Effective Graph Representation Learning for Ranking-based Recommendation

SIWEI LIU

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

School of Computing Science
College of Science and Engineering
University of Glasgow

August 2022

Abstract

Ranking-based recommender systems are designed to generate a personalised ranking list of items for a given user to address the information overload problem. An effective and efficient ranking-based recommender system can benefit users by providing them with items of interest as well as service providers by increasing their exposure and profits. Since more and more users and providers of items have been increasingly interacting with online platforms, the underlying recommendation algorithms are facing more challenges. For example, traditional collaborative filtering-based recommender systems cannot generate effective recommendations to *cold-start* users due to the lack of sufficient interactions. In addition, although recommender systems can leverage deep learning-based techniques to enhance their effectiveness, they are not robust enough against variances in the models' initialisations, which can degrade the users' satisfaction. Furthermore, when incorporating these complex deep models, the training phases of recommender systems become less efficient, which might slower the online platforms from quickly capturing the users' interests.

Graph representation learning includes techniques that can leverage graph-structured data and generate latent representations for the nodes, graphs/sub-graphs and edges between nodes. Since the user-item interaction matrix is in fact a bipartite graph, we can use these graph-based techniques to leverage the interaction matrix and generate more effective node representations for the users and items. Therefore, this thesis aims to enhance the ranking-based recommendations by proposing novel recommender systems based on graph representation learning. In particular, this thesis uses heterogeneous graph representation learning, graph pre-training and graph contrastive learning to improve the effectiveness of ranking-based recommendations while alleviating the aforementioned *cold-start* problem as well as the low-robustness and low training-efficiency issues.

To enhance the effectiveness of ranking-based recommendations and alleviate the *cold-start* problem, we propose to use the heterogeneous graph representation learning technique to encode the typical side information of the users and items, which are usually defined as the attributes of users and the descriptions of items. For example, a user-item interaction matrix, social relations are one of the most naturally available relations that can be used to enrich such an interaction matrix. Therefore, we choose the social relations among different types of side information to build the heterogeneous graph. We propose a novel recommender system, the Social-aware

Gaussian Pre-trained model (SGP), which encodes the user social relations and interaction data using the heterogeneous graph representation learning technique. Next, in the subsequent fine-tuning stage, our SGP model adopts a Gaussian Mixture Model (GMM) to factorise these pre-trained embeddings for further training. Our extensive experiments on three public datasets show that SGP can alleviate the *cold-start* problem while also ensuring effective recommendations for *regular* users.

To alleviate the low-robustness issue and enhance the recommendation effectiveness, we propose to leverage multiple types of side information using the graph pre-training technique. In particular, we aim to generalise the pre-training technique used by SGP for multiple types of side information associated with both users and items. Specifically, we propose two novel pre-training schemes, namely Single-P and Multi-P, to leverage side information such as the ages and occupations of users and the textual reviews and categories of items. Instead of jointly training with two objectives, our pre-training schemes first pre-train a representation model under the users and items' multi/single relational graphs constructed by their side information and then fine-tune their embeddings under an existing general representation-based recommendation model. Extensive experiments on three public datasets show that the graph pre-training technique can effectively enhance the effectiveness of ranking-based recommender systems and alleviates the *cold-start* problem. In addition, our pre-training schemes can provide more effective initialisations for both the users and items; hence the robustness of fine-tuning models namely MF, NCF, NGCF and LightGCN, can be improved.

Finally, to enhance the training efficiency of graph-based recommenders while ensuring their effectiveness, we propose to use the graph contrastive learning technique to improve the traditional random negative sampling approach. In particular, we propose a dynamic negative sampling (DNS) approach that leverages the graph contrastive learning technique to replace the randomly sampled negative items with more informative negative items. Our experiments show that DNS can improve the recommendation effectiveness of four competitive recommenders. Next, we further propose a novel graph-based model, i.e. MLP-CGRec, that leverages a multiple sampling approach to enhance the training efficiency of the graph-based recommender system. In particular, MLP-CGRec uses DNS to sample contrastive negative items and an efficient graph-based sampling method to select pseudo-positive samples. Experimental results on three public datasets show that MLP-CGRec can maintain competitive effectiveness and achieve the best efficiency compared with state-of-the-art recommender systems.

Acknowledgements

I came to Glasgow in 2014 for my bachelor's degree in Electrical and Electronic Engineering. By that time, I could never imagine that I would one day obtain a PhD in such a challenging subject. But, finally, I made it. I want to take this opportunity to thank the people who made this thesis possible and an exciting experience for me.

First and foremost, I would like to express my sincere gratitude to my PhD supervisors, Iadh Ounis and Craig Macdonald for their patience, enthusiasm, and immense knowledge. They provide me with guidance, support and encouragement not only in my PhD research but also in my life. Without their help, this work would not have been possible.

I would also like to thank my colleagues and friends at the Terrier team and the school of computing science for their collaboration and support: Zaiqiao Meng, Xi Wang, Jingming Huang, Xin Xin, Xiao Wang, Zixuan Yi, Graham MacDonald, Richard McCreadie, Yaxiong Wu, Anjie Fang, Xinyu Yang, Yanni Ji and many others. I had lots of fun with them. In particular, Zaiqiao, Xi and Jingming have provided much meaningful feedback to me. It has been an honour and a pleasure to work with them.

I am also thankful to my office mates: Jingming, Florian and Shameem. I will never forget our happy time in office G132. Besides, playing Pokemon with Cong Fu, Qianyu Long, Kai Feng, and Zhaohan Meng is a great pleasure for me. More importantly, no PhD can succeed without a healthy body. Therefore, I have to thank my hiking, badminton, swimming, gym and cycling buddies: Jing Zeng, Debao Guan, Liang Min, Shiyang Liang, Zhengxing Xie, Yiming Zhao, Shengyuan Zhu, Xiaogeng Wang and many others.

Last but not least, I am most grateful to my parents, Su Wang and Dianjun Liu, my fiancee, Yafang Huang, and other family members for their endless support and love throughout my life.

Contents

Abstract i

Acknowledgements iii

1 Introduction 1

1.1 Motivation . . . 1

1.2 Thesis Statement . . . 5

1.3 Contributions . . . 6

1.4 Origins of Material . . . 8

1.5 Thesis Outline . . . 9

2 Background 12

2.1 Overview of Recommender Systems . . . 12

2.1.1 Basic Models for Recommender Systems . . . 16

2.1.1.1 Collaborative Filtering-based Recommender Systems . . . 16

2.1.1.2 Content-based Recommender Systems . . . 18

2.1.1.3 Hybrid Recommender Systems . . . 20

2.1.2 Evaluation of Recommender Systems . . . 20

2.1.2.1 Offline Evaluation Methodology . . . 21

2.1.2.2 Evaluation Metrics . . . 22

2.2 Graph Representation Learning . . . 24

2.2.1 Preliminaries . . . 25

2.2.2 Heterogeneous Graph Representation Learning . . . 27

2.2.3 Graph Pre-training . . . 28

2.2.4 Graph Contrastive Learning . . . 29

2.3 Conclusions . . . 30

3 Social Graph Recommendation 31

3.1 Introduction . . . 31

3.2 Related Work . . . 33

3.2.1 Pre-trained Models . . . 33

3.2.2 Graph-based Recommendation 34
3.2.3 Data Augmentation and Cross-Domain Recommendation 35
3.3 Model Architecture 37
3.3.1 Preliminaries 37
3.3.2 The Pre-training Stage 38
3.3.3 The Fine-tuning Stage 40
3.3.3.1 Information Distillation Stage 41
3.3.3.2 Model Training and Recommendation 42
3.4 Datasets and Experimental Setup 43
3.4.1 Baselines 43
3.4.2 Evaluation Methodology 45
3.5 Results Analysis 47
3.5.1 RQ3.1: Pre-trained Recommendation Performances 47
3.5.2 RQ3.2: GMM Information Distillation 48
3.5.3 RQ3.3: *Cold-start* Performances 49
3.5.4 RQ3.4: Impact of Social Relations 51
3.5.5 RQ3.5: Hyperparameter Analysis 52
3.5.6 Embedding Visualisation 52
3.6 Conclusions 53

4 Graph Pre-training for Recommendations **55**
4.1 Introduction 55
4.2 Side Information for Recommender Systems 59
4.3 Preliminaries 60
4.4 Methodology 61
4.4.1 The Pre-training Scheme for Recommendation 61
4.4.2 Pre-training on Single-Graphs 62
4.4.3 Pre-training on Multi-Graphs 64
4.4.4 Pre-training Loss 65
4.4.5 Complexity Analysis 65
4.4.6 Fine-tuning with Existing Recommenders 66
4.4.7 Discussion 67
4.5 Experimental Setup 68
4.5.1 Research Questions 68
4.5.2 Datasets 69
4.5.3 Experimental Settings 70
4.6 Results and Analysis 72
4.6.1 Effectiveness of Pre-training 73
4.6.2 Effectiveness of Integrating Side Information 74

4.6.3 Ablation Study of Side Information . . . 76
4.6.4 Stability Analysis . . . 77
4.6.5 *Cold-start* Analysis . . . 78
4.6.6 Hyperparameter Analysis . . . 79
4.7 Conclusions . . . 80

5 Graph Contrastive Negative Sampling for Recommendations 82
5.1 Introduction . . . 82
5.2 Negative Sampling in Recommender Systems . . . 84
5.3 Methodology . . . 85
5.3.1 Contrastive Recommendation . . . 85
5.3.2 Dynamic Negative Sampling . . . 87
5.3.3 Model Prediction . . . 89
5.3.4 Entropy Analysis . . . 89
5.4 Datasets and Experimental Setup . . . 90
5.4.1 Baselines . . . 91
5.4.2 Evaluation Methodology . . . 92
5.5 Results Analysis . . . 93
5.5.1 Contrastive Negative Items . . . 93
5.5.2 Effectiveness of DNS . . . 94
5.5.3 Hyperparameter Analysis . . . 96
5.5.4 Entropy Analysis . . . 97
5.6 Conclusions . . . 97

6 Contrastive Multiple Sampling for Graph Recommendations 99
6.1 Introduction . . . 99
6.2 Multilayer Perceptron-based Models . . . 101
6.3 Methodology . . . 101
6.3.1 Graph Neighbourhood Construction . . . 102
6.3.2 Contrastive Negative Sampling . . . 103
6.3.3 MLP-based Learning . . . 103
6.3.4 Contrastive Training of MLP-CGRec . . . 104
6.4 Datasets and Experimental Setup . . . 104
6.5 Results . . . 106
6.5.1 Effectiveness and Efficiency . . . 106
6.5.2 Impacts of Samples and Components . . . 106
6.6 Conclusions . . . 108

7 Integrated Graph Recommender Systems **110**

7.1 Introduction 110

7.2 Method 112

7.2.1 Pre-training with Dynamic Negative Sampling 113

7.2.2 Pre-training with Multiple Sampling 114

7.3 Experiments 114

7.4 Results 115

7.4.1 Effectiveness of the Integrated Models 115

7.4.2 Detailed Performances of the Integrated Models 116

7.5 Conclusions 120

8 Conclusions and Future Work **121**

8.1 Contributions and Conclusions 121

8.2 Directions for Future Work 125

8.3 Concluding Remarks 126

List of Tables

3.1 Main Notations Used in this chapter. 37

3.2 Statistics of datasets. 44

3.3 Performances of SGP and other baselines on the three used datasets. All metrics are computed at rank cutoff 10. The best and second best performances are highlighted in boldface and underlined, respectively; * denotes a significant difference between the performance of SGP and that of the baselines according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$. The 'Social' column indicates whether a model uses social relations or not. 47

3.4 NDCG@10 performances of our SGP model in comparison to the LightGCN baseline across different user groups where n is the number of users' interactions; * denotes a significant difference versus LightGCN (paired t-test, p<0.01). 49

3.5 Performances of SGP (Pre-training) on the *extreme cold-start* users in comparison with the random and popularity-based baselines. In the table, SGP is the only approach where interactions are used. * and $\uparrow$ denote significant differences compared to the random and popularity baselines, respectively (paired t-test, p<0.01). 50

4.1 Statistics of the datasets. 69

4.2 Performance comparison of recommenders with side information. The best and second best performances are marked in boldface or underlined, respectively. We use $*$ to denote a significant difference between the performances of the side information-aware baselines and the best proposed model i.e., LightGCN$_{+\textbf{Multi-P}}$, according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$. 74

4.3 Standard deviations (denoted std.) and means of the NDCG@10 performances over 50 random seeds. Lower standard deviation (in bold) means a better stability. 76

4.4 NDCG@10 performances of our proposed scheme and those of the baselines across different groups of users on the two used datasets. 79

5.1 Statistics of the Amazon-Instant-Video dataset. 90

5.2 Performances and false-negative rates of BPR when using a static random sampler and three dynamic samplers on the three used datasets. The NDCG metric is computed at rank cutoff 10. F-N rate denotes the false-negative rate and * denotes a significant difference between the performance of the dynamic dissimilar sampler and that of other samplers according to the paired t-test with the Holm-Bonferroni correction for p <0.01. The best result and the lowest F-N rate are highlighted in bold. 93
5.3 Performances of two advanced NS-based baselines and four general recommenders with their DNS variants on the three used datasets. All metrics are computed at rank cutoff at 10. The best result is highlighted in bold and the largest percentage improvement is underlined; * denotes a significant difference between the performance of a DNS variant and that of the baseline, while † denotes a significant difference among the four DNS-based models, according to the paired t-test with the Holm-Bonferroni correction for p <0.01. 94

6.1 Experimental results of MLP-CGRec and other baselines on the three used datasets w.r.t. HR@10, NDCG@10, max GPU memory consumption (gigabytes), average epoch time (seconds) and total training time (minutes). The best effectiveness is highlighted in bold and the second best result is highlighted with underline. * denotes a significant difference compared to the result of MLP-CGRec using the TOST (two one-sided test) with p<0.05. 107

7.1 An effectiveness comparison of the integrated models with all baselines. The best and second best effectiveness are marked in boldface or underlined, respectively. We use $*$ to denote a significant difference between the best performing model and the rest, according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$. In addition, we use † to denote a significant difference between the best performing integrated model and other integrated models, according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$. . 116
7.2 NDCG@10 performances of four integrated approaches and our proposed models across different groups of users on the two used datasets. We use $*$ to denote a significant difference between the best performing model i.e., **Multi-P**+DNS, and the rest, according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$. 117
7.3 Standard deviations (denoted std.) and means of the NDCG@10 performances over 50 random seeds. A lower standard deviation indicates a better stability. The best and second best values are marked in boldface or underlined, respectively 117

7.4 Efficiencies comparison of four integrated models with baselines, where Memory, Epoch, Total denote the maximum memory consumption (gigabyte), average epoch time (second) and total training time (minute). The best and second best efficiencies are marked in boldface or underlined, respectively. 119

List of Figures

2.1 *Explicit* and *implicit* feedback of users. 13

2.2 An overview of matrix completion and sequential recommender systems. . . . 14

3.1 An illustration of our SGP model, where the pre-training stage and the fine-tuning stage are located above and below, respectively. 36

3.2 Max/Min/Mean values of NDCG@10 for SGP on 3 datasets with different number of Gaussian distributions. 49

3.3 An ablation study of performances of SGP and Diffnet++ (different proportions of social relations are masked out). 51

3.4 The performances comparison between SGP and LightGCN over different dimensions and different cut-offs on the Yelp dataset. 52

3.5 The t-SNE plot of all users' embeddings of the Librarything dataset obtained from the MF model (a) and our SGP model (b), where the red dot represents an anchor user, the red triangles are this user's friends and the red start is the target item. The similar configuration is also applied to another two users with their corresponding friends and target items, which are plotted with the colour of green and orange, respectively. 54

4.1 Box-and-whisker diagrams for the NDCG performances of the MF (Rendle et al., 2020) and LightGCN (He et al., 2020) models on the Epinions and Foursquare datasets. 57

4.2 An overview of our graph neural pre-training/fine-tuning scheme. Our pre-training model constructs relational graphs based on the feature of entities, and pre-trains the embeddings of entities by using **Multi-P** or **Single-P**. 60

4.3 An illustration describing the input features, single-relational graphs and output entity embeddings of **Single-P**. 62

4.4 An illustration describing the input features, multi-relational graphs and output entity embeddings of **Multi-P**. 63

4.5 Performance comparison of the four selected existing recommenders with the **Single-P** and **Multi-P** pre-training processes. We use $*$ to denote a significant difference between the performances of the baselines and their pre-trained variants, according to the paired t-test with the Holm-Bonferroni correction for p<0.01. 73
4.6 Results of the baselines deploying the **Multi-P** and **Single-P** models over different dropout ratio of entity features. 77
4.7 The performance comparison over different dimensions on the Foursquare dataset. 80
4.8 The performance comparison over different cut-off values on the Foursquare dataset. 80

5.1 Comparison of our proposed DNS scheme with BPR. 85
5.2 NDCG@10 performances of the BPR$_{\text{DNS}}$ model over (a) different update intervals (n) and (b) different number of retrieved negative items (k) on three used datasets. 95
5.3 Entropy comparisons of BPR$_{\text{DNS}}$ and BPR on the MovieLens-1M dataset. . . . 98

6.1 Comparison between different variants of MLP-CGRec, where the green bars are variants with multiple neighbours, orange bars are variants with different MLPs and the red bar is the final MLP-CGRec model. 108

7.1 An overview of the recommendation framework. 111
7.2 An overview of 4 possible combinations between different methods. 112
7.3 An illustration of 4 possible integrated models. 113

Chapter 1

Introduction

1.1 Motivation

With the development of an extreme number of web applications, users are increasingly interested in effective personalised recommendations (Lu et al., 2015; Zhang et al., 2019b). For example, an increasing number of users enjoy online shopping through E-commerce platforms, including the well-known Amazon (Leino and Räihä, 2007) and Alibaba (Wang et al., 2018b). Given that the numbers of items available on those platforms is usually huge (Eksombatchai et al., 2018), recommender systems are essential to help users filter out non-relevant items and generate a ranking of items of interest. Similarly, users may also want to expand their social networks (Tang et al., 2013), where recommender systems can help users explore potential social relations between them and other unconnected users. Traditional collaborative filtering-based (Koren et al., 2022), content-based (Pazzani and Billsus, 2007) and hybrid recommender systems (Thorat et al., 2015) aim to incorporate shallow embeddings and/or detailed descriptions of users and items to generate recommendations when provided with historical interactions. However, limited by the availability of explicit feedback from users, these traditional methods cannot accurately predict explicit ratings (Rendle et al., 2009). In addition, traditional recommender systems cannot represent the heterogeneous relations between different types of entities including users, items and sometimes the extra side information. Specifically, side information is generally defined as the descriptions of items and attributes of users. For example, the descriptions of items include the textual reviews, categories, locations, while the users' attributes include the social relations, ages and occupations. Indeed, existing works (Hui et al., 2021; Zhao and Guo, 2017) have demonstrated that side information can be leveraged to effectively enhance the recommendation performance. Therefore, deep neural recommender systems (He et al., 2017) have been attracting the community's attention because of their strong representational ability to capture not only the interactions information but also the heterogeneous relations among the side information.

