf{e}_{c_1}, \cdots, \mathbf{e}_{c_k}\}$ is the attribute set of item i. $\mathbf{e}_v^i = \sum_{\mathcal{V}_i}\{\mathbf{e}_c\}$ represents the aggregated attribute representation for item i. $\sigma(\cdot)$ is the sigmoid function. $\text{FFN}^{(i)}(\cdot)$ is used to denote a feed forward neural network with i layers where $\text{FFN}^{(i)}(\mathbf{X}) = \sigma(\mathbf{W_i}(\cdots\sigma(\mathbf{W_i}\mathbf{X}+\mathbf{b}_i)+\mathbf{b}_1)$.

Method	**Association Measure** $f(\cdot)$
KGAT	$\|\|\mathbf{W}_r\mathbf{e}_i + \mathbf{e}_r - \mathbf{W}_r\mathbf{e}_c\|\|_2^2$
NFM[1]	$\mathbf{h}^T \cdot \text{FFN}^l(\sum_{c\in C}\mathbf{e}_u \odot \mathbf{e}_c^c + \sum_{c\in C}\mathbf{e}_i \odot \mathbf{e}_c^c + \sum_{c\in C}\sum_{h\in C, h\neq c}\mathbf{e}_c^c \odot \mathbf{e}_c^h)$
CLIP[2]	$\mathbf{e}_i \cdot \mathbf{e}_v^{i\,T}$

Table 4.2: A summary of association loss functions for state-of-the-art baseline algorithms. $s_i = f(\mathbf{e}_i, \mathcal{V}_i)$ is the positive association score whereas $\bar{s_i} = f(\mathbf{e}_j, \mathcal{V}_i), \mathcal{V}_j \neq \mathcal{V}_i$ is the negative association score.

Method	**Loss function**
KGAT	$-\ln\left(\frac{1}{1+e^{s_i-\bar{s_i}}}\right)$
NFM	$-\ln\left(\frac{1}{1+e^{s_i-\bar{s_i}}}\right)$
CLIP	$(-s_i) + \log(\sum \mathbf{e}^{\bar{s_i}})$

4.2.1 KGAT [1]

In KGAT, the user interaction history and item knowledge are encoded in a unified knowledge graph, which is a directed graph with labelled edges. Specifically, each user, item, and attribute is modelled as an entity node. An edge connecting a user and an item captures the interaction between them, and an edge exists between an item and an attribute if there is a relation between them, such as "directed by", "genre", "acted by", etc.) The overall loss function of KGAT is composed of three objectives:

$$L = L_{\text{KG}} + L_{\text{CF}} + \lambda||\Theta||_2^2 \tag{4.5}$$

where λ is the regularization hyperparameter and Θ represents all trainable parameters within the model. L_{CF} is the collaborative filtering loss term that aims to train the user and item embeddings such that the user-item interactions can be reconstructed through an inner product of the two embeddings. $\lambda||\Theta||_2^2$ is the regularization term to prevent over-fitting.

L_{KG} is the objective function used to train knowledge graph node embeddings (user, item, and attribute nodes) such that neighbourhood relations in the local graph structure can be reconstructed. To be more specific, a relationship between entities is preserved by applying the widely used TransR [100] principle, where given embeddings of a head entity-relation-tail (h,r,t) entity triplet, it assumes the head and tail entities are close with each other in the specific relation space and are far away from those that do not hold the same relation. Mathematically, let $\mathcal{G}$ be the knowledge graph and (h, r, t) be an existing entity-relation-entity triplet in $\mathcal{G}$, each associated with an embedding, represented as $\mathbf{e}_h \in \mathbb{R}^d$, $\mathbf{e}_t \in \mathbb{R}^d$, $\mathbf{e}_r \in \mathbb{R}^k$, respectively. The TransR principle can then be expressed as: there exists a transformation matrix of relation r, $\mathbf{W}_r \in \mathbb{R}^{k \times d}$, such that the transformed head embedding $\mathbf{e}_h^r = \mathbf{W}_r \mathbf{e}_h$ and the transformed tail embedding $\mathbf{e}_t^r = \mathbf{W}_r \mathbf{e}_t$ satisfy the following relation:

$$\mathbf{e}_h^r + \mathbf{e}_r \approx \mathbf{e}_t^r \tag{4.6}$$

Based on this principle, we can define a *plausibility score* for each triplet:

$$g(h, r, t) = ||\mathbf{W}_r \mathbf{e}_h + \mathbf{e}_r - \mathbf{W}_r \mathbf{e}_t||_2^2 \tag{4.7}$$

To encourage discrimination between the existing triplets and the non-existing ones, a pairwise ranking loss is used:

$$L_{\text{KG}} = \sum_{(h,r,t,t') \in \mathcal{T}} - \ln \sigma(g(h, r, t') - g(h, r, t)) \tag{4.8}$$

where $\mathcal{T} = \{(h, r, t, t')|(h, r, t) \in \mathcal{G}, (h, r, t') \notin \mathcal{G}\}$. and (h, r, t') represents a broken triplet which does not exist in the knowledge graph (i.e., there is no edge between h and t'). Each broken triplet is constructed by replacing the tail entity in an existing triplet (h, r, t) with t' sampled randomly from the entity set.

The association measure function for KGAT: As introduced in Section 4.1, an association measure function takes in an attribute-item pair and outputs their association score. This is exactly what is defined in Equation (4.7), with head and tail entities limited to items and attributes. Therefore, viewed from the AIA framework, the association measure function of KGAT can be expressed as:

$$s(i, c) = ||\mathbf{W}_r\mathbf{e}_i + \mathbf{e}_r - \mathbf{W}_r\mathbf{e}_c||_2^2 \tag{4.9}$$

where $\mathbf{e}_r$ in our scenario represents the embedding of the relation *attributeOf*.

The association loss function for KGAT: The plausibility score $g(h, r, t)$ is equivalent to the association score in our framework. The association loss function can thus be expressed as:

$$l = -\ln \sigma(\bar{s} - s) = -\ln\left(\frac{1}{1 + \mathrm{e}^{s-\bar{s}}}\right) \tag{4.10}$$

where $\sigma(\cdot)$ is the sigmoid function.

4.2.2 Neural Factorization Machines (NFMs) [2]

As introduced in section 3.1, NFM encoded the given interaction history and item side information in a sparse feature matrix where each row is a binary vector representing an interaction record and the user, item, attribute(s) this record is related to. The attribute-item interactions are then explored through an element-wise product.

The association measure function of NFM: In our framework, given an interaction record of user u interacting with an item i with attributes $C = \{c_1, \cdots, c_k\}$, the feature embedding set can be expressed as: $\mathbf{V}_x = \{\mathbf{e}_u, \mathbf{e}_i, \mathbf{e}_c^1, \mathbf{e}_c^k\}$. Therefore, the association score can be considered as the prediction score defined in NFM, with the exception that this

association score not only measures association between attributes and items, but also association between users and attributes, and users and items:

$$s(u,i,C) = w_0 + \sum_{j\in\{u,i,C\}} w_j + \mathbf{h}^T \cdot \text{FFN}^l(\sum_{c\in C} \mathbf{e}_u \odot \mathbf{e}_c^c + \sum_{c\in C} \mathbf{e}_i \odot \mathbf{e}_c^c + \sum_{c\in C} \sum_{h\in C, h\neq c} \mathbf{e}_c^c \odot \mathbf{e}_c^h) \quad (4.11)$$

The association loss function of NFM: $\hat{y}_{\text{NFM}}(\mathbf{x})$ is equivalent to the association score in our framework. The association loss function can thus be expressed as:

$$l = -\ln \sigma(\bar{s} - s) = -\ln(\frac{1}{1+e^{s-\bar{s}}}) \quad (4.12)$$

where $\sigma(\cdot)$ is the sigmoid function.

4.2.3 CLIP [3]

Originally proposed as a solution to the image classification problem, the idea of CLIP is to enhance the image classification accuracy with the help of a text description of an image. Assuming we are given a collection of image examples $\{i^1, i^2, \cdots, i^N\}$, and their associated text descriptions $\{t^1, t^2, \cdots, t^N\}$, these two sources of information are first encoded into feature vectors through an image encoder and a text encoder, respectively, leading to a batch of (image, text) representation pairs $\{(\mathbf{i}^1, \mathbf{t}^1), \cdots, (\mathbf{i}^N, \mathbf{t}^N)\}$, where $\mathbf{i}^j \in \mathbb{R}^{d_i}$, $\mathbf{t}^j \in \mathbb{R}^{d_t}$. These pairs of representations are then transformed into a latent space:

$$\mathbf{i}_e^j = \mathbf{i}^j \cdot \mathbf{W}_i \quad (4.13)$$

$$\mathbf{t}_e^j = \mathbf{t}^j \cdot \mathbf{W}_t\,, \quad (4.14)$$

where $\mathbf{i}_e^j$, $\mathbf{t}_e^j$ are the projected image and text representations in the latent space, and $\mathbf{W}_i \in \mathbb{R}^{d_i, d_e}$, $\mathbf{W}_t \in \mathbb{R}^{d_t, d_e}$ are the learnable transformation matrices. A pairwise cosine similarity

score can then be calculated as follows:

$$s_j = \mathbf{i}_e^j \cdot \mathbf{t}_e^{j^T} \cdot e^t \,, \tag{4.15}$$

where t is a learnable temperature parameter that is used to control the scale of the similarity score. In order to train the image and the text encoder so that the cosine similarity of the matching (image, text) pair is larger than the similarity of the incorrect pairs in the multi-modal space, a cross-entropy loss is applied:

$$L = (-s_j) + \log(\sum_{l=1, l\neq j}^{N} e^{s_l}) \tag{4.16}$$

Inspired by how CLIP improves the image representation learning through learning from its text description, we[3] adapt this model into a recommendation scenario. To be more specific, we modify CLIP to address the top-K recommendation problem assuming that the user-item interaction history and the item category information are provided. The motivation behind this is that we can consider the (item, item attributes) relation to be equivalent to the (image, text description) relation in the original CLIP model. Therefore, a similar objective can be defined following the setup in CLIP to improve the item representation quality through learning from its corresponding attributes. The adapted model is depicted in Figure.4.2.