Recently, graph representation learning has become a popular trend in different research domains (Jha et al., 2022; Sanchez-Gonzalez et al., 2018; Yi et al., 2022). Broadly speaking, graph representation learning includes techniques that can leverage graph-structured data and generate latent representations for the nodes, graphs/sub-graphs and edges between nodes (Hamilton, 2020). In particular, well-known graph-based techniques include but are not limited to graph neural networks (Kipf and Welling, 2017), graph contrastive learning (You et al., 2020), graph pre-training (Hu et al., 2019), graph-based self-supervised learning (Wu et al., 2021) and graph clustering (Pan and Kang, 2021). Among different techniques of graph representation learning, graph neural networks (GNNs) (Zhou et al., 2020) are of particular interest to the recommendation task. First, GNNs are models that are effective for graph-structure data, where the graph contains nodes and edges for connecting different nodes. Given such input typed as graphs, GNNs aim to learn representations for all nodes, where success is to learn low-dimensional vectors that can be used to accurately predict the links between nodes. Recall that the task of recommender systems is also to explore the complex relations between users and items. We postulate that the target of GNNs naturally aligns with the target of recommender systems. In the case of recommendation, the most basic graph will be a bipartite graph containing the users and items' nodes only with edges denoting their historical interactions (van den Berg et al., 2017). Second, when incorporating different types of side information, existing recommender systems have explored different types of regularisation methods (Ma et al., 2011). However, those regularisation-based methods usually rely on extensive parameter-tuning methods such as grid search to find the best parameters and performance. Different from these regularisation-based models, graph-based recommender systems can leverage such side information by extending the bipartite graph to a heterogeneous graph (Zhang et al., 2019a), where different types of side information could be represented as different types of relations between user-user, item-item and user-item pairs. For example, social relations have been recognised as an important source of recommender systems to infer users' preferences (Ma et al., 2011; Yang et al., 2014). Traditionally, social relations are leveraged by collaborative filtering-based approaches through the regularisation, which gives a penalty to the model when it generates different predictions to socially-related users. In the scenario of graph-based recommender systems, social relations are leveraged by adding social edges between socially related users. Similar to social relations, other side information can be used to categorise users and items. Therefore, the use of graph-based recommender systems is a promising research direction for enhancing recommender systems by better representing the user-item interaction information and exploring more complex relations within side information.

Although effective, there are still a number of remaining challenges for existing graph-based recommender systems (Gao et al., 2021; Wang et al., 2021b). Among these, the *cold-start* issue has become a long-standing issue for recommendation tasks (Lam et al., 2008). Specifically, the *cold-start* issue can be divided into the *cold-start* items issue and the *cold-start* users issue. In general, *cold-start* users are those users who have fewer interactions (the threshold is usually defined as 10 interactions). Similarly, the *cold-start* items are those items that have fewer interactions. In comparison, *regular* users/items are defined as users/items with more interactions. The difference between *regular* and *cold-start* users lies in the number of historical interactions they have made. Existing works have shown that recommender systems, especially those collaborative filtering (CF) based methods usually fail to generate an effective recommendation to *cold-start* users (Park and Chu, 2009). This is because when there are enough historical interactions, CF-based methods are able to effectively recognise the interests of users and hence better recommend items-of-interest to these users. However, for *cold-start* users, CF-based methods fail to find their similar users and items and thus cannot make an effective recommendation. This problem has been stopping CF-based methods from delivering satisfactory recommendations to users since the development of recommender systems. Even though graph-based recommender systems can better aggregate information from the neighbourhoods of users and items (He et al., 2020; Wang et al., 2019c), these *cold-start* users still suffer from less effective recommendations because they have fewer neighbours under the GNN scenario. This *cold-start* problem is becoming more severe with the extreme development of e-commerce and other online applications, because when more and more users register and more and more items become available on those applications, there will naturally be more *cold-start* users and items. Consequently, such cold-start users might decide to leave these online platforms given that they cannot receive satisfactory recommendations in the long run. Furthermore, suppliers may reconsider publishing new items if the recommendation algorithm will not recommend *cold-start* items. A line of research (Gantner et al., 2010; Liu et al., 2020; Puthiya Parambath and Chawla, 2020) investigated leveraging side information to enhance the cold-start recommendation. However, it is still unclear how to incorporate side information of both users and items within the graph-based recommender systems scenario.

Another challenge of graph-based recommender systems is their low robustness. The robustness of a deep neural network (DNN) is usually defined as the variance of their performance when different initialisation of models are applied (Burke et al., 2015). Ideally, a model should be robust enough to achieve similar performance when different initialisation are used. Specifically, an ideal recommender system is expected to perform robustly so that when a wide range of random seeds is used, the devised recommender systems can generate recommendations with a stable level of satisfaction instead of observing a large variance. DNN-based recommender systems have been reported to be less robust than those traditional models. Dacrema et al. (2019) have shown that the performances of many DNN-based recommender systems are not as good as reported. One of the main reasons for this low reproducibility is the low robustness of DNN-based models. Specifically, DNN-based models usually comprise more than one DNN module, where some random operations are included to increase their generalisability. Although effective, many stochastic functions are hard to reproduce. In particular, some stochastic functions, including the random edge/node dropout and graph perpetuations (Wu et al., 2021), might lead to the graph losing some essential edges or nodes. As a result, random seeds play an unexpectedly important role in the final recommendation performance. This effect will, in turn, increase the difficulty of deploying advanced recommendation algorithms in an industry setup. Indeed, the performance of many models is not robust enough, they cannot be used to avoid an unstable satisfaction of users. Many natural language processing models have proposed using the pre-training framework to alleviate the low robustness issue (Devlin et al., 2018; Ma et al., 2021; Reimers and Gurevych, 2019). However, it remains unclear how to leverage a pre-training framework to improve the robustness of recommender systems. Specifically, many existing corpora exist for training NLP models, such as Wikipedia. These existing corpora can be used to train unsupervised language models to capture the abundant context of languages. However, for recommender systems, there is no such existing corpus. Therefore, this challenge remains unresolved.

Furthermore, existing graph-based recommender systems are usually computationally expensive and inefficient. To understand this issue, we first need to have a general idea of the overall architecture of graph-based recommender systems. First, following existing CF methods, graph-based recommender systems initialise the embeddings of users and items with random vectors. Second, to aggregate information from the graph neighbourhood, one or more graph neural network(s) is needed to operate the message passing and neighbourhood aggregation functions. To do so, we need not only the GNNs but also the adjacency matrix of the user-item interaction matrix. This adjacency matrix will be multiplied by the users and items embedding matrix to effectively obtain each node's neighbours. In addition, to obtain multi-hop neighbours, multiple layers of GNNs are usually needed. Although this aggregation process has been demonstrated to be effective for the final recommendations, this multiplication between matrices is indeed inefficient because the size of the adjacency matrix is relatively large. Specifically, given a dataset with 1,000 users and 1,000 items, classic recommender systems might need an embedding matrix $\mathbf{E}^{2000\times d}$. However, graph-based recommender systems need this matrix $\mathbf{E}^{2000\times d}$ and also an adjacency matrix $\mathbf{E}^{2000\times 2000}$. As a result, multiplying enormous matrices may lead to the out-of-memory issue and a lower training efficiency. Given that graph-based recommender systems consume large amount of GPU memory and training time, it can be problematic to adapt them as a practical recommendation algorithm. Therefore, existing works have explored how to simplify the overall GNNs and enhance the efficiency. For example, SGC (Wu et al., 2019a) has been proposed to simplify the GNNs by removing redundant neural operations. However, most of the simplified GNNs (Fu and He, 2021; Wu et al., 2019a; Zhao et al., 2020b) still cannot avoid the graph inference process, which is presumably the most time-consuming operation. Hence, to further improve the efficiency of graph-based recommender systems, one of the challenges is to avoid or largely reduce the graph inference process.

Enhancing the representational power of graph-based recommender systems is one of the solutions to improve the graph-based recommender systems. Besides, improving the training framework has not been explored. Most of the existing graph-based recommender systems (He et al., 2020; Mao et al., 2021; Wang et al., 2019c) adapted the basic Bayesian Personalised Ranking (BPR) (Rendle et al., 2009) as their underlying backbone. Specifically, graph-based recommender systems follow the basic setup of BPR and use the common training triplet consisting of a user, a positive item and a sampled negative item. Each user has a limited amount of positive items but a large amount of negative items. To form these triplets, negative items are randomly sampled from each user's pool of negative items. Though this random negative sampling method is efficient, it may lead to the exposure bias (Khenissi et al., 2020) and the false negative issue (Hariadi and Nurjanah, 2017). With randomly sampled negative items, it is possible that this user has never been exposed to these negative items. Therefore, it is unfair to assume that this user will prefer a positive item to this unexposed item. Second, the negative items of the training set is sampled without knowing the positive items in the test set. This may lead to the false-negative issue when the sampled negative items for the training set is in fact a positive item in the test set. This false-negative issue will largely degrade the final recommendation performance (Liu et al., 2021b). To alleviate this issue, existing methods have attempted to leverage the side information to avoid sampling unexposed or false negative items. For example, using the geographical information can help a recommender system to sample venues from different cities of a specific user to decrease the possibility of sampling false-negative items (Manotumruksa et al., 2017). Another line of research (Yang et al., 2022; Zheng et al., 2020b) investigated the popularity-based sampling method, where items are sampled based on their popularity i.e., the number of times they have been interacted by users. The intuition here is that if an item is popular and it is not interacted by a user, then this popular item should be sampled as a negative item because it is unlikely this user has not been exposed to this item. Even though these aforementioned methods are effective, they are limited by the availability of side information. Indeed, when there is no side information, it will be hard to sample negative items based on some prior knowledge and assumption leading to the simple random sampling used by BPR. As mentioned above, most of the existing graph-based recommender systems still use BPR as their backbone, hence graph-based recommender systems are not superior to other recommender systems from this specific perspective. Therefore, a more advanced negative sampling approach is needed for graph-based recommender systems for a further enhanced performance.

1.2 Thesis Statement

This thesis states that the effectiveness of ranking-based recommender systems can be enhanced by techniques from graph representation learning. Specifically, this thesis argues that by leveraging heterogeneous graph representation learning, graph pre-training and graph contrastive learning methods, we could observe improvements in the overall ranking-based recommendations provided by the graph-based recommender systems while also alleviating their *cold-start* problem, low-robustness and low-efficiency issues. In particular, we hypothesise that we can use the heterogeneous graph representation learning method to alleviate the *cold-start* problem while ensuring enhanced recommendations for *regular* users by leveraging social relations, which are naturally available relations without any pre-processing. We also postulate that the use of graph pre-training can generalise the sole use of social relations to the usage of multiple types of side information, which allows to alleviate the low-robustness issue of graph-based recommender systems and further improves the recommendation performance for both *regular* and *cold-start* users. Finally, we postulate that advanced sampling approaches based on graph contrastive learning can enhance the efficiency of graph-based recommender systems while retaining a competitive recommendation accuracy.

1.3 Contributions

The summary of our contributions is described below:

- **Graph Social Recommendation** To alleviate the *cold-start* issue mentioned in Section 1.2, we start with building a heterogeneous graph to represent the user-item interaction information and social relations between users. In fact, given a user-item interaction matrix, social relations are one of the most naturally available relations that can be used to enrich such an interaction matrix. Therefore, we choose the social relations among all different types of side information, where most of them require unique pre-processing or feature selection. In particular, we hypothesise that users tend to interact with items purchased by their socially-related friends. Hence, social relations are essential side information that can be used to infer the interests of users. Given that recommender systems usually fail to generate accurate recommendations to *cold-start* users, we propose to alleviate this issue by considering the social relations between users. Specifically, we propose to pre-train the recommendation model not only with the interaction data but also with the social relations among users, thereby providing the recommender system with a better initialisation compared with solely relying on the user interaction data. We propose a novel recommendation model, the Social-aware Gaussian Pre-trained model (SGP), which encodes the user social relations and interaction data at the pre-training stage in a Graph Neural Network (GNN). Afterwards, in the subsequent fine-tuning stage, our SGP model adopts a Gaussian Mixture Model (GMM) to factorise these pre-trained embeddings for further training, thereby benefiting the *cold-start* users from these pre-built social relations. By evaluating our SGP model on three real-world datasets, we aim to answer the research question that: Can we use the GNN model to leverage the social relations and generate pre-trained embeddings for both users and items, thereby improving the overall and *cold-start* recommendation performance?

- **Graph Pre-training Recommendation** Despite that SGP also uses pre-training to encode the additional social relations between users, it cannot take multiple types of side information into consideration. Therefore, we aim to generalise such a pre-training technique for multiple types of side information associated with both users and items. Specifically, we propose two novel pre-training schemes, namely **Single-P** and **Multi-P** to leverage the side information including the ages and occupations of users and the textual reviews and categories of items, by pre-training embeddings of users and items using the multi-relational (**Multi-P**) or single-relational (**Single-P**) graph neural network. Compared with SGP, **Single-P** is a more generalised method that is capable of incorporating multiple types of side information, where **Single-P** assigns the same level of importance to the same type of side information. On the contrary, **Multi-P** can learn variable levels of importance to the same type of side information shared by different nodes i.e., users or items, based on the importance of this side information for each specific node. Instead of jointly training with two objectives, our pre-training schemes first pre-train a representation model under the users and items' multi/single relational graphs constructed by their side information, and then fine-tunes their embeddings under an existing general representation-based recommendation model. Our proposed multi-relational and single-relational graph neural networks can generate effective embeddings, while capturing the heterogeneity from the side information simultaneously, thereby improving the performance of the underlying recommendation model. Besides, our pre-training schemes can provide more effective initialisations for both the users and items such that the robustness of fine-tuning models can be improved. We examine our pre-training schemes on three real-world datasets to answer the research question that: Do our pre-trained models help existing recommendation systems obtain better performances?

- **Graph Contrastive Learning for Recommendation** As discussed before, graph-based recommenders suffer from ineffective negative items and low training efficiency. To alleviate these two issues, we first propose a novel scheme called Dynamic Negative Sampling (DNS), which can dynamically explore contrastive negative items and generally improve the recommendation performance of ranking-based RSs. Specifically, DNS uses a similarity-based approach to sample items that are unlikely to be interacted with by a user as his/her contrastive negative items. Therefore, recommender systems using DNS benefit from gradually learning contrastive negative items along the training process. Furthermore, we design a novel approach to measure the entropy among the distributions of users and items. Using such an approach, we show how our DNS scheme for sampling negative items can induce a higher information gain.

  Second, to improve the efficiency of graph-based recommenders, we further propose a novel model i.e., MLP-CGRec using a multiple sampling approach based on graph contrastive learning to sample both negative and positive samples. In relation to DNS proposed above, MLP-CGRec also uses graph contrastive learning to determine contrastive negatives. However, we focus on improving the efficiency of graph-based recommender systems by efficiently sampling the graph neighbours of both users and items to avoid the message passing function of GNN, which is computationally expensive. In addition, we propose a simple algorithm based on Multilayer Perceptron (MLP) for learning users and items' representations with extra non-linearity while lowering computational burden compared with multi-layers GNNs.

By comparing our DNS scheme and MLP-CGRec model with existing sampling approaches, we aim to answer the research question that: Can we use the graph contrastive learning method to enhance the training efficiency and obtain more informative negative items for graph-based recommender systems?

- **Integrated Graph-based Recommender Systems** We have proposed different methods to enhance the performance of graph-based recommender systems and alleviate the *cold-start* problem, low-robustness and low-efficiency issues, where each method has its own advantages. Therefore, it is interesting to investigate whether we can integrate different proposed methods into one graph-based recommender system and retain the advantage of each method. In particular, we integrate two pre-training schemes i.e., **Single-P** and **Multi-P** with two sampling approaches i.e., DNS and the multiple sampling approach used by MLP-CGRec. We do not consider the proposed SGP model as one of the options to avoid repetition because it is a generalised version of **Single-P**. Additionally, we unify the experimental setup used to evaluate our integrated models and all other baselines to avoid the sampling bias, where we use the full negative items for evaluation instead of using the sampled metrics. Finally, we evaluate four integrated graph-based recommender systems and we aim to answer the research question that: Can our integrated graph-based recommender systems achieve enhanced recommendation performance and alleviate multiple issues simultaneously?

1.4 Origins of Material

Most of the material presented in this thesis is based on work considered by various international conferences and journals:

- Chapter 3: We propose a heterogeneous graph pre-training technique to encode social relations between users. Our proposed Social Graph Pre-training (SGP) model uses this social pre-training technique and a Gaussian Mixture Model (GMM) to enhance the recommendation performance and benefit *cold-start* users. This work is under revision for the IPM journal.

- Chapter 4: We propose two pre-training schemes, namely the multi-relational and single-relational graph pre-training schemes that can incorporate multiple types of side information of both users and items. Both schemes can be pre-trained and easily fine-tuned to enhance both the recommendation performance and robustness of a general recommender system. This work is accepted by the TOIS journal.

- Chapter 5: We propose a graph-based dynamic negative sampling approach that provides more informative negative items for pairwise recommender systems. In particular, with these provided negative items, a pairwise recommender will gain higher entropy and enhanced recommendation performance. This work is considered by the TWeb journal.

- Chapter 6: We propose a graph-based recommender (MLP-CGRec) using an efficient multiple sampling method to sample both negative items and positive samples (graph neighbours). In particular, we use a Multilayer Perceptron (MLP) to enhance the recommendation performance without increasing the computational cost. This work was published in SIGIR 2022 (Liu et al., 2022).

1.5 Thesis Outline

The remainder of this thesis is organised as follows:

- Chapter 2 describes the background of the recommendation task and graph representation learning. First, we describe four popular taxonomies to categorise recommender systems. Afterwards, we show the basic models of recommender systems, where we specifically adopt the taxonomy of *Collaborative filtering-based* vs *Content-based* vs *Hybrid* Recommendations to define the scope of this thesis. Then we focus on the offline evaluation and its relevant metrics in the evaluation of recommender systems. Later in the chapter, we introduce the graph representation learning and build the bridge between this learning paradigm with the recommender systems.

- Chapter 3 introduces our proposed technique that incorporates the social relations between users in the graph-based recommendation scenario. Our proposed Social-aware Gaussian Pre-trained model (SGP) can encode the users' social relations by using the graph pre-training technique and a Gaussian Mixture Model. Afterwards, SGP adopts the pre-trained embeddings for the fine-tuning to benefit *cold-start* users. We evaluate our proposed SGP model on three public datasets. Our extensive experimental results compare SGP with 13 competitive baselines. In addition, we examine the recommendation performance of SGP regarding both *cold-start* and *regular* users, respectively, to validate the hypothesis of our proposed thesis.

- Chapter 4 describes a more generalised approach than the proposed SGP model demonstrated in Chapter 3. In this chapter, we propose two schemes i.e., single-relational (**Single-P**) and multi-relational (**Multi-P**) graph pre-training schemes that are extended to multiple types of side information instead of only focusing on the social relations. In particular, we evaluate the effectiveness of two proposed approaches on three public datasets in comparison with state-of-the-art recommender systems.

- Chapter 5 introduces a dynamic negative sampling approach (DNS) to enhance the recommendation performance of ranking-based recommender systems. In particular, inspired by the graph contrastive learning, DNS aims to provide more informative negative items. We conduct extensive experiments on three public datasets to compare the performance of recommender systems with and without using DNS. In particular, we conduct an entropy-based analysis to examine whether DNS can provide more informative negative items.

- In Chapter 6, an efficient graph contrastive recommender i.e., MLP-CGRec is proposed. Our MLP-CGRec is based on an efficient graph neighbour sampling method with a Multilayer Perceptron (MLP). We conduct extensive experiments on three public datasets to evaluate both the recommendation accuracy and efficiency of MLP-CGRec compared with seven baselines. In particular, we measure the training time and memory consumption of MLP-CGRec and seven baselines in order to consolidate our proposed thesis.

- Building on models/methods proposed in Chapters 3, 4, 5 and 6, we introduce four integrated models, namely **Single-P**+DNS, **Multi-P**+DNS, **Single-P**+MLP-CGRec and **Multi-P**+MLP-CGRec in Chapter 7. As mentioned before, **Single-P** is the generalised version of SGP towards multiple types of side information. Hence, it is unnecessary to build extra integrated models based on SGP. This chapter aims to investigate whether our proposed models can be combined to achieve better effectiveness under a unified experimental setup.

- Chapter 8 closes this thesis by highlighting the contributions and the conclusions of each chapter. We also discuss some possible future directions for our research.

To sum up, this thesis focuses on two main research directions, namely recommender systems and graph representation learning. In particular, we aim to enhance ranking-based recommender systems in terms of their recommendation accuracy, robustness and efficiency by incorporating different graph-based techniques. After we bridge the recommender systems and graph representation learning (Chapter 2), we explore how to incorporate the social graph recommendation, where we use a heterogeneous graph to encode both social relations and user-item interactions for the subsequent recommendation (Chapter 3). Building on this social graph recommendation approach (SGP), we further develop two more generalised pre-training schemes, namely **Single-P** and **Multi-P** (Chapter 4). In comparison to SGP, **Single-P** and **Multi-P** are extended to different types of side information instead of only the social relations. Therefore, the development from Chapter 3 to Chapter 4 can be interpreted as the process of generalising the graph representation learning from incorporating a single type of relation (i.e., the social relation) to multiple types of relations for the recommendation task. After investigating how to incorporate more side information for recommendation, we pay attention to enhancing the underlying negative sampling approach of recommender systems. Different with SGP, **Single-P** and **Multi-P**, our proposed dynamic negative sampling approach (DNS) is a more generic sampling approach that does not rely on the side information. Given that all recommender systems proposed in this thesis use the most basic random sampling approach, we can further enhance their effectiveness by developing integrated models. Similar to DNS, MLP-CGRec (Chapter 6) also aims to incorporate more samples during the training of a recommender system. However, instead of focusing on negative samples only, MLP-CGRec can leverage both positive and negative samples. Hence, the underlying sampling method of MLP-CGRec can also be used to enhance the efficiency of the previously proposed models. Finally, in Chapter 7, we investigate integrated model(s) that leverage one model among **Single-P** and **Multi-P** and one sampling method from DNS and MLP-CGRec, such that this integrated model can benefit from each used model/method.

Chapter 2

Background

In Chapter 1, we focused on the functionalities of the recommender systems and their limitations. To alleviate those limitations including the *cold-start*, low expressive power and efficiency issues as well as the lack of ability to incorporate the side information, we have proposed graph-based techniques, including heterogeneous graph representation learning, graph pre-training and graph contrastive learning in Section 1.3. In this chapter, we first provide an overview of the recommendation task in general and then provide the basic notations and information of graph representation learning that this thesis is built on. In general, we describe the existing techniques, especially those deep learning-based models for graph data that our proposed method relies on for modelling the complex structure of user-item interactions in a collaborative filtering manner (Li and She, 2017). In particular, we first describe some classic recommender approaches, including the collaborative filtering-based, content-based and hybrid approaches in Section 2.1.1. Then we move towards describing the commonly-used evaluation methods and metrics in Section 2.1.2. Afterwards, we focus on graph representation learning by first introducing some preliminaries of graph networks such as the graph data structure and graph-based transformations, followed by the relation between the recommender systems and the recently arisen Graph Neural Networks in Section 2.2.1. After that, we introduce the heterogeneous graph representation learning, graph pre-training and graph contrastive learning, which are three main building blocks for this thesis in Sections 2.2.2, 2.2.3 and 2.2.4 respectively.

2.1 Overview of Recommender Systems

Recommender systems are algorithms and techniques that provide some relief from the plethora of choice faced by consumers. Given the rapid growth of information and items available on the Web, users often need to select items aligned with their interests from countless products, movies, or venues. As such, personalisation is an essential strategy for helping users to filter out the large portion of non-relevant items so as to enhance the user experience. Therefore, recommender systems play a crucial and indispensable role in online applications and information

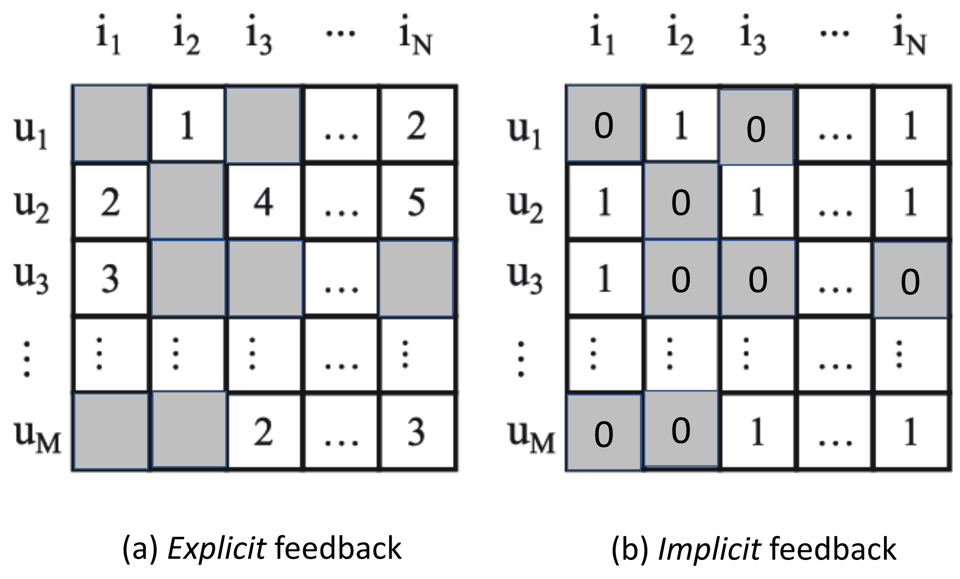
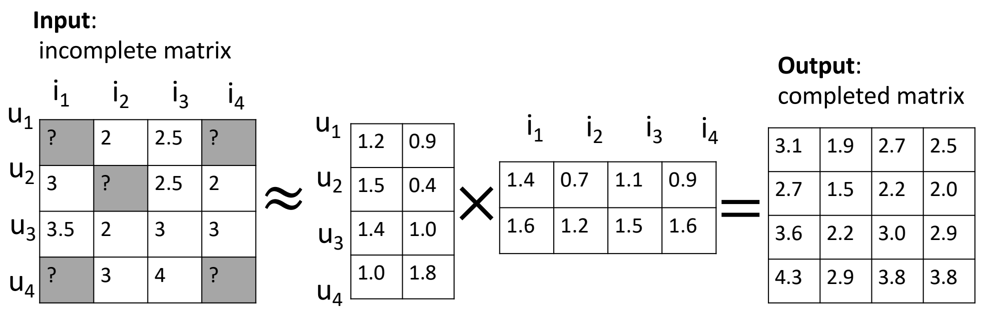


Figure 2.1: *Explicit* and *implicit* feedback of users.

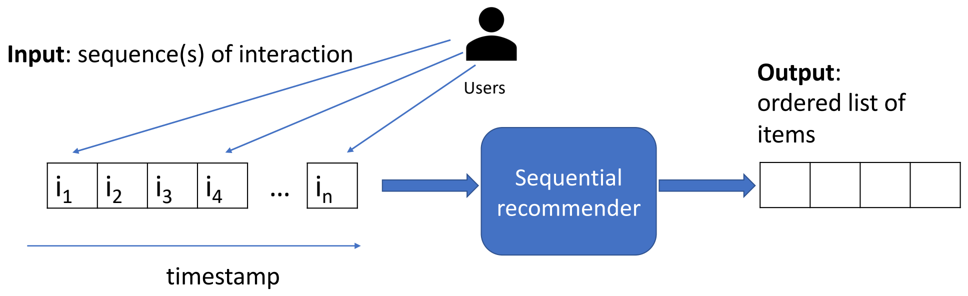
access systems in order to boost the profits of businesses and to facilitate the decision-making process (Jannach et al., 2010; Ricci et al., 2015). In general, the list of recommended items are generated based on the users' historical interactions, the items' features and some other additional information such as temporal (e.g., in sequence-aware recommender systems (Quadrana et al., 2018)) or spatial (e.g., in Point-of-Interest (POI) recommender systems (Bobadilla et al., 2013)) information.

With the emergence of online shopping, users can explicitly provide feedback whether they like or dislike items/services recommended by the used platforms by providing ratings. However, due to privacy issues (Badsha et al., 2017), it has become difficult to fetch the explicit feedback since users are not willing to explicitly rate items. As a result, most of the platforms only have access to the users' implicit feedback i.e., they only know that this user has interacted with an item but do not know if this user appreciates this item or not. The difference between the available data is used to categorise recommender systems into *implicit recommendations* and *explicit recommendations*. Although for the scope of this thesis, we mainly focus on neural recommender systems with implicit feedback, it is essential to introduce all commonly used taxonomies of recommender systems to give a better overview. We describe four of the taxonomies in which the recommendation systems can be categorised.

- *Explicit* vs. *Implicit* Recommender systems: Explicit recommendation is also known as rating prediction, where inputs are the ratings provided by users. The task of explicit recommendation is therefore, given some ratings of users, to predict each user's ratings towards all other items. In comparison, the implicit recommendation is known as the ranking-based recommendation, where inputs are implicit interaction records such as the clicking records. The task of implicit recommendation is, given some interaction records



(a) Matrix completion method



(b) Sequential method

Figure 2.2: An overview of matrix completion and sequential recommender systems.

of users, to generate the ranking of all items for each user, where the top-ranked items should be the items of the users' interests. Traditionally, the user-item interaction is represented as a matrix $\mathbf{R}^{m*n}$ where m and n denote the number of users and items, respectively. For explicit recommendation, r_{ij} of $\mathbf{R}$ indicates the observed rating feedback by user i on item j (e.g., 1-5). In contrast, for implicit recommendation, r_{ij} of $\mathbf{R}$ is a binarised value 0/1 indicating whether user i has interacted with item j or not. Figure 2.1 shows two matrices to illustrate the difference between the implicit and explicit feedback.

- *Sequential* vs. *Non-sequential* Recommender systems: In sequential recommendation, also known as the sequence-aware recommendation, the order of interactions matters (Quadrana et al., 2018). In contrast, the non-sequential recommendation can be interpreted as the matrix completion, where the matrix is defined as the incomplete user-item feedback matrix, and the task is to complete all of the unknown places in this matrix. In sequential recommendation, inputs are ordered sequences for each user and outputs are usually predicted sequences of items that users are expected to click in order. In contrast, non-sequential recommender systems take randomly ordered interactions as inputs and output rankings of items for each user, where a higher ranking indicates a higher preference. Indeed, sequence-aware recommender systems play an indispensable role in specific applications including the music, micro-video, grocery recommendation and community/individual trends detection (Hansen et al., 2020). Figure 2.2 illustrates the inputs, outputs and the overall framework of matrix completion and sequential recommender systems, respectively. Since we are not investigating sequence-aware recommendation in this thesis, we refer to this survey (Quadrana et al., 2018) for a more advanced overview.

- *Collaborative filtering-based* vs. *Content-based* vs. *Hybrid* Recommender systems: Collaborative filtering-based approaches are based on the assumption that users who like similar items in the past will like similar items in the future. A key advantage of collaborative filtering-based approaches is that they do not rely on the description of items and users (Burke et al., 2015; Thorat et al., 2015). Therefore, they are capable of accurately recommending complex items such as movies without requiring an understanding of the items themselves. However, collaborative filtering-based approaches often suffer from the *cold-start* problem when users or items do not have enough historical interactions (Liu et al., 2020; Park and Chu, 2009). Differently, content-based recommender systems are based on descriptions of items and profiles of users. Specifically, content-based approaches are suited to the situation when there is a known data on each item such as its name, its categories and location. In comparison to collaborative filtering-based approaches, content-based approaches do not rely on the historical interactions but they cannot handle the cases when the users' profiles or the items' descriptions are not available. Nowadays, most of the recommender systems use a hybrid approach by combining collaborative filtering-based and content-based approaches. Usually, by unifying the content-based capabilities with a collaborative filtering-based approach, a recommender system can provide more accurate results and alleviate the *cold-start* and data sparsity issues at the same time (Zhang et al., 2019c).

- *Neural network-based* vs *Classic* Recommender systems: Classic recommender systems are those methods that do not involve any modern neural networks. Well-known classic recommender systems include matrix factorisation (MF) (Koren et al., 2009) and Bayesian Personalised Ranking (BPR) (Rendle et al., 2009). To model the interactions between users and items, many linear and matrix completion approaches are used in classic recommender systems. If the side information such as the reviews left by users and categories of items, is involved, one or more regularisation(s) could be used to enhance the performance. For example, a regularisation term can be used with an objective function to increase the similarities between users and/or items sharing the same type of side information. In comparison, neural network-based recommender systems specifically denote those approaches incorporating one or more deep learning-based methods including the Convolutional Neural Networks (CNN) (Yuan et al., 2019), Recurrent Neural Networks (RNN) (Manotumruksa et al., 2018), Multilayer Perceptron (MLP) (He et al., 2017), Autoencoder (AE) (Liang et al., 2018), Deep Reinforcement Learning (DRL) (Xin et al., 2020) and Graph Neural Networks (GNNs) (Kipf and Welling, 2017).

Given that most users are less willing to share their explicit ratings, we will focus on implicit feedback for all the experiments conducted in the following chapters. However, some classic recommender systems based on explicit feedback such as matrix factorisation (Koren et al., 2009) will also be introduced for the completeness. In addition, we aim to develop techniques that can accurately generate a list of ranked items for users instead of a sequentially ordered list of items. Therefore we will focus on introducing two basic collaborative filtering-based approaches followed by the classic content-based and hybrid methods in the next section.

2.1.1 Basic Models for Recommender Systems

The basic models for recommender systems generally leverage two types of data to generate effective recommendations to users, which are (1) the user-item interactions and (2) the content information about users or items i.e., known as the side information. As mentioned above, the user-item interactions can be categorised into explicit interactions or implicit interactions. For the side information, users can have their ages and occupations as their side information and items can have textual descriptions, geographical locations, prices as their side information. Approaches that only use the interactions are referred to as collaborative filtering-based recommender systems whereas approaches that only uses descriptions are referred to as content-based recommender systems and approaches that use both are referred to as hybrid recommender systems. In the following, these three categories of recommender systems will be described in details.

2.1.1.1 Collaborative Filtering-based Recommender Systems

Collaborative filtering (CF)-based recommendation techniques assist users to make choices based on the opinions of other people who share similar interests. CF-based approaches are especially useful when no description of users and items are available. Well-known and effective example of a CF approach is Matrix Factorisation (MF) (Koren et al., 2009). Traditional CF-based approaches assume that users who share similar interactions in the past will like similar items in the future. In particular, an MF-based approach aims to decompose the user-item interaction matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ into two lower dimensional matrices, namely the latent matrix of users $\mathbf{U} \in \mathbb{R}^{m \times d}$ and the latent matrix of items $\mathbf{V} \in \mathbb{R}^{n \times d}$, where d is the latent dimension of users and items. Here, the underlying intuition is that the latent vector of a user is expected to represent this user's interests in a latent space and the latent vector of an item is expected to represent the item's characteristics. Then we can use the dot product to calculate the predicted scores $\hat{r}_{ui}$ of a user u on a target item i as follows:

$$\hat{r}_{ui} = e_u \odot e_i, \tag{2.1}$$

where e_u and e_i denote the latent vectors of user u and the target item i, respectively; $\odot$ denotes the dot product operation.

The objective of MF is to minimise the pointwise loss between the predicted rating $\hat{r}_{ui}$ and the observed rating $\hat{r}_{ui}$ and hence the loss function of MF is defined as follows:

$$\mathcal{L}(\Theta) = \min_{\Theta} \frac{1}{2} \sum_{u=1}^{m} \sum_{i=1}^{n} I_{ui} \cdot (r_{ui} - \hat{r}_{ui})^2 + \frac{\lambda}{2} \|\Theta\|_F^2, \tag{2.2}$$

where I_{ui} is an indicator function that is 1 if user u has interacted with the item i, otherwise 0. λ is a constant value, which is used to control the strength of the regularisation term i.e., $\|\Theta\|_F^2$. θ denotes all learnable parameters including the embeddings of users and items and the regularisation, where $\|\cdot\|_F^2$ denotes the Frobenius norm of a matrix. To optimise the objective function and find an optimum solution, Stochastic Gradient Descent (SGD) (Ketkar, 2017) is commonly applied until the convergence.

In addition to the formulation of the basic MF (Equation (2.2)), there are many other variants shown to have better or more robust performances. Among these, the most generalised and simplest variant is the one (Rendle et al., 2020) that incorporates no additional parameter but the explicit regularisation of the users and items' embeddings so that the objective function of Equation (2.2) is modified as follows:

$$\mathcal{L}(\Theta) = \min_{\Theta} \frac{1}{2} \sum_{u=1}^{m} \sum_{i=1}^{n} I_{ui} \cdot (r_{ui} - \hat{r}_{ui})^2 + \frac{\lambda_1}{2} \|\Theta\|_F^2 + \frac{\lambda_2}{2} \|\mathbf{U}\|_F^2 + \frac{\lambda_3}{2} \|\mathbf{V}\|_F^2, \tag{2.3}$$

where $\mathbf{U}$ and $\mathbf{V}$ are the users and items' embeddings and λ_1, λ_2 and λ_3 control the global bias, user bias and item bias, respectively. Although this modification does not help the model to obtain more content information, it has been demonstrated to be effective for gradient descent-based approaches (Paterek, 2007; Rendle et al., 2020).

While MF has been treated as the primary building block of many classic and deep learning-based recommender systems, it is more suited to a rating prediction task. In practice, most of the users only focus on the top-ranked items, instead of the whole item list. In addition, users are generally unwilling to share their explicit ratings, making it more challenging to predict accurate ratings given only binarised interactions. To alleviate this problem, another line of research focusing on the ranking-based approaches has been proposed to leverage binarised interactions, which is known as implicit feedback. In particular, Bayesian Personalised Ranking (BPR) (Rendle et al., 2009) is a popular and widely implemented pairwise ranking-based approach that is able to leverage more abundant implicit feedback to effectively generate the top-K recommendations. BPR assumes that a user prefers an interacted item over the uninteracted ones. In particular, for each user $u \in \mathcal{U}$, BPR assumes that the probability that user u will interact with an interacted item i is higher than the probability that this user u will interact a uninteracted item

j, which can be mathematically expressed as follows:

$$\mathcal{L}(\theta)=\prod_{u\in\mathcal{U}}\prod_{i\in\mathcal{V}_u^+}\prod_{j\in\mathcal{V}_u^-}P(\hat{c}_{ui}\succ\hat{c}_{uj}|\theta), \tag{2.4}$$

where $\hat{c}_{ui}$ and $\hat{c}_{uj}$ denote the probabilities that the user u interacts with the item i and item j, respectively. Rendle et al. (2009) proposed a sigmoid function, $\sigma(x)=\frac{1}{1+\mathrm{e}^{-x}}$, which is used to approximate the probability function $P(\cdot)$. Therefore, the objective function of BPR is shown as follows:

$$\mathcal{L}(\Theta)=\underset{\Theta}{\operatorname{argmax}}\sum_{(u,i,j)\in D_s}\ln\left(\sigma\left(\hat{x}_{uij}\right)\right)-\lambda\left\|\theta\right\|_F^2, \tag{2.5}$$

where D_s contains all pairs of users with their interacted items and sampled negative items; $\hat{x}_{uij}$ denotes the difference between two probabilities i.e., $\hat{x}_{uij}=\hat{c}_{ui}-\hat{c}_{uj}$.