The architecture consists of two primary components. First, a user-item interaction bipartite graph $\mathcal{G}$ is constructed and the user-item correlation is explored through the state-of-the-art LightGCN [14] approach. Second, we extract the item node embedding from the bipartite graph. A category embedding matrix, $\mathbf{E}_c$, is also learned; this maps each item attribute to an embedding vector. Following the setup in CLIP, these two embeddings are jointly trained with the objective to correctly pair an attribute with an item.

[3]Machine learning researcher from Huawei, Yingxue Zhang, first suggested adapting CLIP to be used in the recommendation scenario. McGill University Ph.D. student, Haolun Wu, conducted the initial experiments to prove the effectiveness of this idea. I implemented the same model independently and tested its performance on another six different datasets.

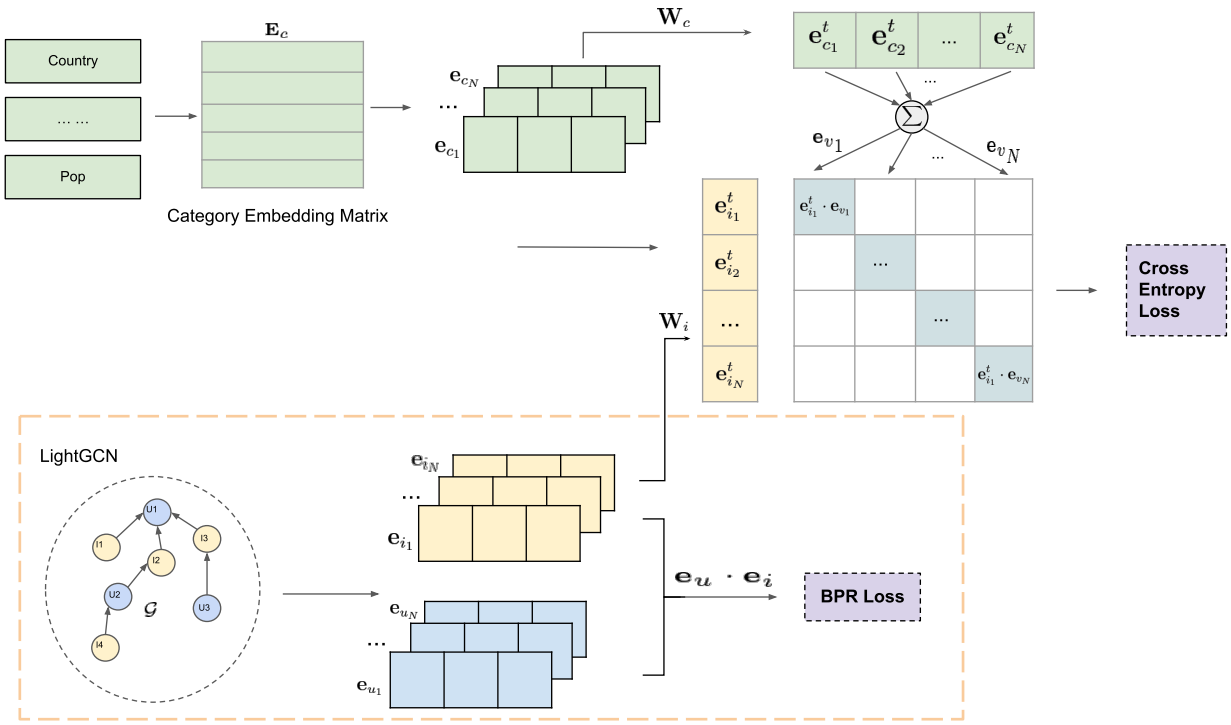


Figure 4.2: The overall architecture of CLIP adapted in a recommendation scenario. While a standard collaborative filtering model (LightGCN [14]) is applied on the UI-interaction bipartite graph to generate user and item embeddings, attribute embeddings $\mathbf{e}_{c_i}$ and item embeddings $\mathbf{e}_{i_j}$ are also jointly trained to predict the correct pairing of a batch of (item, attribute) training examples.

A detailed introduction of the LightGCN network is provided in Section 5.2.2. It generates user and item embeddings by processing the user-item bipartite graph. Specifically, a BPR [57] loss is used to train the LightGCN, which can be expressed as follows:

$$L_{LightGCN} = -\sum_{(u,i,j)\in\mathcal{O}} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda||\mathbf{E}||^2 \tag{4.17}$$

where $\hat{y}_{ui} = \mathbf{e}_u \cdot \mathbf{e}_i$ is the predicted preference score of a given (user, item) pair, $\mathcal{O} = \{(u,i,j)|(u,i) \in \mathcal{G}, (u,j) \notin \mathcal{G}\}$, $\mathbf{E}$ denotes the learnable parameters in the LightGCN network, and λ is a hyperparameter to control the regularization strength.

We first project the extracted item and attribute embedding $\mathbf{e}_i \in \mathbb{R}^{d_i}, \mathbf{e}_c \in \mathbb{R}^{d_c}$, to a multi-modal latent space:

$$\mathbf{e}_i^t = \mathbf{e}_i \cdot \mathbf{W}_i, \tag{4.18}$$

$$\mathbf{e}_c^t = \mathbf{e}_c \cdot \mathbf{W}_c \,. \tag{4.19}$$

Here $\mathbf{W}_i \in \mathbb{R}^{d_i,d}$ and $\mathbf{W}_c \in \mathbb{R}^{d_c,d}$ are the transformation matrices for the item embeddings and the attribute embeddings, respectively.

Next, we compute the aggregated attribute representation for each item through an average operation on all the attributes associated with the same item. Given an item i and the feature representation of its attribute set $\mathcal{C} = \{\mathbf{e_{c_1}^t}, \cdots, \mathbf{e_{c_n}^t}\}$, the aggregated attribute representation for item i is then computed as:

$$\mathbf{e}_{v_i} = \frac{1}{|\mathcal{C}|} \sum_{k \in \mathcal{C}} \mathbf{e_{c_k}} \,. \tag{4.20}$$

Following the set up in CLIP, we construct an item-attribute pairing objective to refine the item embedding with item attribute information. Specifically, given a batch of N (item, attribute) pairs, we aim to predict out of the $N \times N$ possible item-attribute pairs, which N pairs actually exist. Given the item representations in the transformed latent space, $\mathcal{I} = \{\mathbf{e_{i_1}^t}, \cdots, \mathbf{e_{i_N}^t}\}$, and the corresponding aggregated attribute representations, $\mathcal{C}_{\sqsubseteq} = \{\mathbf{e_{v_1}}, \cdots, \mathbf{e_{v_N}}\}$, a pairwise correlation score for each item-attribute pair can be computed by measuring their cosine similarity.

The association measure function: The adapted CLIP model fits straightforwardly into the proposed AIA framework. For each item k, there exists one real pair ($\mathbf{e_{i_k}^t}$, e_{v_k}), and $N-1$ incorrect pairs ($\mathbf{e_{i_k}^t}, \mathbf{e}_{v_j}$), ($j \in [1, N], j \neq k$). The correlation score of the real pair s_k can be seen as the association measure function in AIA framework, and is defined as:

$$s_k = \mathbf{e_{i_k}^t} \cdot \mathbf{e_{v_k}} \tag{4.21}$$

On the other hand, the correlation score of an incorrect item-attribute pair, $\bar{s}_k$, is computed as:

$$\bar{s}_k^j = \mathbf{e}_{\mathbf{i_k}}^{\mathbf{t}} \cdot \mathbf{e}_{\mathbf{v_j}} \tag{4.22}$$

where $(j \in [1, N], j \neq k)$.

The association loss function: The loss function is defined such that the correlation score for each real pair is maximized while the scores of the $N-1$ incorrect pairs are minimized. Specifically, a cross-entropy loss is applied over the correlation scores:

$$L = -(s_k) + \log \sum_{(j \in [1,N], j \neq k)} \bar{s}_k^j \tag{4.23}$$

4.3 Discussion

Although attribute representations can contain rich information about items, efficiently absorb them into item representations can be challenging. The three association measure functions summarized in Table.4.1 explore the interactions between items and attributes from different perspectives. Both KGAT's and CLIP's association measure functions have a clear physical meaning. They both capture the belief that item and attribute representations should be similar in terms of the directions of the two representation vectors. The key difference between KGAT and CLIP is whether this similarity measure is evaluated in the original space (CLIP) of the item and attribute representation vectors, or in a transformed space (KGAT). On the other hand, NFM explores the element-wise interactions between representations of items and attributes. This element-wise operation encourages each dimension in the item and attribute representations to describe a similar feature (*e.g.* , colour, brand, price, *etc.*).

For the loss functions summarized in Table 4.2, one key difference is whether the $\log$ is placed on the smoothed version of the association score difference (KGAT, NFM) or the negative scores only (CLIP). In the design of CLIP's loss function, the positive score s_i is directly used without smoothing (*e.g.* , through e^{s_i} or $\log(s_i)$). Therefore, we can think

of CLIP as putting more weight on positive scores while smoothing the impact from the negative scores.

4.4 Limitations of the AIA Framework

The introduced AIA framework is helpful in terms of exposing how hidden information in the item attributes is explored. However, the framework has limitations. It is only applicable to methods which: 1) explicitly learn feature representations for attributes; and 2) directly define the item and attributes feature interactions, as introduced in Section 4.2. In other words, methods that implicitly use the attribute information cannot fit into our proposed framework.

For example, in one of our baseline models, MGCCF [15], the attribute information is only used in the item-item graph construction (the detailed description of the item-item graph construction method is provided in Section 5.3.3). The model does not learn specific attribute embeddings. There is no direct interaction in the model between the features of the items and those of the attributes. Therefore, we cannot express MGCCF using the AIA framework.

Another example is cVAE [94], which is also a baseline in our experiments. Given a user-item interaction history and item-attribute information, a side information matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ and an interaction matrix $\mathbf{Y} \in \mathbb{R}^{m \times n}$ are constructed, where m denotes the number of users, n denotes the number of items, and d denotes the number of attributes. As is the case for a standard variational auto-encoder, cVAE is composed of two components: 1) an inference network with parameters ϕ, f_ϕ, which encodes the observed data as a distribution over a latent space; and 2) a generation network with parameters θ, f_θ, which takes sampled points from the latent space and recovers the observed data. The basic idea of cVAE is to first train a variational auto-encoder such that the side information matrix $\mathbf{X}$ is recovered:

$$\mathbf{Z} \sim f_\phi(\mathbf{X}), \mathbf{X} \sim f_\theta(\mathbf{Z})\,. \tag{4.24}$$

The same network is then refined to recover the interaction matrix $\mathbf{Y}$:

$$\mathbf{U} \sim f_\phi(\mathbf{Y}), \mathbf{Y} \sim f_\theta(\mathbf{U})\,. \tag{4.25}$$

Here $\mathbf{Z}$ is the latent feature representation matrix and $\mathbf{U}$ is the latent user representation matrix. The vectors in $\mathbf{Z}$, $\mathbf{U}$ are assumed to follow Gaussian distributions.