Since most of the items are negative items for each user, the amount of negative items, i.e., the number of j is enormous. If we consider all negative items, the training of BPR will be computationally expensive. Hence, a sampling approach is usually adopted to alleviate this issue. Random sampling, i.e., sampling one negative item for each user-positive item pair, is a simple yet effective sampling approach. However, it may induce the exposure bias issue (Chen et al., 2020b; Khenissi et al., 2020) since these randomly sampled negative items might have never been presented to the user. Therefore, more effective negative sampling approaches can be used to enhance BPR-based approaches (Hariadi and Nurjanah, 2017; Tran et al., 2019; Yang et al., 2022).

Due to the simplicity of MF and BPR, many variants of such methods have been proposed, including implicit matrix factorisation (Hu et al., 2008) and non-negative matrix factorisation (Wang and Zhang, 2012). In fact, using the collaborative filtering method to capture the interests of users and the characteristics of items is an essential concept when designing an effective recommender system. Most deep learning-based approaches cannot succeed without the building block of BPR. Therefore, we also use BPR and its deep learning-based variants as important baselines in this thesis.

In the next section, another classic building block of recommender systems i.e., content-based filtering will be described in details.

2.1.1.2 Content-based Recommender Systems

Content-based recommender systems directly analyse a set of descriptions of items previously rated by a user to generate a profile containing the users' interests based on their rated items. The profile of a user is usually a structured representation of this user's interests, which will be leveraged to recommend new items. The process of recommendation is to compute the similarity between the profiles of users and items. If a user's interests are precisely reflected by his/her profile, it is extremely convenient for a content-based recommender system to generate a list of relevant items for this user. The overall recommendation process is performed in three steps, namely *Content Analyser*, *Profile Learner* and *Filtering Component*, each of which is handled by a separate component (Pazzani and Billsus, 2007).

- *Content Analyser*: The descriptions of items are usually texts. Therefore, a content analyser is needed to pre-process the input data and then extract structured information or relations for the next step. The main responsibility of the component is to represent the content of items obtained from a platform in a form suitable for the profile learner and filtering components.

- *Profile Learner*: This module collects pre-processed data from the content analyser, in order to construct the user profile (Muruganandam and Srininvasan, 2017). Usually, this process is facilitated through machine learning or deep learning techniques that are able to infer the users' interests given the historical liked or disliked items. For example, the profile learner of a content-based recommender system can implement a relevance feedback method where positive and negative items are represented as vectors and the user profiles are generated by aggregating such a feedback.

- *Filtering Component*: This module aims to explore the user profile to recommend items of interest by matching the vector representation of a user's profile against the representations of all items. In this process, the matching is realised by computing some similarity metrics between the vector representations of users and items.

In most content-based filtering systems, item descriptions are textual features crowd-sourced from Web pages, emails, news articles or product descriptions. Unlike structured data with explicit relations, there are no attributes with pre-defined values. Due to the ambiguity of the natural language, a content analyser cannot learn an explicit profile representation for the users. To alleviate this problem, some simple yet effective term weighting models can be used such as Term Frequency-Inverse Document Frequency (TF-IDF) (Salton and Buckley, 1988).

To apply TF-IDF weighting in content-based recommendation, first let $D = \{d_1, d_2, ..., d_N\}$ denote a set of corpus and $T = \{t_1, t_2, ..., t_n\}$ be the dictionary containing the set of words in this corpus. To obtain a meaningful dictionary T, some standard natural language processing methods, including tokenisation and stopwords removal should be applied. Each description $d_j \in D$ is represented as a $n-$dimensional vector, so that $d_j = [w_{1j}, w_{2j}, ..., w_{nj}]$, where w_{kj} is the weight for term t_k in d_j. To compute each weight t_k for a given description d_j, we can use the following TF-IDF weighting function:

$$\mathrm{TF-IDF}\,(t_k, d_j) = \underbrace{\mathrm{TF}\,(t_k, d_j)}_{\mathrm{TF}} \cdot \underbrace{\log \frac{N}{n_k}}_{\mathrm{IDF}}, \tag{2.6}$$

where N denotes the number of documents in the whole corpus, and n_k is the number of descriptions in the corpus, where the term t_k occurs at least once. The term frequency i.e., TF in Equation (2.6) is computed as:

$$\mathrm{TF}\left(t_k, d_j\right) = \frac{f_{k,j}}{\max_z f_{z,j}}, \tag{2.7}$$

where the maximum is computed over the frequencies $f_{z,j}$ of all terms t_z that occur in document d_j. There are different methods to compute the term frequency and inverse document frequency scores, which are detailed in the existing literature (Salton and Buckley, 1988).

After obtaining the representation for each item, we can also obtain the vector representation of each user by aggregating all weighted features that appear in the interacted items of this user. Then, the cosine similarity can be used to compute the similarities between each user with all of his/her candidate items given the vector representations $\mathbf{e}_u$ and $\mathbf{e}_i$:

$$\mathrm{Cosine}(\mathbf{e}_u, \mathbf{e}_i) = \frac{\mathbf{e}_u \cdot \mathbf{e}_i}{\|\mathbf{e}_u\| \|\mathbf{e}_i\|} = \frac{\sum_{j=1}^{n} \mathbf{e}_{i,k} \mathbf{e}_{u,k}}{\sqrt{\sum_{k=1}^{n} \mathbf{e}_{i,k}^2} \sqrt{\sum_{k=1}^{n} \mathbf{e}_{u,k}^2}}, \tag{2.8}$$

Although effective, content-based recommender systems have a natural limit on the amount of associated features, whether automatically or manually extracted from the corpus. Also, domain knowledge is often needed to determine which features are essential (Wang et al., 2019b). Therefore, they are commonly combined with the aforementioned collaborative filtering-based approaches to make more effective recommendations. In particular, we use the cosine similarities (Equation 2.8) between the social relational vectors of users to build the social graph in Chapter 3. Besides, we use the TF-IDF weighting function (Equation 2.6) to select important features from the reviews left by users in Chapter 4.

2.1.1.3 Hybrid Recommender Systems

Hybrid recommender systems combine two or more of the techniques to improve the overall recommendation performance, usually to deal with the *cold-start* problem. For example, collaborative filtering-based approaches cannot handle new items and content-based approaches cannot handle new users. Different types of hybrid recommendations include weighted, switching, mixed and feature-augmented recommender systems (Aggarwal et al., 2016). In this thesis, we mainly focus on the feature-augmented recommender systems, where we aim to integrate side information to enhance the recommendation performance.

2.1.2 Evaluation of Recommender Systems

The evaluation of a recommender system is essential to assess the quality of a designed recommendation approach. Therefore, evaluating and comparing different recommendation methods have been an active research topic (Chen and Liu, 2017; Del Olmo and Gaudioso, 2008). For an industry developer, it is a challenging task to select the most appropriate recommender system among a large corpus of implemented methods. Furthermore, researchers who propose novel recommender systems are also expected to compare the performance of their proposed method against existing competitive recommender systems. In this section, we discuss how to compare different recommender systems by using an appropriate evaluation method with one or more evaluation metrics. In particular, we focus on the evaluation of offline algorithms, while user studies and online evaluation e.g., A/B tests are not covered since they are not incorporated in this thesis and cannot be conducted without a developed recommender system and a population of real users.

2.1.2.1 Offline Evaluation Methodology

Offline evaluation aims to examine the performance of recommender systems in a setting when historical user-item interactions are collected by real-world commercial platforms such as Amazon and Yelp (Beel et al., 2013). Such historical interactions may also be associated with side information such as the textual comments left by the users and the temporal information determining when this interaction happened. Some side information may be considered when evaluating a recommender system. For example, the timestamps of interactions can be used to split the whole dataset chronologically so that the latest interactions are added into the test set. Then, the evaluated recommender systems aim to recover the latest interactions given the previous interactions. Using the interaction data and side information, the aim is to simulate behaviours of users that interact with a recommender system by assuming that the behaviour of the user when data is collected will be similar enough to the situation when the recommender system is used. The offline evaluation is attractive to both research and industry communities because they do not rely on a user study and thus allow researchers and developers to compare different candidates at a relatively low cost than the online evaluation and user study.

There are different criteria for the offline evaluation of recommender systems. Below, we make a list of commonly used criteria:

- **Random Split**: Given the user-item interaction matrix **R**, the random split method will randomly divide the whole matrix into three portions, namely the training set $\mathcal{D}_{training}$, the test set $\mathcal{D}_{test}$ and the validation set $\mathcal{D}_{validation}$. Usually, $\mathcal{D}_{training}$, $\mathcal{D}_{test}$ and $\mathcal{D}_{validation}$ contain 70%, 20% and 10% of randomly sampled data from **R**, respectively. To make sure that each user can appear at least once in each set, a filtering operation that removes users with fewer than 3 interactions is usually used.

- **Leave-one-out Split**: In this setup, the user-item interaction matrix **R** will also be divided into the training, test and validation sets. However, different from the random split where the amount of data contained in each set is pre-defined by a certain percentage, leave-one-out split only preserves one interaction in the test set and one in the validation set. By doing so, more interactions from **R** can be preserved in $\mathcal{D}_{training}$ so that the evaluated recommender systems are offered more data to learn the users' behaviours.

- **Chronological Split**: A chronological split can only be applied when each interaction comes with a timestamp. It is especially useful when evaluating sequential recommender systems. Moreover, both the random and leave-one-out splits can be adapted to a chronological split. Taking the leave-one-out split method as an example, we can leave the most recent and the second most recent interactions of each user in the test and validation sets, respectively. Hence, we can better simulate the case when given the historical interactions to predict the next item-of-interest, compared with the case when items in the test set are randomly sampled.

In principle, we use the offline evaluation to select the best performing recommendation algorithms from a large set of candidate algorithms. In addition, the offline evaluation can also be beneficial when multiple hyper-parameters need to be tuned to search for the best parameters. In this thesis, we evaluate the effectiveness of our proposed approaches and all baselines using the offline evaluation due to its simplicity. In this thesis, we mainly use the leave-one-out split (see Sections 3.4.2 and 4.5.3) and random split (see Sections 5.4.2 and 6.4) to evaluate our proposed models. Given that we do not consider sequential recommendation, a chronological split is not applicable without the timestamps of interactions. In the next section, we describe various evaluation metrics that are widely used to evaluate the effectiveness of recommendation systems.

2.1.2.2 Evaluation Metrics

To evaluate the effectiveness of a recommender system, different evaluation metrics have been proposed. They can be generally categorised into rating-based metrics and ranking-based metrics. Specifically, rating-based metrics (e.g., Root Mean Square Error or Mean Absolute Error) focus on measuring how accurately an evaluated recommender system is able to predict the rating values of each user towards items, while ranking-based metrics (e.g., Precision, Recall and Normalised Discounted Cumulative Gain) focus on measuring how accurately an evaluated recommender system is able to rank a list of items. In addition to metrics focusing on the rating accuracy, there are other aspects such as novelty (Castells et al., 2022), serendipity (Ge et al., 2010), fairness (Beutel et al., 2019) and diversity (Kunaver and Požrl, 2017), which can be used to evaluated a recommender system. However, within the scope of this thesis, we mainly focus on improving the recommendation accuracy over existing models, hence we do not consider metrics measuring the novelty in the evaluation. Below, we discuss all used metrics throughout this thesis in details.

First, let r_{ui} be the value of the rating given by the user u on an item i, which is a data instance in the test set $\mathcal{D}_{test}$. Then we define $\hat{r}_{ui}$ as the predicted rating given by a specific trained recommender system. To evaluate a rating-based recommender system, two metrics i.e., Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are commonly used, where MAE and RMSE are defined as follows:

$$\text{MAE} = \frac{\sum_{(ui)\in\mathcal{D}_{test}} |\hat{r}_{ui} - r_{ui}|}{|\mathcal{D}_{test}|} \tag{2.9}$$

$$\text{RMSE} = \sqrt{\sum_{(ui)\in\mathcal{D}_{test}} \frac{(\hat{r}_{ui} - r_{ui})^2}{|\mathcal{D}_{test}|}}. \tag{2.10}$$

Both the MAE and RMSE metrics measure the difference between the actual ratings and the predicted ratings. However, unlike MAE, which simply adds up the absolute difference, RMSE takes the sum of the square of the difference before the square root, which means that RMSE gives a relatively high weight to large errors. Therefore, in a particular case when robustness is critical, RMSE is more favourable since it has a lower tolerance to large errors. However, RMSE tends to generate more misleading results than MAE since it cannot reflect average errors (Willmott and Matsuura, 2005).

In contrast to metrics focusing on the rating prediction, the ranking-based metrics pay more attention to the accuracy of the ranks of items. To evaluate ranking-based recommender systems, metrics such as Precision, Recall, Mean Average Precision (MAP) and Normalised Discount Cumulative Gain (NDCG) are commonly used.

Precision calculates the fraction of the recommended items that are relevant, while recall is the fraction of relevant items recommended over the total amount of relevant items. The insight of precision is that it judges how relevant the recommended items are, while the insight of recall is that it determines how many relevant items are successfully exploited among the whole corpus of relevant items. Similarly, precision and recall are binary metrics, and the higher they are, the better. Usually a cut-off value K is typically pre-defined before calculating precision and recall for each user, denoted as Precision@K and Recall@K:

$$\text{Precision}\ @K = \frac{\sum_{i\in R_u} rel_u(i)}{|R_u|} \tag{2.11}$$

$$\text{Recall}\ @K = \frac{\sum_{i\in R_u} rel_u(i)}{\sum_{i\in\mathcal{V}_u} rel_u(i)}, \tag{2.12}$$

where R_u and $\mathcal{V}_u$ denote the set of top-K recommendations of user u and all interacted items of user u, respectively and $rel_u(i)$ returns 1 if item i is relevant to user u, otherwise 0.

Both precision and recall defined above only count the number of relevant items but do not take the position of relevant items among top-K recommended items into consideration. On the contrary, MAP measures the mean of the Average Precision (AP) values over all users (Costa et al., 2007). Differing from precision and recall, relevant items ranked near the top of the ranking contribute more than relevant items ranked near the bottom. MAP is calculated as follows:

$$\mathrm{MAP} = \sum_{u \in \mathcal{U}} \frac{\sum_{k=1}^{K} \mathrm{Pre}(R(u), k) \cdot \mathrm{Rec}(R(u), k)}{|\mathcal{U}|}, \tag{2.13}$$

where $\mathcal{U}$ is the set of users, k is a rank of the recommended item $R(u)$ for user u, $Pre(R(u), k)$ is the precision at cut-off K and $Rec(R(u), k)$ is the change in recall between rank 1 and k.

Although MAP can consider the ranking position of the relevant items, it is built upon a binary relevance grade, hence MAP is not suitable to evaluate a recommender system when there are multiple levels of relevance in the test set $\mathcal{D}_{test}$. It is common to use Normalised Discount Cumulative Gain (NDCG) (Järvelin and Kekäläinen, 2002) when multiple relevance grades are available:

$$\mathrm{NDCG} = \frac{\mathrm{DCG}}{\mathrm{IDCG}}, \tag{2.14}$$

where DCG and IDCG denote the Discount Cumulative Gain (DCG) and the ideal DCG, respectively, where DCG is calculated as follows:

$$\mathrm{DCG} = \sum_{i=1}^{K} \frac{2^{rel(i)}}{\log_2 i + 1}, \tag{2.15}$$

where $rel(i)$ represents the ground-truth relevance score of the recommended item at the position i, with $rel(i)$ = 1 if the recommended item is relevant, otherwise $rel(i)$ = 0.

Specifically, MAP NDCG and Recall are used in Sections 3.4.2, 4.5.3, 5.4.2, 6.4 and 7.3. In this section, we have introduced a number of evaluation methods and metrics used in the literature to evaluate recommender systems in an offline setup. In the next section, we describe the graph representation learning (GRL) and components based on GRL such as heterogeneous graph representation learning, graph pre-training and graph contrastive learning, which are essential building blocks of this thesis.

2.2 Graph Representation Learning

Traditional deep learning-based models can only handle data in the Euclidean space. For example, Convolution Neural Networks (CNNs) (Gu et al., 2018) can effectively learn features from an image input, and Recurrent Neural Networks (RNNs) (Medsker and Jain, 1999) can incorporate sequence data. However, modelling data from a graph, where nodes are connected by edges, is still challenging since CNNs and RNNs assume that data points positioned closely are related to each other, which is not the case for graph data. Therefore, a new learning paradigm is needed to handle data in the non-Euclidean space. In fact, representing data in the graph format is significant for different disciplines such as modelling physical systems (Sanchez-Gonzalez et al., 2018), learning molecular representations (Atz et al., 2021), predicting protein-protein interactions (Jha et al., 2022), classifying gene-disease relations (Yi et al., 2022) on predicting social networks (Fan et al., 2019). To represent graph data, various graph neural networks (GNNs) and different types of graph-based techniques have been proposed recently, which can all be described as graph representation learning methods. In the following, we will explain the motivations (see Section 1.1) of using graph representation learning to enhance recommender systems.

First, the user-item interaction matrix **R** can be viewed as a non-Euclidean bipartite graph, which cannot be handled by those neural networks designed for Euclidean data e.g., images. The non-Euclidean nature of matrix **R** leads to a direct adaption of this matrix to a general GNN without complicated pre-processing, where users and items in **R** can be regarded as two types of nodes in a graph (van den Berg et al., 2017). Second, GNNs are well-known for their effective message passing and neighbourhood aggregation functions, which also align with the assumption of collaborative filtering-based approaches. Recall that collaborative filtering-based approaches assume users with similar purchase histories will prefer similar items. Since graph representation learning (GRL), especially GNNs, can effectively aggregate information from each node's neighbour to itself (Gao et al., 2021), we are motivated to use GRL to enhance the ability of collaborative filtering-based recommender systems to leverage the neighbourhood information. Third, a typical user-item bipartite graph can be easily extended to a heterogeneous graph (Liu et al., 2020), which suits the need of recommender systems to encode different types of relations among or between users and items. Finally, GRL has excellent extensibility (Wu et al., 2020c); hence, it can leverage many recently proposed techniques. For example, pre-training (Qiu et al., 2020) and contrastive learning (You et al., 2020) methods can both be incorporated by GRL. As a result, these advanced GRL methods can be further integrated to enhance the recommender systems' robustness, effectiveness and efficiency.

Next, we will describe the necessary preliminaries (Section 2.2.1) followed by heterogeneous graph representation learning (Section 2.2.2), graph pre-training (Section 2.2.3) and graph contrastive learning (Section 2.2.4), which are the primary building components of this thesis.

2.2.1 Preliminaries

Graphs are a kind of data structure that can model a set of objects (nodes) and their relationships (edges). Formally, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ between these nodes. An edge $(u, v) \in \mathcal{E}$ denotes an ongoing relation from node $u \in \mathcal{V}$ to node $v \in \mathcal{V}$. For most cases in this thesis, we only consider simple graphs, which means that edges are all undirected i.e., $(u, v) \in \mathcal{E} \leftrightarrow (v, u) \in \mathcal{E}$. For the case of heterogeneous graphs, there are different types of nodes, meaning that the whole set of nodes can be divided into disjoint sets i.e., $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \ldots \cup \mathcal{V}_k$, where $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset, \forall i \neq j$. Therefore, a heterogeneous graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{R}$ is the set of nodes' types. In addition, nodes may have associated features, which are also known as the attributes. We usually denote the features of all nodes as a real-valued matrix $\mathbf{X} \in \mathbb{R}^{|V| \times m}$, where m is the number of features.

Besides the definitions of graphs, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is another vital concept for graph data. Indeed, it is a general and convenient method to represent the graph, including all nodes and edges. To represent a graph with an adjacency matrix, the nodes in the graph are ordered following how the users and items are indexed in the recommendation scenario. Afterwards, the edges are represented as entries in the adjacency matrix such that $\mathbf{A}[u, v] = 1$ if $(u, v) \in \mathcal{E}$ and $\mathbf{A}[u, v] = 0$ otherwise. $\mathbf{A}$ is commonly a symmetric matrix if and only if when it contains only undirected edges. Unlike a dense embedding matrix, an adjacency matrix is usually a sparse matrix that only indicates existing relations without encoding any other node-node similarity information. However, it is a reasonable starting point for graph-based models to explore the underlying relations and fetch the related nodes efficiently.

Building on the adjacency matrix $\mathbf{A}$, a Laplacian matrix (Merris, 1994) is another matrix representation of a graph. The most basic Laplacian matrix is defined as follows:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{2.16}$$

where $\mathbf{A}$ is the adjacency matrix defined above and $\mathbf{D}$ is the diagonal degree matrix.

Besides the basic unnormalised Laplacian, the normalised Laplacian and random-walk Laplacian (Fouss et al., 2007) are both popular variants, defined as follows:

$$\begin{aligned} \mathbf{L}_{\text{norm}} &= \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} \\ \mathbf{L}_{\text{RW}} &= \mathbf{D}^{-1} \mathbf{L}. \end{aligned} \tag{2.17}$$

Both of these matrices have similar properties as the basic Laplacian, but they differ in how they are normalised. Both variants can be incorporated in a GNN to encode different types of structural information of a graph. For example, when nodes have an unbalanced number of edges, $\mathbf{L}_{\text{norm}}$ is usually used to decouple the influence of nodes with large number of edges.

Graph Neural Networks (GNNs) are effective models for graph data. Specifically, given a graph $\mathcal{G}$, a GNN firstly initialise each node with an embedding $\mathbf{h}$. Then, the GNN will be trained with the so-called UPDATE and AGGREGATE functions, where the UPDATE function aims to pass incoming information from neighbours and the AGGREGATE function aims to aggregate these information, where a general UPDATE function is defined as follows (Hamilton, 2020):

$$\text{UPDATE}\left(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}\right) = \sigma\left(\mathbf{W}_{\text{self}}\, \mathbf{h}_u + \mathbf{W}_{\text{neigh}}\, \mathbf{m}_{\mathcal{N}(u)}\right), \tag{2.18}$$

where $\mathbf{h}_u$ is the embedding of the node u and $\mathcal{N}(u)$ denotes the set of neighbours of node u; $\sigma(\cdot)$ is an activation function e.g., a tanh or ReLU; $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$ denote two trainable parameter matrices and $\mathbf{m}_{\mathcal{N}(u)}$ is the aggregated information, defined as follows:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v. \tag{2.19}$$

By combing Equation (2.18) and Equation (2.19), the basic GNN (Kipf and Welling, 2017) is defined as follows:

$$\mathbf{h}_u^{(k)} = \sigma \left(\mathbf{W}_{\text{self}}^{(k)} \mathbf{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)} \right), \tag{2.20}$$

where k and b denote the k-th layer and bias term, respectively.

Similar to other deep neural networks, many GNNs use the Binary Cross-Entropy (known as BCE) loss function (He et al., 2017) to compute the loss value and optimise the networks.

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i) + (1 - \mathbf{y}_i) \cdot \log(1 - \hat{\mathbf{y}}_i), \tag{2.21}$$

where $\mathbf{y}_i$ is a ground truth value/label and $\hat{\mathbf{y}}_i$ is the predicted value.

We use Equation (2.20) and the BCE loss function (2.21) to describe the general function of a GNN. To enhance the effectiveness of the basic GNN (Kipf and Welling, 2017) many other variants of GNN such as SGC (Wu et al., 2019a) and HetGNN (Zhang et al., 2019a) have been proposed with different UPDATE and AGGREGATE functions, which are detailed in (Zhou et al., 2020). In the following three sections, we will cover the three main building components of this thesis, namely heterogeneous graph representation learning, graph pre-training and graph contrastive learning. In particular, we will introduce how to apply these techniques for the ranking-based recommendation task.

2.2.2 Heterogeneous Graph Representation Learning

Although the basic GNN defined in the previous section is effective to leverage graphs with a single type of nodes, it cannot handle the situation when the graph is heterogeneous. For example, in the recommendation system scenario, we want to encode multiple types of relations obtained from the side information of users and items. Broadly speaking, tasks that involve different types or nodes and relations need heterogeneous graph representation learning.

A simple approach to incorporate heterogeneous nodes or edges is to directly update the adjacency matrix $\mathbf{A}$ accordingly based on the multiple relations. For example, a modified variant of adjacency matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is proposed in (Liu et al., 2020) to encode two additional types of relations by extending the original matrix:

$$\mathbf{H}=\begin{bmatrix}\alpha\mathbf{S} & \mathbf{R}\\ \mathbf{R}^{\mathrm{T}} & \beta\mathbf{C}\end{bmatrix}, \tag{2.22}$$

where α and β are two relation-based constant parameters controlling the importance of relations contained in $\mathbf{S}$ and $\mathbf{C}$; $\mathbf{R}$ and $\mathbf{R}^{\mathrm{T}}$ are the bipartite graph with its transpose.

Then the corresponding Laplacian matrix of $\mathbf{H}$ is computed based on Equation (2.17) as follows:

$$\mathbf{L}^{\mathbf{H}}_{\text{norm}}=\mathbf{D}^{-\frac{1}{2}}\mathbf{H}\mathbf{D}^{-\frac{1}{2}}. \tag{2.23}$$

Since H has the same size as A, the corresponding GNN i.e., HGNR (Liu et al., 2020) has a similar efficiency to the basic GNN while being more effective for encoding more relations. However, $\mathbf{H}$ is non-symmetrical, which might affect the effectiveness of message passing. Furthermore, from Equation (2.22), we can see that this approach can only be applied to two additional relations. To address this issue, we propose a more generalised method using the social graph pre-training method, which will be detailed in Chapter 3.

2.2.3 Graph Pre-training

Pre-training techniques have become standard practice in deep learning. They are used to encode general information from a relatively large corpus of samples such as texts or images. After a model is pre-trained, it can be fine-tuned for some related downstream tasks (Erhan et al., 2010). For example, the BERT model if pre-trained on the Wikipedia dataset can be used for the downstream textual and semantic classification tasks (Devlin et al., 2018). In the case of GNNs, pre-training a GNN using the neighbourhood reconstruction loss is an effective strategy to improve the performance on a downstream task. For example, we can pre-train a GNN to reconstruct missing edges in the graph before fine-tuning on a node classification loss (Hao et al., 2021a). In (Velickovic et al., 2019), Deep Graph Infomax (DGI) is proposed to maximise the mutual information between node embeddings $\mathbf{z}_u$ and graph embeddings $\mathbf{z}_{\mathcal{G}}$. The objective function of DGI is defined as follows:

$$\mathcal{L}=-\sum_{u\in\mathcal{V}_{\text{train}}}\mathbb{E}_{\mathcal{G}}\log\left(D\left(\mathbf{z}_u,\mathbf{z}_{\mathcal{G}}\right)\right)+\gamma\mathbb{E}_{\tilde{\mathcal{G}}}\log\left(1-D\left(\tilde{\mathbf{z}}_u,\mathbf{z}_{\mathcal{G}}\right)\right), \tag{2.24}$$

where $\mathbf{z}_u$ is the embedding of node u generated based on graph $\mathcal{G}$, while $\tilde{\mathbf{z}}_u$ denotes an embedding of node u generated based on a corrupted graph $\mathcal{G}$ i.e., $\tilde{\mathcal{G}}$. Here, $D(\cdot)$ is a discriminator function, which is a neural network used to predict whether the node embedding comes from the original graph $\mathcal{G}$ or from the corrupted graph $\tilde{\mathcal{G}}$.

There are many approaches to corrupting a graph. For example, one can randomly modify the features of nodes or randomly drop the existing edges. The underlying intuition here is that the GNN model should have the ability to distinguish between the original graph and its corrupted counterpart (Hamilton et al., 2017b).

Although effective, existing graph pre-training techniques do not generalise to the recommendation task (Meng et al., 2021). The main reason is that general graph pre-training techniques treat those corrupted nodes as different views of the original nodes. However, in the recommendation scenario, a corrupted user (randomly dropping some of his/her interacted items) cannot be simply interpreted as a different view in the graph learning task because other users' interaction records may overlap with the corrupted one. Therefore, to generalise the graph pre-training model for the recommendation task, in Chapter 4, we propose to pre-train a GNN using the side information of users and items, then fine-tune the model with an existing recommender system.

2.2.4 Graph Contrastive Learning

Contrastive learning is another emerging learning paradigm that has become popular for image classification. Specifically, contrastive learning involves a set of techniques, including data augmentation, self-supervised or discriminative functions and contrastive negative sampling that can be used to learn the relations between positive and negative samples (Qiu et al., 2020; Yu et al., 2022a). Here, we make a list of popular data augmentation techniques (Chopra et al., 2005).

- **Node dropout**. This will randomly discard a certain percentage of nodes and their corresponding edges among the given graph $\mathcal{G}$. Each node's dropout rate follows a default uniform distribution (or any other distribution).

- **Edge dropout**. This will modify the relations in a graph $\mathcal{G}$ by randomly adding or dropping a certain percentage of edges.

- **Feature masking**. This method randomly masks out some features of nodes and the aim is to recover the masked features based the remaining ones.

- **Subgraph sampling**. This method is based on the random walk technique given a graph $\mathcal{G}$. It assumes that the semantics of $\mathcal{G}$ can be largely preserved in its remaining structure.

Objective functions used by graph contrastive learning models are usually variants motivated by the Noise Contrastive Estimation (NCE) or InfoNCE loss:

$$\mathcal{L} = -\log \frac{\exp\left(\operatorname{sim}\left(\boldsymbol{z}_{n,i}, \boldsymbol{z}_{n,j}\right)/\tau\right)}{\sum_{n'=1, n' \neq n}^{N} \exp\left(\operatorname{sim}\left(\boldsymbol{z}_{n,i}, \boldsymbol{z}_{n',j}\right)/\tau\right)}, \tag{2.25}$$

where $\boldsymbol{z}_{n,i}$ and $\boldsymbol{z}_{n,j}$ denote two positive samples for the n-th graph in a batch. Besides, τ is a constant hyperparameter.

To incorporate the graph contrastive learning technique for enhanced recommendations, we propose to leverage the contrastive negative sampling method to sample more informative negative items. After which, a contrastive objective function is used to leverage multiple negative samples. Details of our proposed model will be presented in Chapter 5.

2.3 Conclusions

In this chapter, we have briefly introduced the important concepts and preliminaries of the ranking-based recommender systems. First, we listed different taxonomies of recommender systems, after which we focused on collaborative filtering-based, content-based and hybrid approaches. Then we introduced different evaluation methods of recommender systems, where we detailed the offline evaluation paradigm used throughout this thesis.

Next, we explored graph representation learning (GRL) by first introducing its advantages and bridging the relations between GRL and the recommendation task. Afterwards, we covered some necessary preliminaries to understand GRL, including the definitions of graphs, graph Laplacian and a basic Graph Neural Network. Finally, we introduced the three primary building components, namely heterogeneous graph representation learning, graph pre-training and graph contrastive learning. Specifically, we will use these graph-based techniques to alleviate the challenges mentioned in Section 1.1 and hence validate our proposed thesis statement (see Section 1.2).

In the next chapter, we begin with our proposed recommender system that leverage the social graph pre-training technique, where the graph containing user-item and user-user interactions is a form of heterogeneous graph as described in Section 2.2.2.

Chapter 3

Social Graph Recommendation

3.1 Introduction

In our thesis statement (see Section 1.2), we postulated that we can incorporate social relations using heterogeneous graph representation learning in order to enhance the overall recommendation effectiveness. In Section 2.2, we also introduced how to construct the heterogeneous graph and the corresponding adjacency matrix. Moreover, as argued in Chapter 1, the *cold-start* problem has been recognised as a long-standing issue of collaborative filtering-based approach. We aim to propose a social graph recommender system that not only improves the overall recommendation performance but also benefits *cold-start* users compared with classical models and other deep learning-based models.

As introduced in Section 2.2.3, the pre-training technique has been commonly used to optimise deep models by providing them with an effective initialisation (Erhan et al., 2010, 2009; Van Engelen and Hoos, 2020). Such a pre-training technique has been shown to lead to state-of-the-art performances when the pre-trained model is further *fine-tuned to address* downstream Natural Language Processing (NLP) (Brown et al., 2020; Devlin et al., 2018; Edwards et al., 2020; Zheng et al., 2020a) or information retrieval tasks (Chakraborty et al., 2020; Ma et al., 2021; Meng et al., 2022). However, this effective technique has been less studied in recommender systems possibly due to the limitations in the existing datasets. For example, in the NLP tasks, one unsupervised deep language model can be pre-trained from unlabelled texts (e.g., Wikipedia) and fine-tuned for a supervised downstream task (Brown et al., 2020; Devlin et al., 2018). In contrast, in the recommendation scenario, each dataset contains its specific information about the corresponding users and items, but no other ground truth knowledge such as Wikipedia could be leveraged from outside the dataset to help estimate the users' preferences and items' attributes.

NCF (He et al., 2017) has been proposed to pre-train the recommendation model with a Multi-Layer Perceptron (MLP) (Ramchoun et al., 2016). Although effective, the MLP module does not consider other available auxiliary side information, such as the social relations among users (Wu et al., 2020a; Zhao et al., 2014) or the items' timestamps (Li et al., 2020b; Song et al., 2016), therefore the applied pre-training technique of NCF is limited in providing the *cold-start* users with a better initialisation. Since the social relations among users have been shown to be essential in enhancing the recommendation performance and alleviating the *cold-start* problem introduced in Section 2.1, we propose to incorporate the social relations and the interaction data at the pre-training stage so that a better initialisation can be obtained for those users who have fewer interactions.

As we have introduced in Section 2.2, Graph Neural Networks (GNNs), a class of deep learning models (Wu et al., 2020c; Zhou et al., 2020), have been used to aggregate the nodes' information from their neighbourhoods so as to learn an overall structure from a given graph's type of data. Indeed, while GNNs have been previously exploited to enhance general recommender systems (He et al., 2020; Wang et al., 2019c), they have only been recently studied under the pre-training scheme (Hao et al., 2021a). In this chapter, we devise a novel Social-aware Gaussian Pre-trained model (SGP), which leverages a graph pre-training technique to encode the users' social relations and attempts to search for a relative optimised solution based on the learned social-aware initialisation during the fine-tuning stage. At the first stage, we pre-train a light GNN model with additional social information to give users/items meaningful initialised embeddings. Given the neighbourhood aggregation property of the GNN model, incorporating the social relations enables socially-connected users to become closer in this latent space through the aggregation process. Then, in the fine-tuning stage, we load the obtained pre-trained embeddings and re-train the model for further recommendations. The most straightforward approach for leveraging these pre-trained embeddings and decoding the social information is to directly reload them. However, it is essential to note that the interaction data, which will be used in the second stage, has already been exploited at the pre-training stage. Therefore, the reuse of the interaction data might cause the overfitting problem (Erhan et al., 2010). Hence, we propose to distil the information from the pre-trained model so that we can later reconstruct meaningful embeddings.

Since all embeddings can be viewed as probability distributions, an intuitive solution for distilling information from those pre-trained embeddings is to follow existing works (He et al., 2020; Rendle et al., 2020) and use Gaussian distributions to model the embeddings. Indeed, different from existing models (He et al., 2020; Wang et al., 2019c) which use the random initialisation, we expect to need more complex distributions to capture prior knowledge from the pre-training stage. Therefore, we propose to apply the Gaussian Mixture Model (GMM) (Reynolds, 2009), which assumes that all the data points are sampled from a mixture of a finite number of Gaussian distributions. By leveraging this well-developed GMM, our proposed method is devised to factorise those pre-trained embeddings into a finite number of Gaussian distributions, where this number is pre-defined and each distribution could be viewed as a specific interest of users or a particular characteristic of items.

To summarise, this chapter makes the following contributions:
• We devise a two-stage end-to-end social pre-trained recommendation model, SGP, which uses the GNN model to leverage social information. We show that SGP can achieve state-of-the-art effectiveness on three real-world datasets of user-item interaction and social relations.
• We leverage the Gaussian Mixture Model to distil information from the pre-trained embeddings for the downstream recommendation task.
• Our proposed model is shown to significantly outperform 13 strong baselines from the literature, while being particularly useful for cold-start and extreme cold-start users (newly registered users).

The remainder of this chapter is organised as follows. In Section 3.2, we position our work in the literature. Section 3.3 introduces all relevant notions used in this chapter and formally defines the detailed architecture of our SGP model. The experimental setup and the results of our empirical experiments are presented in Sections 3.4 and 3.5, respectively, followed by the conclusions of this chapter in Section 3.6.

3.2 Related Work

In the following, we discuss relevant pre-trained models (Section 3.2.1), graph-based recommender systems (Section 3.2.2) as well as the data augmentation and cross-domain transferring techniques used in recommendation (Section 3.2.3).

3.2.1 Pre-trained Models

The pre-training technique has become an emerging research topic especially in the field of NLP. Pre-trained language models such as the BERT (Devlin et al., 2018) and the more recent GPT-3 (Brown et al., 2020) models have demonstrated their robust performance on different downstream NLP tasks. Through pre-training, a language model can learn contextualised embeddings for tokens from a large corpus of texts, so that these tokens can be reused for subsequent tasks with enhanced performances. Such models can then be later fine-tuned for a new downstream task, thereby enhancing the overall performance of the corresponding model and outperforming other handcrafted models.