Although an attribute embedding matrix is defined in cVAE (*i.e.* $\mathbf{Z}$), there is no direct measurement of the correlation between item embeddings and the attribute embeddings. Therefore, neither the association measure function nor the association loss function can be specified. As a result, cVAE cannot be expressed in our proposed framework.

In summary, in order to fit a method into the AIA framework, the method has to maintain a trainable attribute representation matrix **and** there must be functions defined to measure the correlation between an item representation and the representations of its associated attributes. This is the major limitation of our proposed AIA framework.

4.5 Summary

In this chapter, we present a novel framework named AIA to encompass recommender systems that incorporate item side information. The framework consists of two components: the association measure function and the association loss function. We demonstrate how different side information enhanced recommendation algorithms can be expressed in the proposed framework and discuss their major differences by comparing their association measures and loss functions, viewed from the perspective of the AIA framework. In the next chapter, we describe our proposed item side information enhanced recommendation system in detail and show how the different designs of association measure functions we identified in this chapter have inspired us in the design of our proposed method.

Chapter 5

Mutual Information Alignment (MIA)

In this chapter, the problem we are aiming to solve is first defined, followed by the introduction of the overall structure of our proposed method, Mutual Information Alignment (MIA) for recommender systems. Subsequently, the key components of MIA in details are explained. We then describe experiments set up and compare the performance of MIA with the state-of-the-art baselines. Experimental results, together with a discussion of our findings are then presented.

5.1 Problem Definition

Our model is designed to learn high-quality representations of items by incorporating item side information. These representations can then be used in the top-N recommendation task. Specifically, following the notation defined in Table 3.1, given a user-item interaction history, $\mathbf{R} \in \mathcal{R}^{n \times m}$, and the item categorical indicator matrix, the model predicts a preference score for each user-item pair. Items with the top N scores are then recommended to the users.

5.2 Overall Structure

The overall architecture of MIA is presented in Figure 5.1. MIA adopts a multi-view alignment schema with three key components: (i) a user-item view that aims to generate embeddings for users and items by processing the information in the UI interaction matrix; (ii) an attribute view that acts as an encoder for item attribute features; and (iii) an item-attribute alignment component with a mutual information estimator which strives to maximize the mutual information between cross-view item-attribute pairs.

5.2.1 MIA in the AIA Framework

MIA is designed closely follows the AIA framework. In particular, the item-attribute association function is equivalent to the item-attribute association measure function and the alignment loss is equivalent to the association loss function in the AIA framework. Details about these two functions are provided in Section 5.2.4.

5.2.2 User-item View

User-item (UI) relations are modeled in the UI view using a bipartite graph and explored by LightGCN [14], a state-of-the-art collaborative filtering method. This component of the recommender system is not the focus of our design, so in theory any approach could be used (e.g. matrix factorization-based approaches [57,60], auto-encoder-based approaches [66,101], and other graph-based approaches [9,102])

Construction of User-Item View: The user-item (UI) view mainly contains a UI interaction graph. Given the UI interaction history, interactions can be represented in a bipartite graph $\mathcal{G}_i = \{(u, y_{ui}, i)|u \in U, i \in I\}$, where U and I represent the user and item sets, respectively. An edge e_{ui} is considered a positive feedback of item i made by user u.

Learning in the UI View: Learning in the UI view is equivalent to learning node embeddings for the UI bipartite graph. Here, we choose to use a state-of-the-art col-

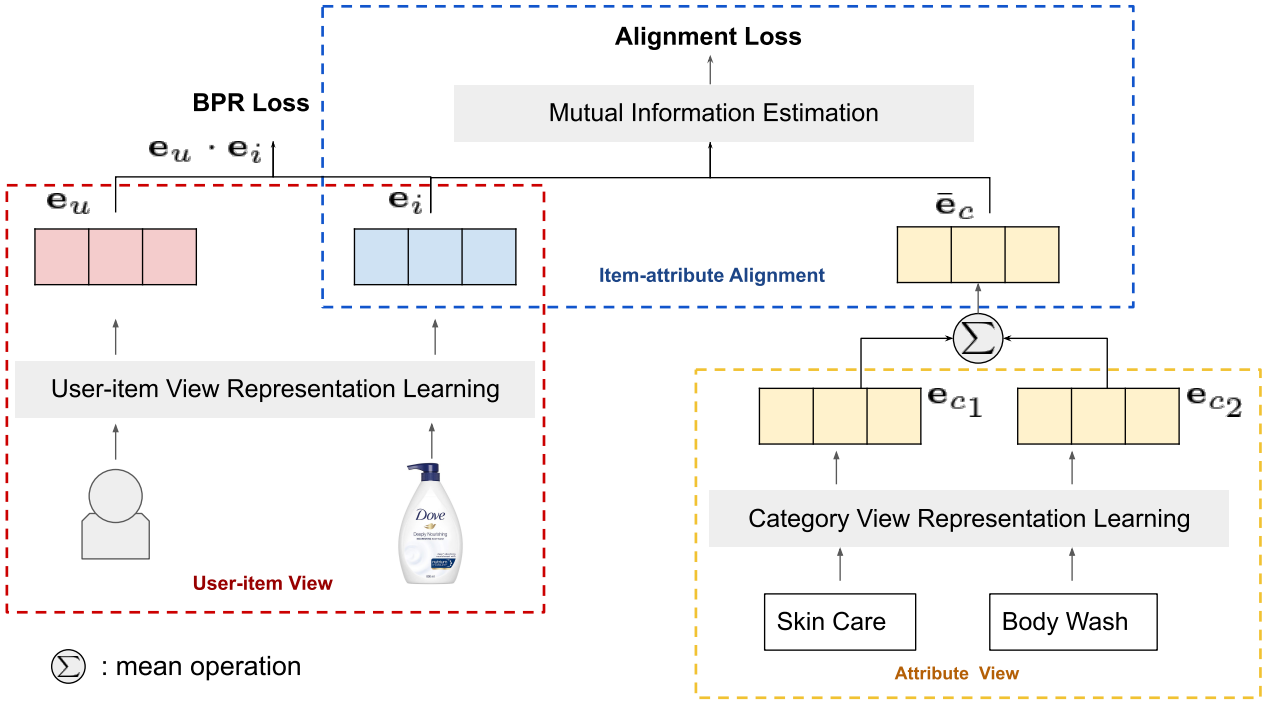


Figure 5.1: The overall architecture of MIA. Arrowed lines denote the flow of information. Every user, item and attribute is embedded into a d-dimensional space and $\mathbf{e}_u \in \mathcal{R}^d$, $\mathbf{e}_i \in \mathcal{R}^d$ indicate the latent representations of user u, item i learned from the user-item view and $\mathbf{e}_c \in \mathcal{R}^d$ and $\bar{\mathbf{e}}_c$ represent the latent representation of attribute c learned from the attribute view and the aggregated attribute representation for an item, respectively.

laborative filtering method named *LightGCN*. The training objective of LightGCN [14] is to learn item and user representations in order to achieve the best predictions of the user preference. More formally, given the binary user-item interaction matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$, the adjacency matrix of the bipartite graph can then be defined as a binary matrix $\mathbf{A} \in \mathbb{R}^{(n+m)\times(n+m)}$ where:

$$\mathbf{A} = \begin{pmatrix} 0 & \mathbf{R} \\ \mathbf{R}^T & 0 \end{pmatrix} \tag{5.1}$$

The graph convolution in LightGCN is defined as:

$$\mathbf{E}^{(k+1)} = \tilde{\mathbf{A}}\mathbf{E}^{(k)} = (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{E}^{(k)} \tag{5.2}$$

where k is the number of layers and $\mathbf{D} \in \mathbb{R}^{(n+m)\times(n+m)}$ is a diagonal matrix with $\mathbf{D}_{ii}$ being equal to the number of nonzero entries in the i^{th} row of $\mathbf{A}$. $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrical normalized Laplacian matrix. $\mathbf{E}^{(k)}$ denotes the embeddings from the k^{th} layer. The embeddings of the 0^{th} layer, $\mathbf{E}^{(0)} \in \mathbb{R}^{(n+m)\times d}$, are the only trainable parameters. The final embeddings used for model prediction are obtained by summing the embeddings from each layer to form a single representation:

$$\begin{aligned}\mathbf{E} &= \mathbf{E}^{(0)} + \mathbf{E}^{(1)} + \mathbf{E}^{(2)} + \cdots + \mathbf{E}^{(K)} \\ &= \mathbf{E}^{(0)} + \tilde{\mathbf{A}}\mathbf{E}^{(0)} + \tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \cdots + \tilde{\mathbf{A}}^K\mathbf{E}^{(0)}\end{aligned} \tag{5.3}$$

To train the LightGCN model, a Bayesian Personalized Ranking (BPR) loss [57] is employed. This loss encourages the assignment of higher prediction scores to observed interactions. The BPR loss is:

$$L_{item\_view} = -\sum_{u=1}^{n}\sum_{i\in N_u}\sum_{j\notin N_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) \,, \tag{5.4}$$

where N_u is the one-hop neighborhood of u and $\sigma(\cdot)$ is the sigmoid function.

5.2.3 Attribute View

An attribute view is used to record the item attribute information. In MIA, we propose to learn a representation for each attribute through a simple encoder approach. In this approach, each attribute is mapped to a vector embedding. Given a set of attributes $\mathcal{Z}$, the encoder function is simply an "embedding lookup":

$$\text{ENC}(z_i) = \mathbf{z}_i\mathbf{E}_c \,, \tag{5.5}$$

where $\mathbf{E}_c \in \mathcal{R}^{|C|\times d}$ is a matrix containing embedding vectors for all the attributes and $\mathbf{z}_i \in \mathcal{I}_{\mathcal{C}}$ is an one-hot indicator vector indicating the row of $\mathbf{E}_c$ corresponding to attribute $\mathbf{z}_i$. In this approach, the set of trainable parameters is simply $\Theta_{\text{ENC}} = \{\mathbf{E}_c\}$, *i.e.* , the

embedding matrix $\mathbf{E}_c$ is optimized directly and there is no specific attribute-view loss function for this approach.