As mentioned in Section 3.1, the pre-training technique was also adopted in recommendation models. Indeed, He et al. (2017) proposed the Neural Collaborative Filtering (NCF) model to introduce a novel deep learning-based method to the recommender systems community, which has attracted a substantial attention from researchers since then. The most important contribution of the NCF model is that it successfully incorporates the multi-layer perceptron (MLP) module, which can in theory effectively approximate various types of prediction functions (Sifaoui et al., 2008). However, it is noticeable that this NCF model also uses a generalised matrix factorisation (GMF) module to generate pre-trained embeddings, which limits the NCF model from incorporating auxiliary information at the pre-training stage. To this end, we propose to use instead the GNN technique to replace the MLP pre-trained module due to the former's ability of supporting the incorporation of heterogeneous relations such as the relations among users as well as the users' interaction data. Moreover, the GNN technique introduced in Section 2.2, has been initially devised to implement the node classification and link prediction tasks. Therefore, GNNs are expected to perform better than the GMF module on aggregating similar users and items (Wang et al., 2018a). Hence, compared with the GMF module, when the GNN model is used for the pre-training stage, the embeddings of the socially related users can be better aggregated in closer proximity in the latent space. Apart from using the GMF module to pre-train on the interaction data, Wen et al. (2018) introduced a linear pre-trained recommender using the network embedding method. However, their proposed model failed to leverage the multi-hops social relations (i.e., a friend's friends), which can be seamlessly addressed by the GNN methods. A study by Hao et al. (2021a) is a more recent related work to ours, which tried to tackle the *cold-start* problem by pre-training the recommendation model in a meta-learning setting. However, the contribution of their work is to use the fundamental structure of the interaction graph, which is different from our research goal of using social information to obtain better initialised users and items' representations. In addition, existing works have leveraged other GNN pre-training (see Equation (2.24)) and contrastive pre-training techniques for sequential (Li et al., 2021; Xiao et al., 2021; Xie et al., 2020) and conversational (Wong et al., 2021) recommender systems, respectively, which are distinct from our proposed recommender SGP.

3.2.2 Graph-based Recommendation

As mentioned in Section 2.2, GNNs have the so-called neighbourhood aggregation property, which has been demonstrated to be effective in enhancing the performances of recommender systems. Therefore, various graph-based recommenders (He et al., 2020; van den Berg et al., 2017; Wang et al., 2019c; Ying et al., 2018; Yu et al., 2022b,c) have been proposed and they have achieved state-of-the-art performances through the development of GNNs (Kipf and Welling, 2017; Xinyi and Chen, 2018; Zhang et al., 2019a). Since the user-item interaction data can be intrinsically depicted as an interaction graph, the GNN technique and its variants have been seamlessly applied in various recommender systems and achieved good performances. For example, NGCF (Wang et al., 2019c), has been shown to outperform many competitive baselines by incorporating a Graph Convolutional Network (GCN) to encode the collaborative signals and to model the users' and items' embeddings. Building on NGCF, the LightGCN model (He et al., 2020) further enhanced its recommendation performance by eliminating redundant neural components from NGCF. Recently, GF-CF (Shen et al., 2021) was proposed to further enhance the performance by incorporating a graph filtering method. However, GF-CF is not an embedding-based model, hence it cannot be adapted to normal pre-training methods. In addition, Diffnet (Wu et al., 2020a) was proposed to encode the social relations between users. However, it only considers the plain user-user social graph while ignoring the heterogeneous network consisting of both the social relations and user-item interactions. Other variants of LightGCN including SGL (Wu et al., 2021) and UltraGCN (Mao et al., 2021) have also achieved competitive performances. However, they incorporate memory-consuming data augmentation methods as described in Section 2.2.4, which will be more challenging if side information is also considered. Inspired by the generalisability of LightGCN and its good trade-off between effectiveness and efficiency, we also adopt the simplified GCN (Wu et al., 2019a). Moreover, we incorporate the social information into the embedding generation and updating process, which enables our proposed SGP model to encode the social relations into the users' embeddings. We will show how this auxiliary social information benefits our model by allowing it to obtain a better model initialisation thereby alleviating the *cold-start* problem.

3.2.3 Data Augmentation and Cross-Domain Recommendation

Data augmentation techniques were well developed in Recommender Systems before the emergence of deep recommender systems (Zhang et al., 2019c). The classic hybrid models (Basu et al., 1998; Cotter and Smyth, 2000; Melville et al., 2002) attracted attention by applying *feature augmentation* to combine content-based and collaborative filtering recommenders (see Section 2.1.1.3). For example, a hybrid recommender might combine a one-hot encoding technique with matrix factorisation, as performed by (Cotter and Smyth, 2000; Melville et al., 2002; Ning and Karypis, 2011). These classic models were shown to deliver effective rating predictions in different domains, such as in movie recommendations (Basu et al., 1998; Melville et al., 2002) and TV recommendations (Cotter and Smyth, 2000). Although the classic hybrid models are effective, as we have mentioned in Section 2.1, implicit feedback is increasingly becoming the most commonly leveraged interaction data because users are typically less willing to give explicit feedback such as ratings (Rendle et al., 2009; Zhang et al., 2019c). Therefore, these hybrid rating prediction models (Basu et al., 1998; Cotter and Smyth, 2000; Melville et al., 2002) are becoming less popular compared with the ranking-based models such as the BPR model described in Section 2.1.1.1 and its variants (Parambath et al., 2021; Wang et al., 2019c; Zhao et al., 2014), which are designed to leverage the more abundant implicit feedback. According to the taxonomy introduced in Section 2.1, our proposed SGP model can also be seen as a hybrid recommender system, which augments users' and items' embeddings through the pre-training technique. It differs from the aforementioned classic hybrid models in leveraging a graph neural model, which models the complex relationships between users and their friends through multi-hops neighbourhood aggregation.

On the other hand, cross-domain recommender systems have been proposed to extract or transfer knowledge from a source domain to a target domain (Li and Tuzhilin, 2020; Man et al., 2017; Manotumruksa et al., 2019; Tang et al., 2012; Wang et al., 2021c; Yu et al., 2022d). There are two different categories of cross-domain recommender systems, namely overlapping (Farseev et al., 2017; Sanz-Cruzado et al., 2020; Tang et al., 2012) or non-overlapping (Manotumruksa et al., 2019; Yu et al., 2022d; Zhao et al., 2020a) recommenders, depending on whether they have shared users and/or items between the source domain and target domain. Our work is more related to the overlapping cross-domain models, since we also have shared users between the target domain (recommendation) and the source domain (social network). Although both our work and the overlapping cross-domain recommendation models have shared users, in the cross-domain recommendation scenario, the actual tasks for both the target and source domains are to make recommendations. In contrast, in our work, the social domain is only used to supply auxiliary information. Therefore, we argue that it is not appropriate to compare our work with those addressing the cross-domain recommendation.

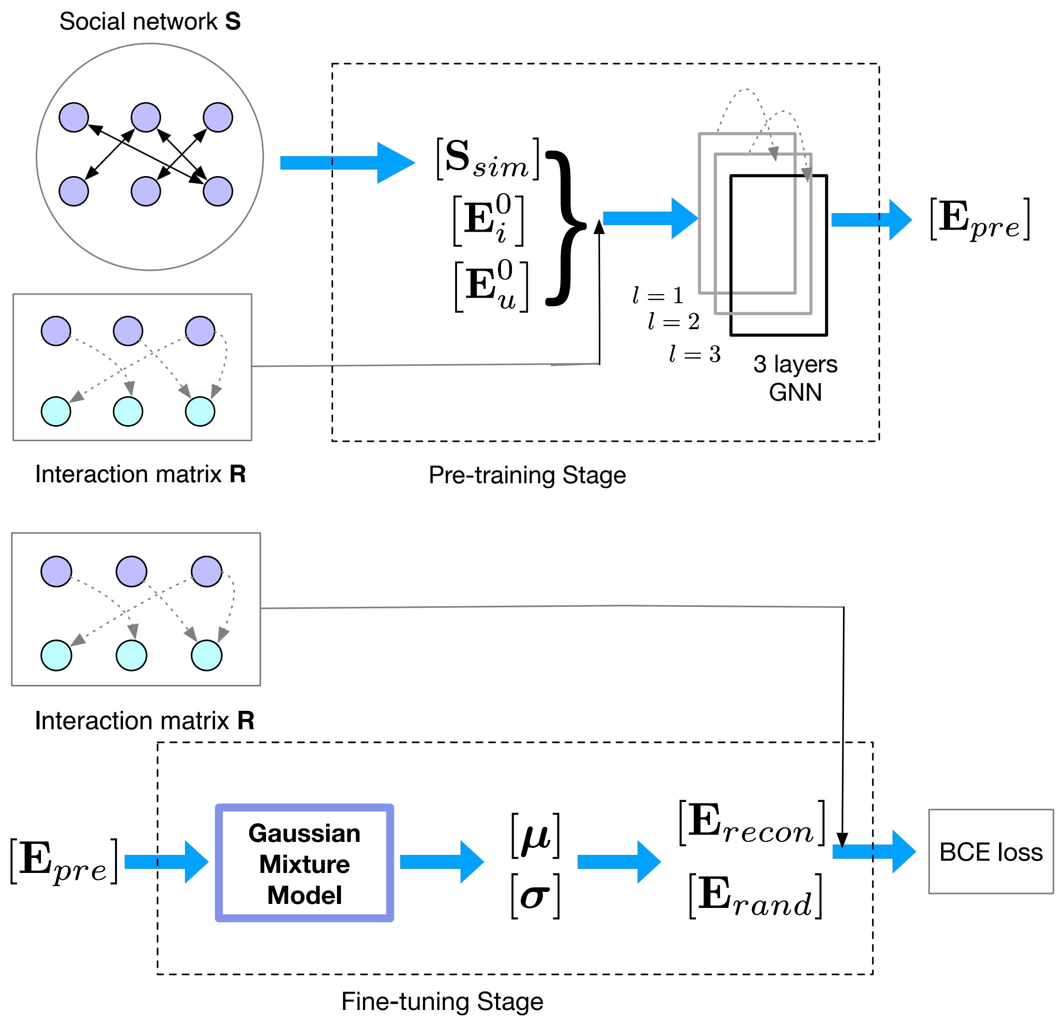


Figure 3.1: An illustration of our SGP model, where the pre-training stage and the fine-tuning stage are located above and below, respectively.

Table 3.1: Main Notations Used in this chapter.

Notation	Description
$\boldsymbol{R}$	the matrix of implicit feedback data
$\mathcal{U}, \mathcal{I}$	the sets of users and items
$\mathbf{S}$	the social network matrix of all users
$\mathbf{S}_{sim}$	the social similarity matrix of all users
$\mathbf{E}$	the embedding matrix
$\mathbf{E}_{pre}$	the pre-trained embedding matrix
$\mathbf{E}_{recon}$	the reconstructed embedding matrix
$\mathbf{e}$	the embedding vector
$\mathcal{L}$	the Laplacian matrix
$\mathbf{W}$	the learnable weight matrix
μ, σ	the mean and standard deviation of an embedding
$\boldsymbol{\mu}$, $\boldsymbol{\sigma}$	the mean and standard deviation matrices of an embedding matrix
ζ	the loss
M, N	the number of users and the number of items
k	the number of Gaussian distribution
Θ	the trainable model parameters
λ	the controlling factor of the L_2 regularisation
y_{ui}, $\hat{y}_{ui}$	the observed interaction and the predicted interaction
$\mathcal{N}$	the normal distribution
$\mathbf{I}$	the identity matrix

3.3 Model Architecture

Our proposed SGP model consists of two main stages: (1) a social-aware pre-training stage, where a multi-layer GNN is employed to generate the pre-trained embeddings and (2) an information distillation stage, where we incorporate the Gaussian Mixture Model (GMM) to distil information from the pre-trained embeddings for the subsequent model's training and generation of recommendations. In the following, we first define some preliminaries in Section 3.3.1. Next, Section 3.3.2 describes how to incorporate the social relations and a light GNN model to propagate the social information into users' and items' embeddings. Finally, in Section 3.3.3, we demonstrate how to employ the GMM to distil the social information from those pre-trained embeddings for the subsequent training and the production of final recommendations. To clearly illustrate our model, Figure 3.1 depicts the overall structure of our proposed SGP model, where the upper and bottom regions describe the pre-training and fine-tuning stages, respectively.

3.3.1 Preliminaries

In this section, we introduce the notations used across this chapter. Throughout this chapter, we use calligraphy typeface alphabets to denote sets (e.g., $\mathcal{U}$ is the set of users). Matrices and vectors are denoted by bold letters with uppercase letters representing matrices and lowercase letters representing vectors. In Table 3.1, we summarise main notations used in this article for a fast reference.

As introduced in Section 2.1, our task is to highly rank relevant items for each user given their historical interactions. Specifically, in this chapter, we consider social network information as an important source for improving the recommendation performance, especially when a user does not have enough interactions with items. Therefore, in addition to the implicit feedback $\mathbf{R}$ introduced in Section 2.1.1.1, we set $\mathbf{S} \in \{0,1\}^{M \times M}$ be the user-user social network matrix, where the content of $\mathbf{S}$ represents the presence of social connections between each pair of users and M denotes the number of users.

3.3.2 The Pre-training Stage

A GNN model can leverage the nodes' information and their corresponding relational information in a graph by effectively aggregating information from each node's neighbours. In the recommendation scenario, each node represents either a user or an item. Therefore, suppose that only the interaction information is given, then each user node's neighbours could correspond to those items that have been interacted with by this user (and vice-versa for an item node). On the other hand, when the social network information is also available, then the user's neighbours can be his/her interacted items or friends. This friendship information is also important for the recommender system, because users are more likely to interact with those items that have been previously interacted with by their social neighbours (Yang et al., 2014). Hence, the users' available social relations provide useful insights for inferring their interests and predicting the items that they will interact with.

To effectively propagate the friends' information into each user who is socially connected, we firstly initialise each user with a randomised embedding vector $\mathbf{e}_u$ to represent his/her interests. Similarly, we can assign each item with a randomised embedding vector $\mathbf{e}_i$. We set $\mathbf{E}_u$ to be the embedding matrix containing all latent vectors of users and $\mathbf{E}_i$ to be the embedding matrix for all items. A graph neural network (GNN) can be employed here to aggregate the users' social information for each user node. By stacking multi-layers GNNs, we can propagate high-order connectivities of social relations from multi-hop neighbours. In our case, we use the most commonly used 3-layers GNNs to capture reasonable depths in the social connectivities while avoiding the possible over-smoothing effect of the GNN.

To use multi-layers GNNs (Wu et al., 2019a), we rely on the Laplacian matrix $\mathcal{L}$ defined by Equation (2.16), so that the information propagation and convolution functions described in Section 2.2.1 can be executed effectively in a matrix multiplication form. Different from the NGCF (Wang et al., 2019c) model, which only tries to encode the interaction signal into both the users' and items' embeddings, our model focuses on the social information propagation in the pre-training stage. Furthermore, to further improve the Diffnet++ model (Wu et al., 2020a), which only incorporates the plain social relation links, our model pre-computes the cosine similarity between each user's social relation vector[1]. These vectors constitute one-dimensional binarised vectors indicating the social links between users. Therefore, building on this advanced user-user social similarity graph $\mathbf{S}_{sim}$, the GNN function can better classify similar users. Given the user-user social graph $\boldsymbol{S} \in \{0,1\}^{M \times M}$, we can pre-compute the social similarity graph S_{sim} as follows:

$$\mathbf{S}_{sim} = \Big(\frac{\mathbf{S} \cdot \mathbf{S}^{\mathrm{T}}}{\sqrt{\mathrm{diag}(\mathbf{S} \cdot \mathbf{S}^{\mathrm{T}})}}\Big)^{\mathrm{T}} \cdot \frac{1}{\sqrt{\mathrm{diag}(\mathbf{S} \cdot \mathbf{S}^{\mathrm{T}})}} \tag{3.1}$$

where T represents the transpose of a matrix and $\mathrm{diag}()$ computes the diagonal matrix of the corresponding matrix. Entries of $\mathbf{S}_{sim}$ are set to 0 if the corresponding user has no social relationships in S. Given the similarity graph $\mathbf{S}_{sim}$, we can derive its corresponding Laplacian matrix $\boldsymbol{\mathcal{L}}_{sim}$ follows:

$$\boldsymbol{\mathcal{L}}_{sim} = \mathrm{diag}(\mathbf{S}_{sim})^{-\frac{1}{2}} \cdot \mathbf{S}_{sim} \cdot \mathrm{diag}(\mathbf{S}_{sim})^{\frac{1}{2}} \tag{3.2}$$

With the Laplacian matrix $\boldsymbol{\mathcal{L}}_{sim}$, we can present the embedding updating function of our proposed SGP model:

$$\mathbf{E}_u^{(l)} = \phi\Big((\boldsymbol{\mathcal{L}}_{sim} + \mathbf{I}) \cdot \mathbf{E}_u^{(l-1)} \cdot \mathbf{W}^{(l)}\Big) \tag{3.3}$$

where $\mathbf{I}$ is the identity matrix, $\phi(\cdot)$ is the LeakyReLU (Maas et al., 2013) function (also used in other graph-based recommenders (van den Berg et al., 2017; Wang et al., 2019c)) and $\mathbf{W}$ is a trainable weight matrix.

Starting from a randomly initialised $\mathbf{E}_u^0$, we stack 3 layers of the GNNs given in Equation (3.3) and update the embeddings for each user correspondingly. Following the LightGCN model, we discard those redundant neural components from the variant of GCN used in NGCF. Indeed, the self-connection setup, i.e., adding the dot product of an embedding with itself into Equation (3.3), was initially proposed in (Wang et al., 2019c) to keep each node's original information and to avoid being possibly overloaded with information from the nodes' neighbours. However, this self-connection was later demonstrated in (He et al., 2020) to bring no benefit to recommendation performance; instead, it will reduce the training efficiency. Hence, we choose to also remove this redundant part in our embedding updating function following the existing work (He et al., 2020).

We follow the GNN technique proposed in the aforementioned LightGCN model to update and aggregate the users' embeddings. However, different from LightGCN, which uses the GNN to incorporate the interaction data, we only incorporate the social information propagation. Similar with other graph-based recommendation models (Hamilton et al., 2017a; Mao et al., 2021; Wang et al., 2019c; Wu et al., 2020a), we keep the interaction data as the ground truth for supervising the pre-training of our model. At each training epoch, Equation (3.3) is invoked to

[1] We use the cosine similarity because it can be efficiently computed for our sparse user-user matrix. In addition, the similarities between social relation vectors do not need estimating the magnitude, hence other similarity measures - e.g., the dot product or the Euclidean similarities, may not be appropriate.

Algorithm 1: The pre-training stage

```
Input: Interaction matrix $\mathbf{R}$; Social network matrix $\mathbf{S}$
Output: Pre-trained embedding $\mathbf{E}_{pre}$.
Initialise embeddings $\mathbf{E}^0$ and other learnable parameters $\Theta$;
Compute $\mathcal{L}_{sim}$ according to Equation (3.1) & (3.2);
while early stopped do
    $\zeta_{BCE} = 0$;
    for each training instance in $\mathbf{S}$ do
        Propagate social information according to Equation (3.3);
    end
    for each training instance in $\mathbf{R}$ do
        Compute epoch loss $\nabla\zeta$ according to Equation (3.4);
    end
    $\zeta_{BCE} \leftarrow \zeta_{BCE} + \nabla\zeta$;
    Update $\Theta$, $\mathbf{E}$;
end
```

perform the social aggregation, after which we use the binary cross-entropy (BCE) loss introduced in Section 2.2.1 as the objective function:

$$\zeta_{BCE} = -\sum_{(u,i)\in\mathbf{R}} y_{ui} \cdot \log\left(\hat{y}_{ui}\right) + \left(1 - y_{ui}\right) \cdot \log\left(1 - \hat{y}_{ui}\right) + \lambda \left\|\Theta\right\|^2 \tag{3.4}$$

where y_{ui} is the observed interaction, $\hat{y}_{ui}$ is the predicted interaction, which is a dot product of the item embedding and the user embedding obtained from Equation (3.3), while $\Theta = \{\{\mathbf{E}_u^l, \mathbf{E}_i^l, \mathbf{W}^l\}_{l=1}^3\}$ denotes all trainable model parameters and λ controls the L_2 regularisation strength to prevent overfitting.