5.2.4 Mutual Information-Based Multi-view Alignment

A mutual information estimator is added to act as a multi-view alignment operator. The intuition behind this component is that the attribute embeddings generated from the attribute view should provide complementary information about the items, which is not captured in embeddings generated from the user-item view. Therefore, to incorporate the additional information into the item representations, we introduce a separate loss function that strives to align each item embedding with its associated attribute embeddings. In particular, let $\mathbf{e}_i$ denote the i-th item embedding generated from the user-item view, and let $\mathcal{Z}_i = \{\mathbf{e}_{c_1}, \cdots, \mathbf{e}_{c_k}\}$ be the set of attribute representations of item i. We assume that the attribute information can be aggregated through an element-wise sum operation. The intuition behind this is as follows. Consider a scenario in which movie recommendations are made for different users and the movie "Harry Potter" is associated with three attributes: drama, adventure and fantasy. Intuitively, users who have interacted with movies associated with all of the above attributes are more likely to be interested in "Harry Potter" than users who have interacted with movies that only match one or two of them. Therefore, we argue that all the attribute information should contribute to the aggregated attribute representation in order for the system to better distinguish between different movies. In other words, we want to aggregate the attribute information through an "and" operation without losing information contained in any one of the attributes. Therefore, the element-wise sum operation is chosen. The aggregated attribute representation of an item i is defined as:

$$\bar{\mathbf{e}}_c^i = \frac{1}{|\mathcal{Z}_i|} \sum_{l \in \mathcal{Z}_i} \{\mathbf{e}_{c_l}\} \tag{5.6}$$

Our goal is to maximize the mutual information between the item representation and the aggregated attribute representation:

$$\operatorname{argmax} \hat{I}(\mathbf{e_i}; \bar{\mathbf{e}}_c^i)\,, \tag{5.7}$$

where $\hat{I}(\cdot)$ is the mutual information estimator.

Mutual Information Estimator To estimate the mutual information between a pair of inputs, we use the Deep InfoMax (DIM) model proposed in [22]. This estimates the mutual information by training a classifier to distinguish between samples from different attributes. In particular, we use the formulation where the lower bound on the MI is based on the Jenson-Shannon divergence (JSD) [22]. The objective of the alignment component can then be defined as:

$$\begin{aligned} L_{alignment} &= -\operatorname*{argmax}_{\omega} \hat{I}^{\text{JSD}}(\mathbf{e}_i; \bar{\mathbf{e}}_c^i)\,, \\ &:= \mathbb{E}[sp(T_\omega(\bar{\mathbf{e}}_i; \mathbf{e}_c^i))] - \mathbb{E}[sp(T_\omega(\mathbf{e}_i; \mathbf{e}_c^i))]\,, \end{aligned} \tag{5.8}$$

where $\bar{\mathbf{e}}_i$ is the negative item sampled from items with a completely different set of attributes from item i, $sp(z) = \log(1 + \mathbf{E}^z)$ represents the softplus function, and $T_\omega : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$ is an item-attribute association function which measures the association score between an item and its corresponding attribute(s) with parameters ω.

Item-Attribute Association Measure Function: We investigate three different methods to compute the association score: a bi-linear network, a fully connected feed-forward network (FFN), and a bi-interaction network proposed in [1]. The definitions are described in Table 5.1.

[1] $\mathbf{W} \in \mathcal{R}^{d \times d}$ is the trainable weight matrix.

[2] '||' denotes the concatenation operation. $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ are the trainable parameters in the fully forward network.

[3] $\mathbf{W}_1, \mathbf{W}_2 \in \mathcal{R}^{d \times d}$ are the trainable weight matrices.

Table 5.1: Comparison of different association measure functions, T. $\sigma(\cdot)$ denotes the sigmoid function.

Method	Formula
$T_{\text{Bi-Linear}}$[1]	$\sigma(\mathbf{e}_i^T \cdot \mathbf{W} \cdot \bar{\mathbf{e}}_\mathbf{c}^\mathbf{i})$
T_{FFN} [2]	$\mathbf{W}_3(\sigma(\mathbf{W}_2 \cdot \sigma(\mathbf{W}_1(\mathbf{e}_i \| \bar{\mathbf{e}}_\mathbf{c}^\mathbf{i}) + \mathbf{b}_1)) + \mathbf{b}_2) + \mathbf{b}_3$
$T_{\text{Bi-Interaction}}$[3]	$\sigma(\mathbf{W}_1(\mathbf{e}_i + \bar{\mathbf{e}}_\mathbf{c}^\mathbf{i})) + \sigma(\mathbf{W}_2(\mathbf{e}_i \odot \bar{\mathbf{e}}_\mathbf{c}^\mathbf{i}))$

5.2.5 Model Prediction

The preference score of each user-item pair is generated by calculating the inner product of the final item and user embeddings extracted from the user-item view:

$$y_{ui} = \mathbf{e}_u^T \mathbf{e}_i \,, \tag{5.9}$$

where $\mathbf{e}_u$ denotes the representation of user u and $\mathbf{e}_i$ denotes item i's representation.

5.2.6 Model Training

To optimize MIA, the objectives of each individual task are combined linearly to form the final loss function:

$$L_{total} = \alpha L_{item\_view} + L_{alignment} + \lambda ||\Theta||^2 \,. \tag{5.10}$$

Here α is the weight controlling how much each objective contributes to the final loss, Θ is the model parameter set and λ denotes the L_2 regularization hyperparameter. Note that the trainable parameters in the model are: the embedding matrix of the 0^{th} layer, $\mathbf{E}^{(0)}$, in the user-item view, the attribute embedding matrix, $\mathbf{E}_c$, in the attribute view and potentially the weight matrix in the multi-view alignment component.

5.3 Experimental Settings

In this section, we describe how data are prepared and how experiments are conducted to evaluate our proposed framework on the *Top-K* recommendation task. To evaluate the effectiveness of the proposed method, we conduct experiments on six real-world datasets. We first conduct comprehensive experiments on different variants of MIA, to study how different item-attribute association functions can impact the final performance. Next, we compare the best MIA variant with state-of-the-art recommendation approaches with and without incorporating side information. Overall, the experiments are designed to answer the following research questions:

- **RQ1**: Among the three item-attribute association functions, which one can best capture interactions between items and features?
- **RQ2**: How does MIA perform compared to state-of-the-art item side information-enhanced collaborative filtering methods?

In what follows, we first present the experimental settings and then address the research questions.

5.3.1 Dataset

In this work, experiments are conducted on six benchmark datasets: *Amazon-CDs*, *Amazon-Movies*, *Amazon-Books*, *Amazon-Sports*, *Yelp* and *LastFM*. All datasets are publicly accessible with various domains, sizes and sparsity:

- **Amazon Reviews**[4]: Amazon Reviews is a widely used dataset for recommendation task [1, 15] which contains customer feedback and product metadata for various collections of products on the Amazon e-commerce platform. Specifically, we treat the explicit user ratings as implicit feedback by considering ratings higher than or equal to four as positive feedback. Any other ratings are treated as negatives (as

[4]http://deepyeti.ucsd.edu/jianmo/amazon/index.html

are any items that are not rated). Category information is used as the item side information. A product can belong to multiple categories.

- **Yelp** [5]: This dataset contains user feedback and details about local businesses such as restaurants and bars. As it is very large, we choose to use the transaction records dated from January 1st, 2018 to December 31st, 2019. Again, we adopt a similar approach, defining positive feedback from a user if the rating is higher than or equal to four. Business categories are used as the only source of item side information.

- **LastFM**[6]: This is a music artist recommendation dataset that contains user listening behaviours and user tags for artists. We adopted the same version used in [1, 21] where a subset of the dataset is taken to include records with timestamps dated from Jan 2015 to June 2015. Each artist is viewed as an item, and the tags assigned by users are viewed as item attributes.

For all the datasets, we first remove items with no associated item side information. We then apply a 10-core setting to filter out users and items with less than 10 interactions. For each dataset, 70% of the interaction history of each user is used as the training set, and 10% of interactions are treated as a validation set to tune hyperparameters. The remaining 20% of the interactions are used as the test set. The statistics of the processed data are summarized in Table 5.2.

Table 5.2: Statistics of evaluation datasets.

Dataset	# Userss	# Items	# Interactions	# Categories	Density (%)
Amazon-CDs	35,423	13,306	713,693	280	0.151
Amazon-Movies	11,840	10,801	275,939	282	0.216
Amazon-Books	16,457	13,343	697,971	292	0.318
Amazon-Sports	9,371	5569	139,779	928	0.268
Yelp	18,280	13,334	779,968	766	0.319
LastFM	1090	3646	52551	388	1.322

[5]https://www.yelp.com/dataset/documentation/main
[6]https://grouplens.org/datasets/hetrec-2011/

5.3.2 Evaluation Protocols

The Setting of "K": In the literature of recommendation systems and the problem setting in our thesis, we are interested in the system accuracy of the top-K recommended items. Therefore, instead of computing evaluation metrics over all the items, we compute them over the first K items. Thus, in the metrics introduced below, *Recall@K* and *Ndcg@K*, K is a definable integer selected to match the top-K recommendation objective. Throughout the experiments, $K = 20$ is used, which is a common choice in the recommender system literature [1, 15, 52, 94].

Recall@K indicates the percentage of relevant items found in the top-K recommendation list. In particular, for a user-item pair, an item is considered *relevant* if the user is truly interested in it (had a positive user rating). Mathematically, it is defined as:

$$Recall@K = \frac{\text{\#. of relevant items in the top-K recommendation list}}{\text{\#. of total relevant items}} \tag{5.11}$$

For example, given a user with 10 relevant items in the test set, if we find that Recall@$20 = 40\%$ for this user in our top-20 recommendation system, we know that 40% of the total number of the relevant items, *i.e.* 4 out of 10 relevant items, appear in the top-20 results. Recall@K of a system is computed by averaging Recall@K over all its users.