As a result, our pre-trained embeddings $\mathbf{E}_{pre}$ can be obtained by minimising the objective function in Equation (3.4). For a better understanding, the training framework of our pre-training stage is summarised in Algorithm 1.

3.3.3 The Fine-tuning Stage

After detailing the pre-training stage (Section 3.3.3.1), we first present the information distillation stage, where we describe how to use the Gaussian Mixture Model to distil hierarchical relations from the pre-trained embeddings $\mathbf{E}_{pre}$. Then, we demonstrate how to use the reconstructed embeddings $\mathbf{E}_{recon}$ for the final recommendations (Section 3.3.3.2). Similar with the previous section, we summarise the training framework of the fine-tuning stage in Algorithm 2.

Algorithm 2: The fine-tuning process

```
Input: Rating matrix $\mathbf{R}$; Pre-trained embedding $\mathbf{E}_{pre}$; Pre-defined integer $k$.
Output: The recommended list for each user.
Inherit $\mathbf{E}_{pre}$ for initialising embeddings;
Use GMM to factorise $\mathbf{E}_{pre}$ according to Equation (3.6).
Randomly sample from $k$ Gaussian distributions;
Generate reconstructed embeddings $\mathbf{E}_{recon}$ according to Equation (3.7).
Initialise other learnable parameters $\hat{\Theta}$;
while not convergent do
    $\zeta_{BCE} = 0$;
    for each training instance in $\mathbf{R}$ do
        Compute epoch loss $\nabla\zeta$ according to Equation (3.4);
    end
    $\zeta_{BCE} \leftarrow \zeta_{BCE} + \nabla\zeta$;
    Update $\hat{\Theta}$, $\mathbf{E}_{recon}$;
end
Do recommendation to find the recommended list based on the trained
  embeddings according to Equation (3.8);
```

3.3.3.1 Information Distillation Stage

By using Equation (3.4) for the pre-training and Equation (3.3) for the social aggregation, we aim to encode social information into our pre-trained embeddings $\mathbf{E}_{pre}$. The latter constitutes the obtained embedding matrix from optimising Equation (3.4). However, it is not obvious how these pre-trained embeddings can be reused. Since we have already used the interaction data as the ground truth during the pre-training stage , directly reloading these embeddings is likely to cause either an overfitting or a marginal improvement. Therefore, in the information distillation stage, we propose to distil information from these pre-trained embeddings. Next, we concatenate these distilled information at the tail of a randomly initialised embedding to add more generalisation to the final embeddings.

Before extracting useful information from the pre-trained embeddings, we propose to model each user's or item's latent vector as a multi-Gaussian distribution. This is consistent with the implementation details of existing works (He et al., 2020; Rendle et al., 2009, 2020; Wang et al., 2019c). Indeed, the matrix factorisation technique introduced in Section 2.1.1.1, can be interpreted as the search for the best fitted distribution for the users and items in a latent space. This is why in most implementations (He et al., 2020; Rendle et al., 2020), the embeddings are initialised with a Gaussian distribution with a given mean (μ) and standard deviation (σ) e.g., $\mu = 0$ and $\sigma^2 = 0.1$. As discussed in Section 3.1, we expect the pre-training stage to capture hierarchical relations between users and items (Devlin et al., 2018; Reimers and Gurevych, 2019). However, a standard Gaussian distribution cannot represent these learned complex relations from the pre-trained embeddings $\mathbf{E}_{pre}$, because its low representational power (Reynolds, 2009) limits its ability to convey the users' different preferences and their complex social relations, thereby potentially leading to an information loss. Hence, a mixture model is needed to leverage the possible multivariate Gaussian distributions learned from the pre-training stage and to avoid any possible information loss.

With the aforementioned proposal, we employ a well-developed statistical analysis tool, the Gaussian Mixture Model (GMM) (Reynolds, 2009), which can effectively decompose a multivariate Gaussian distribution into multiple (i.e., k) Gaussian distributions, where k is pre-defined. Since we assume that all users' and items' embeddings correspond to combinations of Gaussian distributions, we can extract meaningful information from these embeddings by analysing each pair of (μ, σ), as these represent each user's most important preferences or each item's most important characteristics.

As a result, we use the following equation to decompose the embeddings obtained from the pre-training stage in Section 3.3.2:

$$p(e'_u) = \sum_{i=1}^{k} \mathbf{W}_i \cdot \mathcal{N}(e_u | \mu_i, \sigma_i) \tag{3.5}$$

where $\mathbf{W}$ is a weight matrix representing the importance of each decomposed distribution, $\mathbf{e}'_u$ is the predicted user embedding and $\mathbf{e}_u$ is the original user embedding; k is an integer number, which defines how many Gaussian distributions should be obtained by decomposing the original embedding. A similar equation can also be applied to an item's embedding vector.

From the $\mathbf{E}_{pre}$, we use the following equation to calculate all pairs of μ and σ:

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = \mathrm{GMM}(\mathbf{E}_{pre}, k) \tag{3.6}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are two matrices consisting of all (μ ,σ) pairs for each user and item.

After employing the GMM to those pre-trained embeddings, we obtain k pairs of μ and σ for each user and item, from which we have enough prior knowledge to reconstruct the embeddings containing the social information encoded at the pre-training stage. For each user or item, we use the obtained (μ ,σ) pairs to generate k Gaussian distributions, where we randomly sample the same number of elements from each distribution to reconstruct socially-aware embeddings. After that, we obtain the reconstructed embeddings $\mathbf{E}_{recon}$ as follows:

$$\mathbf{E}_{recon} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \tag{3.7}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are both obtained from Equation (3.6).

3.3.3.2 Model Training and Recommendation

After obtaining the reconstructed embeddings $\mathbf{E}_{recon}$, we use again the BCE loss function (see Equation (2.21)) to train the model but, this time, the model is initialised with $\mathbf{E}_{recon}$ instead of a random matrix. We concatenate the reconstructed embeddings $\mathbf{E}_{recon}$ with the randomly initialised embeddings to represent the users' preferences and items' characteristics. Therefore, the model is less likely to fall into the same relative optimised solution within the pre-training stage. To recommend items of interests to a user, we compute the dot product of the concatenated trained embeddings of this user with the trained embeddings of all items in the corpus. Hence, our proposed *Social-aware Gaussian Pre-trained* model (SGP) is devised to predict the interaction $\hat{y}_{ui}$ between user *u* and item *i* as follows:

$$\hat{y}_{ui} = (\mathbf{e}_{u-recon} \parallel \mathbf{e}_{u-rand}) \odot (\mathbf{e}_{i-recon} \parallel \mathbf{e}_{i-rand}) \tag{3.8}$$

where $\parallel$ denotes the concatenation and $\odot$ is the dot product. The obtained list of scores are then used to identify the items that a given user will be interested to interact with.

3.4 Datasets and Experimental Setup

To evaluate our proposed SGP model, we perform experiments on three public datasets: Librarything[2], Epinions[3] and Yelp[3]. These datasets are widely used in the recommender systems community. Librarything is a book review dataset, Epinions is a general customer review dataset, while Yelp is a venue check-in dataset. Table 3.2 provides the statistics of the three used datasets. In the following, we aim to address the following research questions:
RQ3.1. Can we use the GNN model to leverage the social information and generate pre-trained embeddings for both users and items, thereby improving the overall recommendation performance?
RQ3.2. Can we employ the Gaussian Mixture Model to distil information from the pre-trained embeddings and further enhance the recommendation performance?
RQ3.3 Does our SGP model help in alleviating the cold-start problem, especially for those extreme cold-start users?
RQ3.4. What is the impact of using the social relations on the pre-training stage of our SGP model?
RQ3.5. How do the embeddings dimension and different ranking cut-offs affect the recommendation performances of the pre-trained recommenders?

In the following, we describe the 13 baselines used to evaluate the performance of SGP (Section 3.4.1), the used evaluation methodology and the corresponding experimental setup (Section 3.4.2).

3.4.1 Baselines

We compare the performance of our SGP model to classical strong non-neural baselines, as well

[2] http://cseweb.ucsd.edu/~jmcauley/datasets.html [3] https://www.yelp.com/dataset

Table 3.2: Statistics of datasets.

	Librarything	Epinions	Yelp
Users	60,243	114,738	215,471
Items	200,422	34,577	93,379
Interactions	930,053	110,671	1,506,039
Social edges	110,637	150,859	1,397,180
Interaction density(%)	0.008	0.003	0.007
Social density(%)	0.003	0.001	0.003

as existing state-of-the-art neural models:

- **MF** (Rendle et al., 2020). This is the conventional matrix factorisation model, which can be optimised by the Bayesian personalised ranking (BPR (Rendle et al., 2009)) or the BCE losses. The regularisation includes the user bias, the item bias and the global bias. Details of this baseline can be found in Equation (2.2).

- **SBPR** (Zhao et al., 2014). SBPR is a classic model, which adds the social regularisation to the matrix factorisation method.

- **UserKNN** and **ItemKNN** (Sarwar et al., 2001). Two neighbourhood-based models using collaborative user-user or item-item similarities.

- **SLIM** (Ning and Karypis, 2011). This is an effective and efficient linear model with a sparse aggregation method.

- **NCF** (He et al., 2017). The method is a CF model, which uses a generalised matrix factorisation method to generate pre-trained embeddings. A MLP module is also used in NCF to capture the nonlinear features from the interactions.

- **NGCF** (Wang et al., 2019c). NGCF is devised to employ a multi-layer GCN (see Section 2.2.1) on top of the user-item interaction graph to propagate the collaborative signal across multi-hops user-item neighbourhoods.

- **LightGCN** (He et al., 2020). Building on NGCF, LightGCN has fewer redundant neural components compared with the original NGCF model, which makes it more efficient and effective.

- **UltraGCN** (Mao et al., 2021). UltraGCN is a more efficient GNN-based recommender. It gains higher efficiency than LightGCN by skipping the message passing via a constraint loss.

- **SGL** (Wu et al., 2021). SGL leverages the self-supervised learning and graph contrastive learning methods to generate augmented views for nodes to enhance the model's robustness and accuracy. In particular, the graph contrastive learning method has been introduced in Section 2.2.4.

- **VAE-CF** (Liang et al., 2018). A state-of-the-art variational autoencoder-based collaborative filtering recommender system.
- **GraphRec** (Fan et al., 2019). This is the first GNN-based social recommendation method, which models both user-item and user-user interactions.
- **Diffnet++** (Wu et al., 2020a). This method is a graph-based deep learning recommender system, which uses the additional social links to enrich the user-item bipartite graph and improve the recommendation performance.

3.4.2 Evaluation Methodology

Following the offline evaluation methods described in Section 2.1.2.1, we use a leave-one-out evaluation strategy to split the interactions of each dataset into training, validation and testing sets. To speed up the evaluation, we adopt the sampled metrics (He et al., 2017; Rendle et al., 2020; Wang et al., 2019c), which randomly sample a small set of the non-interactive items as negative items (rather than take all the non-interactive items as negatives) of the validation and testing sets, and evaluate the metric performance on this smaller set. Here, we sample 100 negative items for each user in the testing sets for evaluation (He et al., 2017; Rendle et al., 2020). However, different from prior works (He et al., 2017; Rendle et al., 2020) that only use one oracle testing set per dataset with the sampled negative items, we construct 10 different testing sets with different sampled negative items for each dataset using different random seeds, in order to reduce the evaluation bias on some specific testing negatives (Krichene and Rendle, 2022). Hence, the reported performance of each run is based on the average of the 10 testing sets.

In order to answer **RQ3.1** and validate the hypothesis of our proposed thesis (see Section 1.2), we compare our SGP model with all baselines in terms of different metrics introduced in Section 2.1.2.2, including Normalised Discounted Cumulative Gain@10 (NDCG), Recall@10 and Mean Average Precision@10 (MAP). We also compare the SGP model with both its pre-training and fine-tuning stages to a variant where only the pre-training stage is used (called SGP (Pre-training)), so as to address **RQ3.2**.

To answer **RQ3.3**, we further examine the ability of our proposed SGP model to alleviate the *cold-start* problem introduced in Section 2.1, especially for those users who newly registered on the sites. In particular, we first compare the performances of our SGP model to the best performing baseline across different groups of users who have less than {5, 10, 15, 20} interactions, respectively. Second, to simulate the *extreme cold-start* situation when a user starts using an app that was suggested by his/her friends, we select those users who have social relations but less than five interactions. We define these users as the *extreme cold-start*[4] users and we remove all their interactions, so that the situation of newly registered users (no historical interaction) is simulated. Hence, through this defined *extreme cold-start* setup, we aim to recommend relevant items to those newly registered users solely based on their social relations.

In order to tackle **RQ3.4,** we conduct an ablation study to determine the effect of the social relations in our proposed SGP model and the Diffnet++ model. In this ablation study, we randomly drop {20%,40%,60%,80%} of social relations from both models and measure the resulting recommendation performance across the three used datasets, in order to determine if the performance improvements are indeed gained from the social-aware pre-training. Finally, to answer **RQ3.5**, we provide a detailed analysis on the largest dataset (i.e., Yelp) to evaluate the performances of SGP and LightGCN on different embedding dimensions and different cut-offs for the recommended items. In addition, in order to directly observe the effect of social relations in the latent space, we use the t-distributed stochastic neighbour embedding (t-SNE) technique (Van der Maaten and Hinton, 2008) to visualise the final embeddings obtained by our SGP model, in comparison to the embeddings obtained by a classic MF model.

All models are implemented using the Beta-RecSys open source framework (Meng et al., 2020). We use the Adam (Kingma and Ba, 2015) optimiser for all the neural network models' optimisations. To tune all hyper-parameters, we apply a grid search on the validation set, where the learning rate is tuned in $\{10^{-2}, 10^{-3}, 10^{-4}\}$; the latent dimension in $\{32, 64, 128\}$ and the L_2 normalisation in $\{10^{-2}, ..., 10^{-5}\}$. The node dropout technique is used in the NGCF, LightGCN, UltraGCN and GraphRec models as well as our proposed SGP model. The dropout ratios vary amongst $\{0.3, 0.4, ..., 0.8\}$ as suggested in (van den Berg et al., 2017). To control how many Gaussian distributions are extracted from the pre-trained embeddings, we vary the number of pre-defined multivariate Gaussian distributions k in Equation (3.5) in $\{2, 4, 6, 8, 10\}$. Note that due to the limit of the latent dimension, further increases in the k value might result in less data extracted from each pre-trained embedding. We conduct the paired t-test with the Holm-Bonferroni correction to examine if the performance difference between a baseline and SGP is significant. For each k value, we run our SGP model for 50 times with different random seeds and we plot the results on the three datasets as a box plot, where we illustrate not only the mean values but also the variance across different random seeds. For a fair comparison with (He et al., 2020, 2017; Wang et al., 2019c; Wu et al., 2020a), we set the number of neural network layers of the models including NCF, NGCF, Diffnet++, LightGCN, UltraGCN, SGL , GraphRec and SGP to three. For the non-neural models, namely SBPR, MF, UserKNN, ItemKNN and SLIM, we tune them within the same range of learning rates and L_2 normalisations used for the neural baselines, while for the rest of parameters we follow the same implementation details as suggested in (Dacrema et al., 2019).

[4] We sampled users with less than 5 interactions for the simulation because at least 3 interactions are needed for the train/valid/test set, and in order to keep enough users in the evaluation pool.

Table 3.3: Performances of SGP and other baselines on the three used datasets. All metrics are computed at rank cutoff 10. The best and second best performances are highlighted in boldface and underlined, respectively; * denotes a significant difference between the performance of SGP and that of the baselines according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$. The 'Social' column indicates whether a model uses social relations or not.

	Social	Epinions			Librarything			Yelp		
		NDCG	Recall	MAP	NDCG	Recall	MAP	NDCG	Recall	MAP
NCF	✗	0.0819*	0.1662*	0.0585*	0.3132*	0.4971*	0.2304*	0.2504*	0.4109*	0.1908*
NGCF	✗	0.0816*	0.1668*	0.0589*	0.2974*	0.4894*	0.2498*	0.2378*	0.3904*	0.1794*
LightGCN	✗	0.0830*	0.1723*	0.0615*	0.3310*	0.5081*	0.2484*	0.2735*	0.4304*	0.2194*
UltraGCN	✗	0.0825*	0.1700*	0.0603*	0.3313*	0.5083*	0.2480*	0.2598*	0.4002*	0.2011*
SGL	✗	0.0831*	0.1720*	0.0610*	0.3216*	0.4982*	0.2417*	0.2632*	0.4100*	0.2098*
VAE-CF	✗	0.0710*	0.1424*	0.0475*	0.3003*	0.4934*	0.2302*	0.2100*	0.3715*	0.1639*
Diffnet++	✓	0.0819*	0.1678*	0.0549*	0.3011*	0.4873*	0.2259*	0.2589*	0.4184*	0.1988*
GraphRec	✓	0.0810*	0.1661*	0.0527*	0.2997*	0.4807*	0.2248*	0.2478*	0.4097*	0.1901*
SBPR	✓	0.0791*	0.1571*	0.0509*	0.2997*	0.4931*	0.2300*	0.2398*	0.3937*	0.1808*
MF	✗	0.0720*	0.1481*	0.0484*	0.2903*	0.4893*	0.2291*	0.2011*	0.3348*	0.1698*
UserKNN	✗	0.0752*	0.1678*	0.0497*	0.3123*	0.4987*	0.2345*	0.2297*	0.3797*	0.1758*
ItemKNN	✗	0.0743*	0.1667*	0.0486*	0.2977*	0.4872*	0.2139*	0.2238*	0.3709*	0.1712*
SLIM	✗	0.0719*	0.1522*	0.0497*	0.2918*	0.4821*	0.2298*	0.2098*	0.3407*	0.1766*
SGP (Pre-training)	✓	0.0725	0.1498	0.0497	0.2953	0.4913	0.2284	0.2201	0.3897	0.1798
SGP	✓	**0.0876**	**0.1794**	**0.0657**	**0.3569**	**0.5431**	**0.2647**	**0.2972**	**0.4631**	**0.2347**
%Improv.		5.5	4.1	6.8	7.8	6.9	6.0	8.7	7.6	7.0

3.5 Results Analysis

In this section, we report the experimental results and answer our five research questions in turn.