Ndcg@K (Normalized Discounted Cumulative Gain), on the other hand, takes into consideration the positions of the relevant items in the length-K recommendation list:

$$Ndcg@K = \frac{DCG@K}{iDCG@K} \tag{5.12}$$

where

$$DCG@K = \sum_{i}^{K} \frac{2^{pre_i} - 1}{\log_2(i+1)} \tag{5.13}$$

$$iDCG@K = \sum_{i}^{K} \frac{1}{\log_2(i+1)} \tag{5.14}$$

pre_i is the preference of the item at index i. Since we are considering binary preferences, $pre_i = 1$ if the item at the i-th position belongs to the ground truth, and 0 otherwise. In this way, an item at rank 1 would receive a relative weight of 1, whereas an item at rank 20 would receive a weight of $1/\log_2(21)$.

For all experiments, we compare our model and baselines in terms of *Recall@20* and *Ndcg@20*.

5.3.3 Baseline Algorithms

To investigate the effectiveness of the proposed method, we compare the performance of MIA with the following baselines. Note that I implemented all the baselines along with the proposed method MIA in PyTorch.

Baseline 1: A state-of-the-art collaborative filtering (CF) method that does not incorporate side information. Specifically, we choose the LightGCN [14] algorithm, which is also employed in the user-item view component in our proposed method MIA. Comparison with this baseline is mainly to assess the impact of the extra components (attribute view and item-attribute alignment).

LightGCN [14]: This is the state-of-the-art graph-based CF method that explicitly integrates a bipartite graph structure into the embedding learning process to model the high-order connectivity in the user-item graph. The popular feature transformation component and nonlinear activation layers are eliminated because these are observed to have a detrimental effect on the performance in the recommendation scenario.

Baseline 2-6: Five state-of-the-art side information enhanced methods are studied and implemented as baselines. In particular, we compare our proposed method MIA with a matrix factorization method (NFM [2]), an auto-encoder-based method (cVAE [94]), graph neural network-based methods (KGAT [1], MGCCF(Adapted) [15]) and an adapted tag-based recommendation method (CLIP(Adapted) [3]).

NFM [2]: The state-of-the-art factorization-based recommender system that generalizes the factorization machine (FM) by incorporating neural networks. In our experiment, one hidden layer is used, as recommended in [9].

cVAE [94]: An auto-encoder based recommender system which jointly recovers user ratings and side information through a variational auto-encoder. Following the settings in the original paper, we first pre-train the cVAE by feeding it side information only and then refine the model by feeding it user ratings.

KGAT [1]: A graph-based approach that aims to model high-order end-to-end relations to provide a better recommendation. The categories are represented as additional entity nodes in a collaborative knowledge graph, and an edge is added between an item and a category if the item belongs to the category.

MGCCF [15]: Designed to capture the intrinsic differences between item-item and user-item relationships, MGCCF provides a framework to combine relational embeddings learned from separate graphs.

CLIP (Adapted) [3]: CLIP was originally designed for the image classification task. The model trains the image label encoder and the image encoder simultaneously by predicting the correct label for an image. Inspired by CLIP, we modified the model such that the image-label paring task is replaced by the item-attribute prediction task. Details about the adapted CLIP are provided in Section 4.2.3.

5.3.4 Parameter Settings

All models are optimized using the Adam optimizer. The embedding size is fixed to 50 and the number of negative samples is set to 10. For all models, the *loguniform* sampling method of the Ray.tune [7] package is applied to tune the learning rate in $\{1e^{-4}, 1e^{-3}, 1e^{-2}\}$, and the L_2 regularization coefficient λ in $\{1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}\}$. In most cases, the best learning rate magnitude is $1e^{-3}$. The best regularization coefficient is $1e^{-3}$ or $1e^{-4}$.

For models with a LightGCN [14] component (LightGCN, MIA), we follow the best setting as reported in [14] and set the number of layers to 3 and dropout probability to 0.5.

[7] https://docs.ray.io/en/master/tune/index.html

For MIA specifically, we search for the best objective weight α in $\{1e^{-4}, 1e^{-3}, \cdots, 1e^{4}\}$. In most cases, the best tuned weight is of the order of magnitude as $1e^{-3}$.

Early-stopping is also implemented, i.e., training is stopped if *Recall@20* on the validation set does not increase for 50 successive epochs. The maximum number of training epochs is 1500.

5.4 Results

5.4.1 Impact of Item-Attribute Association Functions (RQ1)

To study the effect of the three item-attribute association functions defined in Section 5.2.4, we compare these choices in terms of their impact on the model performance. Table 5.3 reports the average performance result over five trials with random weight initialization.

Table 5.3: Comparison of different item-attribute association methods on six datasets. Method with the best performance is highlighted with underlined text.

		MIA (Bi-Linear)	MIA (FFN)	MIA (Bi-Interaction)
Amazon-Movies	Recall@20	0.0962±0.0015	0.0886±0.0016	<u>0.1055±0.0011</u>
	Ndcg@20	0.0496±0.0007	0.0412±0.0008	<u>0.0539±0.0005</u>
Amazon-CDs	Recall@20	0.1156±0.0008	0.1112±0.0008	<u>0.1245±0.0007</u>
	Ndcg@20	0.0591±0.0004	0.0557±0.0005	<u>0.0655±0.0004</u>
Amazon-Books	Recall@20	0.0815±0.0009	0.0799±0.0009	<u>0.0831±0.0008</u>
	Ndcg@20	0.0428±0.0005	0.0404±0.0006	<u>0.0442±0.0005</u>
Amazon-Sports	Recall@20	0.0776±0.0014	0.0763±0.0014	<u>0.0812±0.0013</u>
	Ndcg@20	0.0415±0.0008	0.0374±0.0008	<u>0.0433±0.0007</u>
Yelp	Recall@20	0.0613±0.0005	0.0593±0.0004	<u>0.0660±0.0003</u>
	Ndcg@20	0.0502±0.0004	0.0480±0.0005	<u>0.0544±0.0003</u>
LastFM	Recall@20	0.0914±0.0021	0.0899±0.0021	<u>0.1015±0.0020</u>
	Ndcg@20	0.0464±0.0011	0.0455±0.0011	<u>0.0503±0.0011</u>

From the results demonstrated, we make the following observations:

- The *bi-Interaction* association measure function outperforms both *Bi-Linear* and *FFN* functions for all datasets.

- *Bi-Linear* and *Bi-Interaction* are substantially superior to *FFN* across all datasets. This highlights the importance of feature interactions between the item and the attribute embeddings.

- Comparing the results of *Bi-Linear* function and *Bi-Interaction* function, although both functions take into account the pairwise feature interactions, the *Bi-Interaction* association function consistently outperforms the *Bi-Linear* association function. One possible reason is that the element-wise operation (summation, multiplication) used in *Bi-Interaction* association function is able to aggregate more information in the item and attribute embeddings than the vector projection operation used in the *Bi-Linear* function.

Overall, these observations are in line with our expectation that feature interaction is important in assessing the association between items and attributes. Moreover, the results indicate the effectiveness of the element-wise operation in the case of item-attribute interaction.

5.4.2 Comparison with Baselines (RQ2)

We compare the performance of the six baseline methods with the best variant of our proposed method, MIA (Bi-Interaction). Results are presented in Table 5.4. For each model, the average performance (with standard deviation) over 10 trials with random weight initialization is reported.

Significance test - Wilcoxon signed rank test:

To determine whether the observed performance difference between our proposed method and the best baseline method is significant across trials, we applied the Wilcoxon signed-rank test. We use a $*$ in Table 5.4 to denote a statistically significant difference.

Since our sample size is relatively small ($n = 10$), we compare the obtained test statistic, T, directly to the critical value of the Wilcoxon signed-rank test [103]. Specifically, given a sample size $n = 10$, the obtained performance difference is statistically significant if $T \leq 10$ at the 5% significance level.

Table 5.4: Overall performance comparison w.r.t **Recall**@20 and **Ndcg**@20. The best and the second-best model are denoted in underlined and bold fonts respectively. "$*$" indicates a statistically significant difference between MIA and the best baseline method on the Wilcoxon signed-rank test. $\%$*Improv.* denotes the percentage of improvement of MIA compared to the best baseline method.

	Amazon-Movies	Amazon-CDs	Amazon-Books	Amazon-Sports	Yelp	LastFM
LightGCN	0.0809±0.0008	0.0797±0.0003	0.0685±0.0006	0.0580±0.0012	0.0550±0.0002	0.0759±0.0015
MGCCF	0.0908±0.0010	0.0999±0.0005	0.0730±0.0007	0.0766±0.0014	0.0609±0.0003	0.0869±0.0018
cVAE	0.0690±0.0012	0.0712±0.0005	0.0555±0.0007	0.0512±0.0009	0.0514±0.0004	0.0709±0.0017
NFM	0.0886±0.0013	0.0912±0.0005	0.0721±0.0009	0.0648±0.0011	0.0600±0.0004	0.0862±0.0021
KGAT	**0.1036±0.0011**	**0.1213±0.0006**	0.0794±0.0007	**0.0793±0.0013**	0.0679±0.0002	**0.0972±0.0019**
CLIP	0.0932±0.0013	0.1171±0.0007	**0.0804±0.0010**	0.0769±0.0012	0.0603±0.0003	0.0905±0.0019
MIA (Bi-Interaction)	0.1055±0.0011*	0.1245±0.0007*	0.0831±0.0008*	0.0812±0.0013*	**0.0660±0.0003**	0.1015±0.0020*
%Improv.	1.83%	2.64%	3.36%	2.40%	-2.80%	4.42%

(a) Recall@20

	Amazon-Movies	Amazon-CDs	Amazon-Books	Amazon-Sports	Yelp	LastFM
LightGCN	0.0386±0.0004	0.0417±0.0002	0.0401±0.0004	0.0305±0.0006	0.0401±0.0003	0.0333±0.0008
MGCCF	0.0412±0.0004	0.0566±0.0003	0.0409±0.0004	0.0374±0.0007	0.0480±0.0003	0.0408±0.0009
cVAE	0.0290±0.0003	0.0303±0.0002	0.0337±0.0004	0.0299±0.0006	0.0385±0.0002	0.0287±0.0008
NFM	0.0399±0.0006	0.0527±0.0004	0.0406±0.0005	0.0337±0.0006	0.0482±0.0003	0.0402±0.0011
KGAT	**0.0521±0.0005**	**0.0641±0.0004**	**0.0423±0.0005**	**0.0428±0.0006**	0.0549±0.0003	**0.0489±0.0011**
CLIP	0.0448±0.0004	0.0613±0.00004	0.0410±0.0004	0.0396±0.0007	0.0493±0.0003	0.0457±0.0010
MIA (Bi-Interaction)	0.0539±0.0005*	0.0655±0.0004*	0.0442±0.0005*	0.0433±0.0007	**0.0544±0.0003**	0.0503±0.0011*
%Improv.	3.45%	2.18%	4.49%	1.17%	-0.91%	2.86%

(b) Ndcg@20

Based on the results, we make the following observations:

- MIA outperforms all the baselines in five out of six datasets, except for Yelp, where MIA is outperformed by KGAT. In all cases, the performance difference between MIA and KGAT is less than 5% on both metrics. Therefore, we conclude that MIA and KGAT achieve comparable accuracy, with MIA achieving a small, but statistically significant, improvement. Further analysis is conducted below to compare these two methods.

- Comparing MIA with LightGCN, we can see that MIA outperforms LightGCN in all cases, which demonstrates the effectiveness of the additional components in MIA (attribute view and item-attribute alignment) that are specifically designed to incorporate item side information.

- Among all the side information-enhanced methods, cVAE is the only algorithm whose performance is worse than LightGCN, a method that does not incorporate side information, across all six datasets. This shows that cVAE cannot make effective use of the item knowledge. Incorporating item side information does not automatically guarantee a better performance; it must be integrated in a meaningful way.

- Based on the study of the AIA framework, we saw that KGAT and NFM share the same association loss function. But KGAT substantially outperforms NFM in all cases. One reason for this is that KGAT has an extra objective which explores the knowledge graph, as introduced in Section 4.2.1. Another possible reason is that in NFM, pairwise interaction is not only conducted between user-item and item-attribute pairs, but also among the user-attribute and attribute-attribute pairs. It is difficult to see why these interactions provide meaningful information; their inclusion may in fact harm the generalization power of the model.

5.4.3 Further Analysis of KGAT VS. MIA

From Table 5.4, we find that although MIA can outperform KGAT in some of the datasets, the percentage improvement is relatively small (less than 5%). Therefore, we conduct further analysis to compare these two methods. In particular, we compare KGAT and MIA (Bi-Interaction) variants from two perspectives: computation complexity and convergence speed.

Computation Complexity Analysis

We first analyze the computation complexity of KGAT and MIA.

Let $|E_{uia}|$ be the number of edges in the user-item-attribute knowledge graph employed in KGAT, and let $|E_{ia}|$ be the number of edges in the item-attribute graph, and let $|E_{ui}|$ be the number of edges in the user-item graph. In most cases, we expect that the graphs are sparse, so that $|E_{ui}|$ is $O(|U| + |I|)$.

As reported in [1], the overall computational complexity of KGAT is:

$$O(|E_{ia}|d^2 + L|E_{uia}|d_l d_{l-1} + |E_{uia}|d) \tag{5.15}$$

where $d_0 = d$ is the initial embedding size, d_l is the embedding size at layer l of the GNN, and L is the number of layers in the GCN. The first term denotes the complexity of knowledge graph embedding learning through the TransR [100] principle. The second term represents the complexity of information propagation on the knowledge graph. The third term is the complexity of the final prediction step.

The computational complexity of MIA arises from two components. In the user-item view, the sparse matrix multiplication for the l-th layer in the LightGCN algorithm has computational complexity $O(|E_{ui}|d)$ (Eq. 5.3). The attribute view involves a simple lookup operation for the attribute latent representations and thus its complexity can be ignored. For the item-attribute alignment view, the computational complexity of the matrix multiplication operation in the *bi-interaction* association function is $O(|E_{ia}|d^2)$. The

complexity of the final prediction layer is $|E_{ui}|d$. In summary, the overall computation complexity of MIA(Bi-Interaction) is:

$$O(|E_{ia}|d^2 + L|E_{ui}|d) \tag{5.16}$$

Theoretically, we see that our method has reduced complexity compared to KGAT. In part, this is because we employ LightGCN, avoiding the $O(d^2)$ complexity of the convolution operations. However, an additional factor arises because MIA processes the user-item and item-attribute graphs separately, whereas KGAT operates over a combined user-item-attribute graph. Empirically, we compare the per epoch training time for both methods. All the experiments were conducted on a single GPU of NVIDIA P100 Pascal with 16G HBM2 memory. Results are reported in Table 5.5.

Table 5.5: Comparison of the per epoch training times over six datasets. Note that the sampling time is excluded from the results.

	Amazon CDs	Amazon Movies	Amazon Books	Amazon Sports	Yelp	LastFM
KGAT	250.3s	122.4s	217.3s	75.5s	278.7s	41.1s
MIA	**90.8s**	**50.5s**	**79.2s**	**33.6s**	**103.1s**	**20.2s**

As shown in Table.5.5, MIA is faster than KGAT in terms of training speed. For example, KGAT requires approximately 250.3s for each epoch while MIA only needs 90.8s per epoch for the Amazon-CDs dataset. Thus, we conclude that MIA has a better training efficiency than KGAT both theoretically and empirically.

Convergence Analysis

We now analyze the convergence speed of KGAT and MIA. In this analysis, we compare KGAT and MIA in terms of their training efficiency with respect to Recall@20. The results are shown in Figure 5.2.

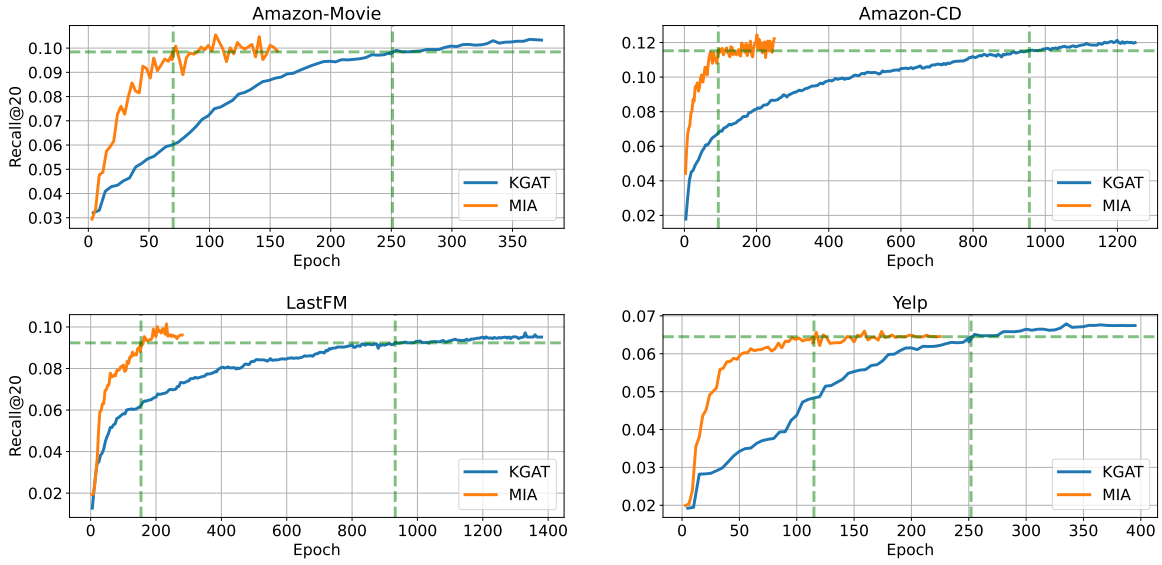


Figure 5.2: Convergence speed comparison between MIA and KGAT on four evaluation datasets *wrt* Recall@20. The horizontal dotted line denotes the 95% of KGAT's best Recall performance. The vertical dotted lines represent when both models reach within 5% of KGAT's best Recall for the first time.

From the results, we find that both models start from the same Recall@20. KGAT slowly converges to its best value while MIA converges much faster. Specifically, MIA settles to within 5% of KGAT's best Recall@20 at around epochs 80, 100, 180 and 120 on Amazon-Movies, Amazon-CDs, LastFM and Yelp, respectively, while KGAT reaches the same performance at around epoch 250, 950, 900, 250. Thus KGAT requires at least twice as many epochs as MIA for the four evaluation datasets.

To analyze the reason behind this significant difference in convergence speed, we compare the information flow in both methods. Recall that we denote the j-th user as u_j, the j-th item as i_j, and the j-th item side attribute as a_j.

In MIA, item side information directly interacts with items only, and this occurs in the item-attribute alignment component. The collaborative signal is propagated among user-item pairs in the bipartite graph in the user-item view. On the other hand, in KGAT, the collaborative signal is propagated over the entire knowledge graph, which leads to information flow from items to users. Therefore, multi-hop connectivity can only be de-

fined between users and items in MIA. By contrast, direct connections exist between all the entities (users, items, attributes) in KGAT. Taking a three-hop relation path for example, in MIA it can be defined as: $u_1 \rightarrow i_1 \rightarrow u_2 \rightarrow i_2$, whereas in KGAT it can be: $a_1 \rightarrow i_1 \rightarrow u_1 \rightarrow i_2$.

Considering the fact that item side information is mainly providing extra information about items, propagating item side information to users (or via users) is potentially inefficient. Overall, based on the results, the significant convergence difference demonstrates the efficiency and effectiveness of our proposed item-attribute alignment component.

5.5 Summary

In this chapter, a multi-view alignment-based recommendation framework MIA has been introduced to tackle the top-k recommendation problem given user interaction history and item side information. The extensive experiments have demonstrated the effectiveness and efficiency of MIA. In particular, a thorough comparison to a broad selection of baselines and a careful complexity and convergence rate analysis on KGAT and MIA algorithms have shown that MIA is able to achieve state-of-the-art performance, at least comparable to KGAT, with a notable speed improvement.

Chapter 6

Conclusions

The project we presented in this thesis contributes to the field of recommender systems. Specifically, the focus is on the incorporation of item side information in state-of-the-art recommendation algorithms.

The need to develop side information-based enhanced recommendation systems arises from the increasing collection of information about users and items. However, relative to the number of new side information enhanced recommendation algorithms that have been developed, there has been considerably less work investigating the key differences between these algorithms. Therefore, the first contribution I make in this thesis is to develop a unified framework named AIA to analyze these algorithms. The framework identifies two key components that arise in most of the state-of-the-art algorithms for incorporating item side information: (i) an item-attribute association function that describes how side information is absorbed into item representations, and (ii) an association loss function that is used to train the association function. Although recommendation architectures have been presented in very different ways (e.g., knowledge graph versus multi-view), we show in Chapter 4 that by re-expressing them in the AIA framework, we can expose their key differences, and these differences lie in how the association function is defined and the loss function used to optimize it.

After presenting the AIA framework, we develop a multi-view item side information enhanced recommendation algorithm called MIA, which is the second contribution of this thesis. MIA performs item-attribute association by maximizing the mutual information between related item-attribute pairs. We implemented the proposed method, along with state-of-the-art baseline models in PyTorch and conducted extensive experiments to evaluate the effectiveness of our method. The results demonstrate that the proposed method achieves state-of-the-art recommendation accuracy, achieving a small, but statistically significant improvement compared to the best baseline model for both recall@20 and ndcg@20 across most of the evaluation datasets. We present a complexity and convergence speed analysis for further comparison between MIA and the best performing baseline, KGAT. The results show that not only does MIA achieve a small improvement in accuracy compared to KGAT, but its computational complexity is smaller, leading to convergence speed. Therefore, the overall training time required for MIA is less than half of that required for KGAT.

Future work

The proposed method can be extended in multiple ways. First, our method only focuses on one type of side information input, the item side information. A natural extension is to include other types of side information, particularly user attributes. However, considering more types of input is a more challenging task. Theoretically, MIA can support numerous views. It would be interesting to explore MIA with extra views and see how it affects the performance. For example, given users' profiles, extra views (e.g., a user-side attribute view, a user-attribute association view) can be added to increase the system's understanding of users. This may have a positive impact on the system's performance.

Another interesting direction of future work would be to explore other aggregation methods such that a more complex scenario can be handled. For example, given an item i and its set of attributes $C_i = \{$Beauty & Personal Care, Skin Care, Body, Cleansers, Shower Gels$\}$, there exists a hierarchical relationship within the attribute data. Current

implementation simply aggregates item attributes by a summation operation, *i.e.* , each attribute contributes the same amount to the final attribute representation. To explore the hierarchical information in the attribute set, one strategy is to apply an attention mechanism, which learns to assign a higher attention weight to an attribute that is more indicative of an item's characteristics [104–106].

Bibliography

[1] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: Knowledge Graph Attention Network for Recommendation," in *Proc. ACM Conf. Knowledge Discovery and Data Mining*, Anchorage, AK, USA, Jul. 2019.

[2] X. He and T.-S. Chua, "Neural Factorization Machines for Sparse Predictive Analytics," in *Proc. ACM Conf. Research and Development in Information Retrieval (SIGIR)*, Tokyo, Japan, Aug. 2017.

[3] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," in *Prod. Int. Conf. Machine Learning (ICML)*, Virtual, Jul. 2021.

[4] G. Linden, B. Smith, and J. York, "Amazon.com Recommendations: Item-to-Item Collaborative Filtering," *J. IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, Jan. 2003.

[5] Q. Zhao, Y. Zhang, D. Friedman, and F. Tan, "E-commerce Recommendation with Personalized Promotion," in *Proc. ACM Conf. Recommender Systems*, TU Wien, Austria, Sep. 2015.

[6] P. Covington, J. Adams, and E. Sargin, "Deep Neural Networks for YouTube Recommendations," in *Proc. ACM Conf. Recommender Systems*, Boston, MA, USA, Sep. 2016.

[7] A. Anderson, L. Maystre, I. Anderson, R. Mehrotra, and M. Lalmas, "Algorithmic Effects on the Diversity of Consumption on Spotify," in *Proc. Int. Conf. World Wide Web*, Taipei, Taiwan, Apr. 2020.

[8] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. ACM Conf. Knowledge Discovery and Data Mining*, Las Vegas, NV, USA, Aug. 2008.

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural Collaborative Filtering," in *Proc. Int. Conf. World Wide Web*, Perth, Australia, Apr. 2017.

[10] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational Autoencoders for Collaborative Filtering," in *Proc. Int. Conf. World Wide Web*, Geneva, Switzerland, Apr. 2018.

[11] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders Meet Collaborative Filtering," in *Proc. Int. Conf. World Wide Web*, Florence, Italy, May 2015.

[12] N. Dehmamy, A.-L. Barabasi, and R. Yu, "Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, Canada, 2019.

[13] M. Wang, Y. Lin, G. Lin, K. Yang, and X.-m. Wu, "M2GRL: A Multi-task Multi-view Graph Representation Learning Framework for Web-scale Recommender Systems," in *Proc. ACM Conf. Knowledge Discovery and Data Mining*, San Diego, CA, USA, Aug. 2020.

[14] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation," in *Proc. ACM Conf. Research and Development in Information Retrieval (SIGIR)*, Virtual , Jul. 2020.

[15] J. Sun, Y. Zhang, C. Ma, M. Coates, H. Guo, R. Tang, and X. He, "Multi-graph Convolution Collaborative Filtering," in *Proc. IEEE Int. Conf. Data Mining*, Beijing, China, Nov. 2019.

[16] N. Golbandi, Y. Koren, and R. Lempel, "Adaptive bootstrapping of recommender systems using decision trees," in *Proc. ACM Conf. Web Search and Data Mining*, San Diego, CA, USA, Feb. 2011.

[17] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey state-of-the-art and possible extensions," *J. IEEE Transactions Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[18] X. Du, X. Wang, X. He, Z. Li, J. Tang, and T.-S. Chua, "How to Learn Item Representation for Cold-Start Multimedia Recommendation?" in *Proc. ACM Conf. Multimedia*, Virtual, Oct. 2020.

[19] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," in *Proc. Int. Conf. Artificial Intell.*, Melbourne, Australia, Aug. 2017.

[20] F. Vasile, E. Smirnova, and A. Conneau, "Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation," in *Proc. ACM Conf. Recommender Systems*, Boston, MA, USA, Sep. 2016.

[21] K. Zhou, H. Wang, W. X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen, "S3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization," in *Proc. ACM Conf. Information and Knowledge Management*, Virtual, Oct. 2020.

[22] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *Proc. Int. Conf. Learning Representations (ICLR)*, Vancouver, Canada, Sep. 2018.

[23] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep Graph Infomax," in *Proc. Int. Conf. Learning Representations (ICLR)*, New Orleans, LA, USA, Sep. 2018.

[24] T. Zhang, M. Wang, J. Xi, and A. Li, "LPGNMF: Predicting Long Non-Coding RNA and Protein Interaction Using Graph Regularized Nonnegative Matrix Factorization," *J. IEEE Transactions Computational Biology and Bioinformatics*, vol. 17, no. 1, p. 189–197, Jan. 2020.

[25] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph Neural Networks for Social Recommendation," in *Proc. Int. Conf. World Wide Web*, San Francisco, CA, USA, May 2019.

[26] X. Chen and L. Pan, "A Survey of Graph Cuts/Graph Search Based Medical Image Segmentation," *J. IEEE Reviews in Biomedical Engineering*, vol. 11, pp. 112–124, Jan. 2018.

[27] W. L. Hamilton, "Representation Learning on Graphs: Methods and Applications." *J. IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, 2017.

[28] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proc. Int. Conf. World Wide Web*, New York, NY, USA, May 2013.

[29] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning Graph Representations with Global Structural Information," in *Proc. Int. Conf. Information and Knowledge Management*, Melbourne, Australia, Oct. 2015.

[30] A. Grover and J. Leskovec, "Node2vec: Scalable Feature Learning for Networks," in *Proc. ACM Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, Aug. 2016.

[31] W. L. Hamilton, *Graph Representation Learning*. Morgan and Claypool, 2020.

[32] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. ACM Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, Aug. 2014.

[33] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale Information Network Embedding," in *Proc. Int. Conf. World Wide Web*, Geneva, Switzerland, May 2015.

[34] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral Networks and Locally Connected Networks on Graphs," in *Proc. Int. Conf. Learning Representations (ICLR)*, Banff, Canada, May 2014.

[35] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017.

[36] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning Convolutional Neural Networks for Graphs," in *Prod. Int. Conf. Machine Learning (ICML)*, Jun. 2016.

[37] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *J. IEEE Transactions Neural Networks*, vol. 20, no. 1, p. 61–80, Jan. 2009.

[38] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," in *Proc. ACM Conf. Knowledge Discovery and Data Mining*, London, UK, Jul. 2018.

[39] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, Red Hook, NY, USA, Dec. 2016.

[40] S. Tang, B. Li, and H. Yu, "ChebNet: Efficient and Stable Constructions of Deep Neural Networks with Rectified Power Units using Chebyshev Approximations," *arXiv: 1911.05467*, Dec. 2019.

[41] A. Micheli, "Neural network for graphs: A contextual constructive approach," *J. IEEE Transactions Neural Networks*, vol. 20, no. 3, p. 498–511, Mar. 2009.

[42] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional Networks on Graphs for Learning Molecular Fingerprints," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, Canada, Dec. 2015.

[43] H. Gao, Z. Wang, and S. Ji, "Large-Scale Learnable Graph Convolutional Networks," in *Proc. ACM Conf. Knowledge Discovery and Data Mining*, London, UK, Jul. 2018.

[44] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Red Hook, NY, USA, Dec. 2016.

[45] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017.

[46] J. Shang and M. Sun, "Geometric Hawkes Processes with Graph Convolutional Recurrent Neural Networks," in *Proc. AAAI Conf. Artificial Intell.*, Honolulu, HI, USA, Jul. 2019.

[47] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based Recommendation with Graph Neural Networks," in *Proc. AAAI Conf. Artificial Intell.*, Honolulu, HI, USA, Jul. 2019.

[48] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-Based Social Recommendation via Dynamic Graph Attention Networks," in *Proc. ACM Conf. on Web Search and Data Mining*, Melbourne, Australia, Jan. 2019.

[49] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous Graph Attention Network," in *Proc. Int. Conf. World Wide Web*, New York, NY, USA, May 2019.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *Proc. Int. Conf. on Learning Representations*, New Orleans, LA, USA, Feb. 2018.

[51] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs," in *Conf. Uncertainty in Artificial Intell.*, Monterey, CA, USA, Mar. 2018.

[52] J. Sun, Y. Zhang, W. Guo, H. Guo, R. Tang, X. He, C. Ma, and M. Coates, "Neighbor Interaction Aware Graph Convolution Networks for Recommendation," in *Proc. ACM Conf. Research and Development in Information Retrieval (SIGIR)*, Virtual , Jul. 2020.

[53] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2013.

[54] J. A. Caliwag, R. A. Pagaduan, F. C. Reyes, J. P. C. Olos, and R. Castillo, "TrackMe: A Recommender System for Preschools in Quezon City using Content-based Algorithm," in *Proc. Int. Conf. on Information Science and Systems*, Tokyo, Japan, Mar. 2019.

[55] X. Wang and Y. Wang, "Improving Content-based and Hybrid Music Recommendation using Deep Learning," in *Proc. ACM Conf. Multimedia*, New York, NY, USA, Nov. 2014.

[56] H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proc. Int. Conf. Artificial Intell.*, Melbourne, Australia, Aug. 2017.

[57] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. Conf. Uncertainty in Artificial Intell.*, Arlington, Virginia, USA, Jun. 2009.

[58] J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, and J. Tang, "NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization," in *Proc. Int. Conf. World Wide Web*, New York, NY, USA, May 2019.

[59] X. Chen, C. Du, X. He, and J. Wang, "JIT2R: A Joint Framework for Item Tagging and Tag-based Recommendation," in *Proc. ACM Conf. Research and Development in Information Retrieval (SIGIR)*, Virtual , Jul. 2020.

[60] S. Rendle, "Factorization Machines," in *Proc. IEEE Int. Conf. Data Mining*, Sydney, Australia, Dec. 2010.

[61] P. Forbes and M. Zhu, "Content-boosted matrix factorization for recommender systems: Experiments with recipe recommendation," in *Proc. ACM Conf. Recommender Systems*, New York, NY, USA, Oct. 2011.

[62] H. Zhang, I. Ganchev, N. S. Nikolov, Z. Ji, and M. O'Droma, "FeatureMF: An Item Feature Enriched Matrix Factorization Model for Item Recommendation," *J. IEEE Access*, vol. 9, pp. 65 266–65 276, 2021.

[63] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide and Deep Learning for Recommender Systems," in *Proc. Workshop on Deep Learning for Recommender Systems*, New York, NY, USA, Sep. 2016.

[64] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems," in *Proc. ACM Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, Jul. 2018.

[65] J. Han, Y. Ma, Q. Mei, and X. Liu, "DeepRec: On-device Deep Learning for Privacy-Preserving Sequential Recommendation in Mobile Commerce," in *Proc. Web Conf.*, Virtual, Apr. 2021.

[66] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative Denoising Auto-Encoders for Top-N Recommender Systems," in *Proc. ACM Conf. on Web Search and Data Mining*, San Francisco, CA, USA., Feb. 2016.

[67] N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi, "Sequential Variational Autoencoders for Collaborative Filtering," in *Proc. ACM Conf. on Web Search and Data Mining*, Huston, TX, USA, Jan. 2019.

[68] R. van den Berg, T. N. Kipf, and M. Welling, "Graph Convolutional Matrix Completion," in *Proc. ACM Conf. Knowledge Discovery and Data Mining*, London, UK, Aug. 2018.

[69] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural Graph Collaborative Filtering," in *Proc. ACM Conf. Research and Development in Information Retrieval (SIGIR)*, Paris, France, Jul. 2019.

[70] R. Sun, X. Cao, Y. Zhao, J. Wan, K. Zhou, F. Zhang, Z. Wang, and K. Zheng, "Multimodal Knowledge Graphs for Recommender Systems," in *Proc. ACM Conf. on Information & Knowledge Management*, Virtual, Oct. 2020.

[71] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *J. Neural Computation*, vol. 7, no. 6, pp. 1129–1159, 1995.

[72] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. MA, USA:MIT Press: Cambridge, 2003.

[73] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual Information Neural Estimation," in *Prod. Int. Conf. Machine Learning (ICML)*, Stockholm, Sweden, Jul. 2018.

[74] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization," in *Proc. Int. Conf. Learning Representations (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020.

[75] M. D. Donsker and S. R. S. Varadhan, "Asymptotic evaluation of certain markov process expectations for large time. IV," *Communications on Pure and Applied Mathematics*, vol. 36, no. 2, pp. 183–212, 1983.

[76] S. Nowozin, B. Cseke, and R. Tomioka, "*F*-GAN: Training generative neural samplers using variational divergence minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016.

[77] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Prod. Int. Conf. Artificial Intell. and Statistics (AISTATS)*, Virtual, 2010.

[78] A. van den Oord, Y. Li, and O. Vinyals, "Representation Learning with Contrastive Predictive Coding," *arXiv:1807.03748*, Jan. 2019.

[79] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning Representations by Maximizing Mutual Information Across Views," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, Canada, Dec. 2019.

[80] Y. Tian, D. Krishnan, and P. Isola, "Contrastive Multiview Coding," in *Proc. Int. Conf. Learning Representations (ICLR)*, New Orleans, LA, USA, Sep. 2019.

[81] A. Sankar, Y. Wu, Y. Wu, W. Zhang, H. Yang, and H. Sundaram, "GroupIM: A Mutual Information Maximization Framework for Neural Group Recommendation," in *Proc. ACM Conf. Research and Development in Information Retrieval (SIGIR)*, Virtual, Jul. 2020.

[82] J. Cao, X. Lin, S. Guo, L. Liu, T. Liu, and B. Wang, "Bipartite Graph Embedding via Mutual Information Maximization," in *Proc. ACM Int. Conf. Web Search and Data Mining*, Virtual, Mar. 2021.

[83] H. Hotelling, "Relations Between Two Sets Of Variates," *Biometrika*, vol. 28, no. 3/4, pp. 321–377, Dec. 1936.

[84] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, "Deep Canonical Correlation Analysis," in *Prod. Int. Conf. Machine Learning (ICML)*, Atlanta, USA, May 2013.

[85] B. Tan, E. Zhong, E. W. Xiang, and Q. Yang, "Multi-transfer: Transfer learning with multiple views and multiple sources," *Statistical Analysis and Data Mining*, vol. 7, no. 4, Aug. 2014.

[86] Y. Li, M. Yang, and Z. Zhang, "A Survey of Multi-View Representation Learning," *J. IEEE Transactions Knowledge and Data Engineering*, vol. 31, no. 10, pp. 1863–1883, Oct. 2019.

[87] C. Xu, D. Tao, and C. Xu, "A Survey on Multi-view Learning," *arXiv:1304.5634*, Apr. 2013.

[88] Y. Zhang, Q. Ai, X. Chen, and W. B. Croft, "Joint Representation Learning for Top-N Recommendation with Heterogeneous Information Sources," in *Proc. ACM Conf. Information and Knowledge Management*, Singapore, Singapore, Nov. 2017.

[89] Q. W. Y Guan and G. Chen, "Deep learning based personalized recommendation with multi-view information integration," *Decision Support Systems*, vol. 118, pp. 58–69, Mar. 2019.

[90] T. Liang, L. Zheng, L. Chen, Y. Wan, P. S. Yu, and J. Wu, "Multi-view factorization machines for mobile app recommendation based on hierarchical attention," *Knowledge-Based Systems*, vol. 187, no. 104821, p. 1–11, Jan. 2020.

[91] D. Li, N. Dimitrova, M. Li, and I. K. Sethi, "Multimedia content processing through cross-modal association," in *Proc. ACM Conf. Multimedia*, New York, NY, USA, Nov. 2003.

[92] A. M. Elkahky, Y. Song, and X. He, "A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems," in *Proc. Int. Conf. World Wide Web*, Florence, Italy, May 2015.

[93] J.-J. Cai, J. Tang, Q.-G. Chen, Y. Hu, X. Wang, and S.-J. Huang, "Multi-View Active Learning for Video Recommendation," in *Proc. Int. Conf. Artificial Intell.*, Macao, China, Aug. 2019.

[94] Y. Chen and M. de Rijke, "A Collective Variational Autoencoder for Top-N Recommendation with Side Information," in *Proc. Workshop on Deep Learning for Recommender Systems*, Vancouver, Canada , Oct. 2018.

[95] S. Liu, I. Ounis, C. Macdonald, and Z. Meng, "A Heterogeneous Graph Neural Model for Cold-start Recommendation," in *Proc. ACM Conf. Research and Development in Information Retrieval (SIGIR)*, Virtual, Jul. 2020.

[96] Y. Zheng, B. Mobasher, and R. Burke, "CSLIM: Contextual SLIM recommendation algorithms," in *Proc. ACM Conf. on Recommender Systems*, Silicon Valley, CA, USA, Oct. 2014.

[97] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Netoworks," in *Proc. Int. Conf. Learning Representations (ICLR)*, Toulon, France, Apr. 2017.

[98] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative Filtering beyond the User-Item Matrix: A Survey State Art and Future Challenges," *ACM Computing Surveys*, vol. 47, no. 3, p. 1–45, May 2014.

[99] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *Proc. ACM Conf. Research and Development in Information Retrieval (SIGIR)*, Beijing, China, Jul. 2011.

[100] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proc. AAAI Conf. Artificial Intell.*, Austin, TX, USA, Jan. 2015.

[101] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational Autoencoders for Collaborative Filtering," in *Proc. Int. Conf. World Wide Web*, Geneva, Switzerland, Apr. 2018.

[102] C. Feng, Z. Liu, S. Lin, and T. Q. Quek, "Attention-based Graph Convolutional Network for Recommendation System," in *Int. Conf. Acoustics, Speech and Signal Processing*, Brighton, UK, May 2019.

[103] S. Kokoska and C. Nevison, *Critical Values For The Wilcoxon Signed-Rank Statistic*, 1989, vol. Statistical Tables and Formulae.

[104] W. Huang, E. Chen, Q. Liu, Y. Chen, Z. Huang, Y. Liu, Z. Zhao, D. Zhang, and S. Wang, "Hierarchical Multi-label Text Classification: An Attention-based Recurrent Network Approach," in *Proc. ACM Conf. on Information and Knowledge Management*, Beijing, China , Nov. 2019.

[105] D. Zhang, S. Zhao, Z. Duan, J. Chen, Y. Zhang, and J. Tang, "A Multi-Label Classification Method Using a Hierarchical and Transparent Representation for Paper-Reviewer Recommendation," *ACM Transactions on Information Systems*, vol. 38, no. 5, p. 1–20, Feb. 2020.

[106] P. Yao, Q. Peng, and T. Han, "Hierarchical Label Embedding Networks for Financial Document Sentiment Analysis," in *Proc. Int. Conf. Computing and Artificial Intell.*, Sinapore, Sinapore, Apr. 2020.