3.5.1 RQ3.1: Pre-trained Recommendation Performances

In order to answer **RQ3.1**, we use Table 3.3 to report the overall performance of our SGP model in comparison to 13 other baselines and the pre-training stage (the first stage only of the SGP model) in terms of 3 different metrics, namely NDCG, Recall and MAP. Comparing the performance of SGP (Pre-training) with other baselines, we can conclude that the pre-training stage itself cannot outperform all baselines. However, through the information distillation stage when we use randomly initialised embeddings concatenated with Multivariate Gaussian distributions extracted from the pre-trained embeddings, our SGP model achieves the best performance, constantly and significantly (according to the paired t-test with Holm-Bonferroni correction with $p < 0.01$) outperforming all other baselines in terms of all metrics on three used datasets. These results demonstrate that solely employing the GNN model with the available social relations is not sufficient to enhance the recommendation performance. This is likely because the social information should not be considered equally to the interaction information, since the interaction information makes the actual ground truth when inferring the users' main preferences and next items of interest. By reusing these pre-trained embeddings concatenated with randomly initialised embeddings, our SGP model can markedly and significantly enhance the recommendation performance. It is of note that the performances of all the evaluated models on the Epinions dataset are lower than on the Librarything and Yelp datasets. However, these performances are in line with those reported in the literature (e.g., NDCG@10 $\approx$ 0.3 on the Librarything dataset (Mauro et al., 2019; Palumbo et al., 2018; Valcarce et al., 2019); NDCG@10 $<$ 0.1 on the Epinions dataset (Abdollahpouri et al., 2019)). These differences may be explained by the differing densities of user-item interactions in the used datasets (see Table 3.2). In addition, by comparing other graph-based models, we observe that those more recent models such as SGL and UltraGCN do sometimes outperform the LightGCN model. However, LightGCN remains the second-best performing model since it has achieved second-best performances for most of the times as shown in Table 3.3. This observation is probably related to our used data split, where we use 10 different testing sets to avoid the oracle testing set. In other words, our results imply that those more recent graph-based models have not achieved robust and constant enough improvement over the LightGCN model, which demonstrates the necessity of addressing the low-efficiency issue as mentioned in our proposed thesis statement (see Section 1.2). As a consequence, we will use LightGCN as the main model to be compared with SGP in Sections 3.5.3 & 3.5.5. Overall, in answer to **RQ3.1**, we can conclude that using the GNN model to leverage the social relations and generate pre-trained embeddings can improve the recommendation performance compared with SGP (Pre-training) and 13 competitive (neural and non-neural) baselines.

3.5.2 RQ3.2: GMM Information Distillation

To address **RQ3.2**, we show a box plot of our SGP model on the 3 used datasets across different number of pre-defined multivariate Gaussian distributions, k, in terms of NDCG@10, where for each k value, the model is trained and evaluated 50 times with different random seeds. In Figure 3.2, the max and min values for each set of experiments are shown as two bars at the top and bottom of each box, respectively. The mean value of each set of experiments is shown as an orange line lying in the middle of each box. We also report the best mean for each dataset in Table 3.3 (i.e., $k = 6$ for the Epinions dataset and $k = 8$ for the Librarything and Yelp datasets). Figure 3.2 shows that our SGP model only achieves better performances when k is larger than 4, whereas for all datasets when $k = 2$ or 4, the SGP model has a lower performance than several baselines. This can be explained by the fact that the users' preferences are hard to be estimated with simple distributions. Indeed, usually the users' preferences are formed by combinations of distributions, which cannot be easily factorised with 2 to 4 factors. Therefore, a small number of Gaussian distributions is not sufficient enough to represent the users' preferences. However, we also observe a performance degradation when k is too large. This is likely because the latent dimension has a limited size (usually up to a few hundreds), while each reconstructed embedding is a sample from the multiple extracted Gaussian distributions. Therefore, when k becomes larger, elements sampled from each distribution become fewer, thereby leading to a loss in the accuracy of the representation of its intended original factor. For example, when the

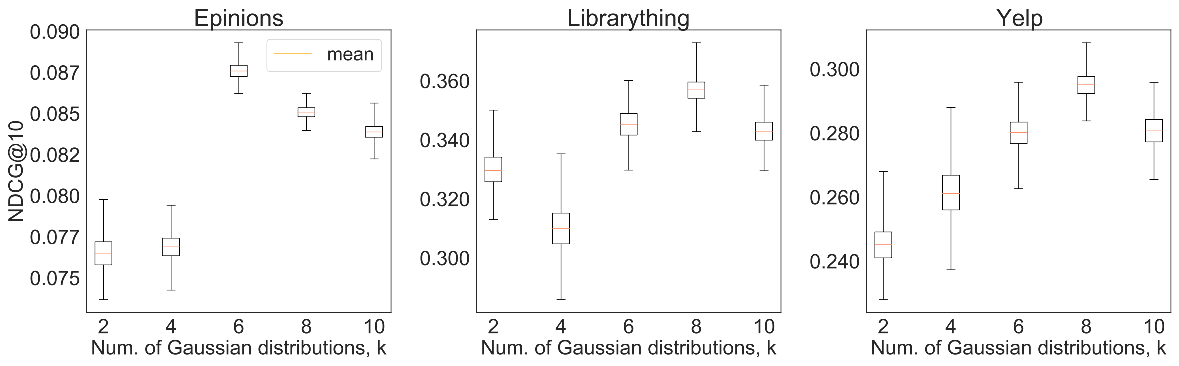


Figure 3.2: Max/Min/Mean values of NDCG@10 for SGP on 3 datasets with different number of Gaussian distributions.

Table 3.4: NDCG@10 performances of our SGP model in comparison to the LightGCN baseline across different user groups where n is the number of users' interactions; * denotes a significant difference versus LightGCN (paired t-test, p<0.01).

		NDCG@10				
Dataset	Model	n=5	n=10	n=15	n=20	n=all
Epinions	SGP	0.0763*	0.0812*	0.0857*	0.0864*	0.0876*
	LightGCN	0.0708	0.0751	0.0810	0.0818	0.0830
	%Improv.	7.77	8.12	5.80	5.62	5.50
Librarything	SGP	0.3014*	0.3354*	0.3410*	0.3554*	0.3569*
	LightGCN	0.2631	0.2821	0.3101	0.3275	0.3310
	%Improv.	14.5	15.9	9.96	8.52	7.80
Yelp	SGP	0.1987*	0.2435*	0.2669*	0.2848*	0.2972*
	LightGCN	0.1671	0.2086	0.2381	0.2590	0.2735
	%Improv.	18.9	16.7	12.1	9.97	8.71

latent dimension is 100, if $k = 10$ is applied, only 10 elements are sampled from each Gaussian distribution. Moreover, when k is larger, the performance of our SGP model is relatively stable. This demonstrates that our SGP model is effective in distilling information from the pre-trained embeddings given that enough Gaussian distributions are employed i.e., when k is sufficiently large, the model stabilises and shows less variance. In answer to **RQ2**, we can conclude that the GMM can be used to effectively distil information from the pre-trained embeddings. We also suggest preferable k values, which can be used to enhance the recommendation performance. In addition, we consolidate the hypothesis in our thesis statement by showing that heterogeneous graph representation learning can effectively encode the social relations among users.

3.5.3 RQ3.3: *Cold-start* Performances

To address **RQ3.3**, in Table 3.4, we examine the performance of our SGP model for different groups of users who have less than {5, 10, 15, 20} interactions, respectively, in comparison to the best baseline, LightGCN, in terms of NDCG@10. From the table, we note that our SGP

Table 3.5: Performances of SGP (Pre-training) on the *extreme cold-start* users in comparison with the random and popularity-based baselines. In the table, SGP is the only approach where interactions are used. * and ↑ denote significant differences compared to the random and popularity baselines, respectively (paired t-test, p<0.01).

	Epinions			Librarything			Yelp		
	NDCG	Recall	MAP	NDCG	Recall	MAP	NDCG	Recall	MAP
Random	0.0676	0.1334	0.0403	0.0998	0.1938	0.0707	0.0887	0.1806	0.0615
Popularity	0.0712	0.1448	0.0423	0.1693	0.3011	0.1082	0.1937	0.3219	0.1287
SGP	$0.0870^{*\uparrow}$	$0.1691^{*\uparrow}$	$0.0589^{*\uparrow}$	$0.3324^{*\uparrow}$	$0.5134^{*\uparrow}$	$0.2958^{*\uparrow}$	$0.2972^{*\uparrow}$	$0.4631^{*\uparrow}$	$0.2347^{*\uparrow}$
SGP (Pre-training)	$0.0728^{*\uparrow}$	$0.1583^{*\uparrow}$	$0.0450^{*\uparrow}$	$0.2029^{*\uparrow}$	$0.3360^{*\uparrow}$	$0.1382^{*\uparrow}$	$0.2279^{*\uparrow}$	$0.3469^{*\uparrow}$	$0.1488^{*\uparrow}$

model overall significantly outperforms LighGCN according to the paired t-test with $p < 0.01$, while users with less than 10 interactions particularly benefit from our model compared with the other groups of users. Overall, it is reasonable to observe that *cold-start* users benefit more from our model because when their interaction information is too sparse, incorporating more social information will likely enable the SGP model to predict their possible unknown preferences. However, users with sufficient interactions tend to have their preferences accurately captured by the recommender systems, therefore adding more social relations may not be beneficial for them. Indeed, from Table 3.4, we observe that there is a clear decrease in the reported percentage improvement when we consider the group of users who have less than 10 interactions in comparison to those users who have more than 15 interactions.

Table 3.5 shows a comparison of our SGP model with a random recommender and a popularity-based recommender for the *extreme cold-start* users case. The random and popularity-based recommenders are two commonly used baselines (Noia et al., 2016) when no interaction data is available. Here, we aim to simulate the situation when users register to an App or a Web service following the suggestions of their friends. In this case, the model only knows about the users' friends while it does not have access to the historical interactions. Instead of making random recommendations or only recommending popular items, our SGP model generates embeddings by constructing multivariate Gaussian distributions by evenly sampling elements from their friends' embeddings, which is also produced by the pre-training stage of our SGP model. By comparing our proposed SGP model with its first pre-training stage only (denoted by SGP (Pre-training)) with both baselines and the full SGP model for the *extreme cold-start users*, we find that SGP (Pre-training) significantly outperforms both the random and the popularity-based recommenders. On the other hand, it is reasonable and natural that when no interactions are observed and no training is conducted, SGP (Pre-training) is far worse than the full SGP model. However, SGP (Pre-training) significantly outperforms both the random and the popularity-based recommenders on the Librarything and Yelp datasets and is comparable to the results of SGP on the Epinions dataset. Overall, in answer to **RQ3.3**, we can conclude that our proposed method SGP is effective at tackling the *cold-start* problem and is particularly useful in alleviating the practical extreme cold-start issue. Furthermore, we validate the hypothesis in our proposed thesis statement that graph representation learning can alleviate the *cold-start* problem while ensuring effective recommendations for *regular* users.

3.5.4 RQ3.4: Impact of Social Relations

In order to determine the effect of social relations and to answer **RQ3.4**, we conduct an ablation study where we randomly dropout different proportions of social relations. Figure 3.3 shows how the performance of our SGP model and that of the Diffnet++ model are affected when different proportions of social relations are randomly masked out. From this figure, we can clearly observe a trend that the more social relations are masked during the pre-training, the more the recommendation performances of SGP and Diffnet++ are degraded across three datasets. This trend reveals that the social relations do indeed help the SGP model to achieve a better pre-training thereby enhancing the final recommendation performance. However, we also observe some variance in the performance on the Epinions dataset, compared with the consistent decline of performance on the Librarything and Yelp datasets. This is because the raw data of the Epinions dataset provides bidirectional social relations i.e., both the 'trust' and 'trustedby' relations are given. Since our current SGP model cannot distinguish between these bidirectional relations, for the sake of simplicity, we unify these two types of relations as one unidirectional social network to fit our implementation. Although unifying the bidirectional relations does bring an overall performance improvement to SGP over other baselines, this unifying method itself is not optimal and can possibly induce noise, because the social influences are not bidirectionally equal. Therefore, from our conducted ablation study, in answer to **RQ3.4**, we can conclude that using the social relations on the pre-training stage can help enhance the recommendation performance of our SGP model. Furthermore, we postulate that the performance can be further enhanced by enabling our current SGP model to distinguish among bidirectional social relations. In particular, through this ablation study, we gain more confidence in our proposed statement because we further justify that the improvements in effectiveness brought by our SGP model come from incorporating social relations using heterogeneous graph representation learning.

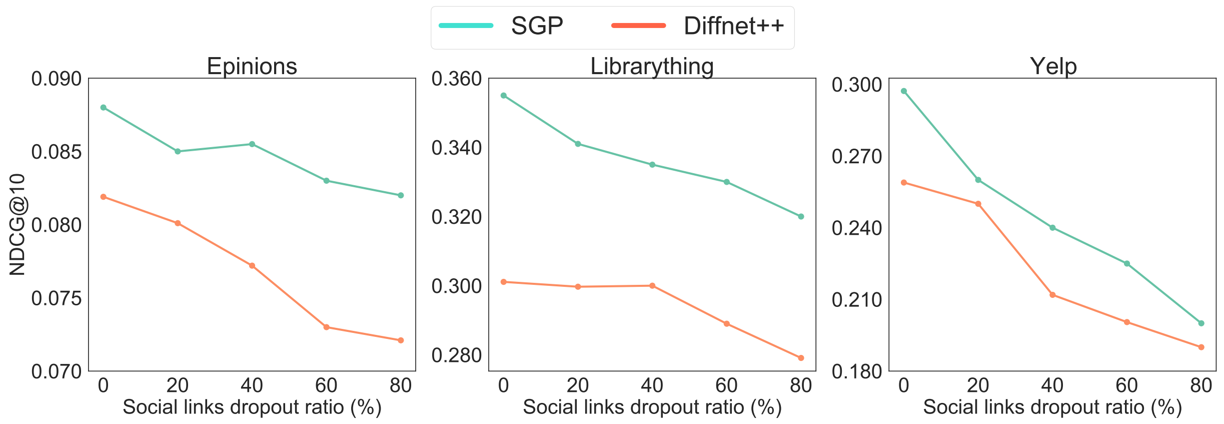


Figure 3.3: An ablation study of performances of SGP and Diffnet++ (different proportions of social relations are masked out).

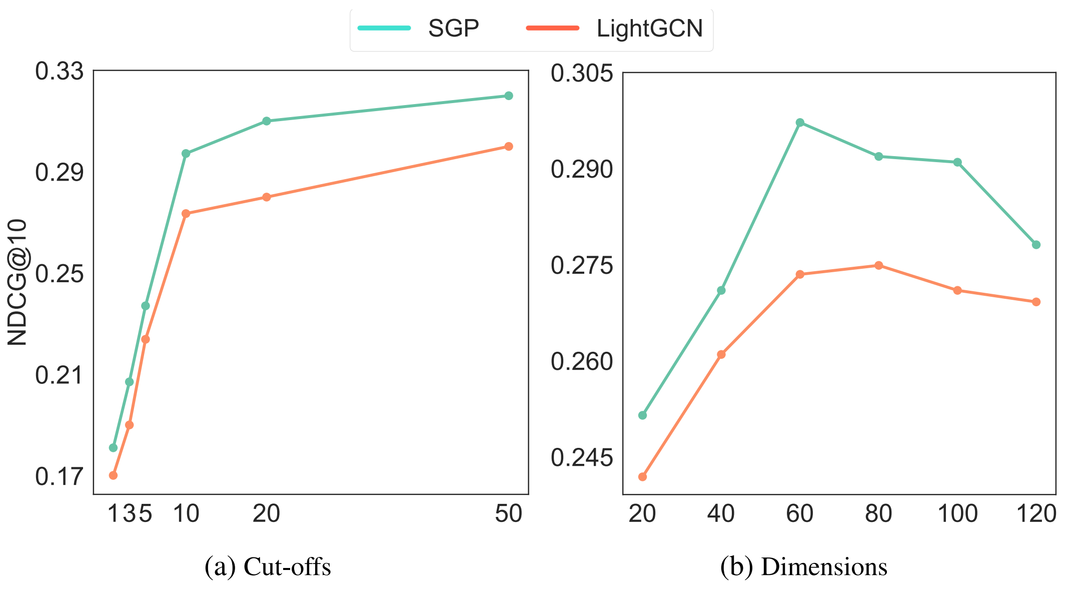


Figure 3.4: The performances comparison between SGP and LightGCN over different dimensions and different cut-offs on the Yelp dataset.

3.5.5 RQ3.5: Hyperparameter Analysis

In this section, we aim to answer **RQ3.5** by examining how the performance of SGP and that of the second-best baseline LightGCN are affected by different recommendation cut-offs and embedding dimensions. First, we plot when the different number of items are recommended to users in Figure 3.4a. From this figure, we can observe that our SGP model consistently outperforms LightGCN across different cut-offs. Specifically, SGP mainly surpasses LightGCN for larger cut-offs (i.e., when cut-offs $\geq$ 10). This is due to the fact that we regard social relations as side information, and they are only leveraged during the pre-training stage. Hence, those items that are easy to be predicted will be preserved as top-ranked items, while social relations play an important role in obtaining more accurate tailed items. As a result, users will benefit more from our proposed SGP model when the top-ranked items are unsatisfactory for them. Figure 3.4b shows how the NDCG@10 measure is affected when different embedding dimensions are applied to SGP and LightGCN. This figure demonstrates that our SGP can bring consistent improvements over the baseline for different embedding dimensions. To conclude on **RQ3.5**, our SGP model can constantly outperform the strong baseline i.e., LightGCN when different hyperparameters are applied. In addition, we further consolidate our proposed thesis statement by showing the consistent improvements of SGP over LightGCN.

3.5.6 Embedding Visualisation

In this section, we aim to analyse how our SGP model affects the users' embeddings in the latent space, compared to the embeddings obtained from a classic MF model[5] (see Section 2.1.1.1) that does not encapsulate social relations. We visualise all users' embeddings of the Librarything dataset[6], trained by the SGP model in comparison with embeddings trained by the MF model using the t-distributed stochastic neighbour embedding (t-SNE) approach. Figure 3.5(a) shows the t-SNE for MF, while Figure 3.5(b) provides the t-SNE for SGP. In both plots, we highlight three anchor users (represented as yellow/green/red dots), along with their corresponding friends (triangles) and their target items (stars). Both the green and orange anchor users are fortunate to have their friends close to their target items, hence, these two anchor users are pulled closer to their target items, as shown in Figure 3.5(b). In contrast, in Figure 3.5 (a), these two anchor users are clustered far apart from their target items by MF, due to the fact that social relations are not considered by MF. For the red user's case, he/she has a dissimilar friend, who is located relatively far away from the target item and his/her friends. Our SGP model can still handle this case by relocating the red user to the space between this dissimilar friend and two other similar friends, thereby bringing this user closer to the target item. Through the provided three examples of users, we illustrated different situations where users might possibly benefit from our SGP model, thereby improving the recommendation performances as observed in the reported results across three datasets. Furthermore, we use this detailed case study to obtain more insights into how heterogeneous graph representation learning benefits the recommendation hence further supporting our proposed thesis statement.

3.6 Conclusions

In the proposed thesis statement, we postulated that we can use heterogeneous graph representation learning to alleviate the *cold-start* problem while ensuring enhanced recommendations for *regular* users by leveraging social relations. Therefore, in this chapter, we explored how to leverage graph representation learning to generate pre-trained embeddings using the existing social relations among users. Next, we used the Gaussian Mixture Model to carefully extract prior knowledge contained in those pre-trained embeddings for subsequent fine-tuning and recommendations. Our proposed *Social-aware Gaussian Pre-trained* (SGP) model can significantly outperform competitive baselines (according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$), as demonstrated by the extensive experiments conducted on three public datasets (see Table 3.3). Furthermore, a detailed user analysis in Section 3.5.3 showed that by incorporating the social relations, users who have less than 10 interactions especially benefit

[5] The MF model is chosen because it is also an embedding-based method and is not socially aware, and therefore can offer us a clear comparison between a social-aware model and a non-social model. [6] The Librarything dataset is a less sparse dataset with a higher or equal social density compared to the Epinions and Yelp datasets therefore, for illustration purposes, we have more users to choose from. However, note that we do nevertheless observe similar trends on the Epinions and Yelp datasets, hence for space constraints, we only visualise the Librarything dataset

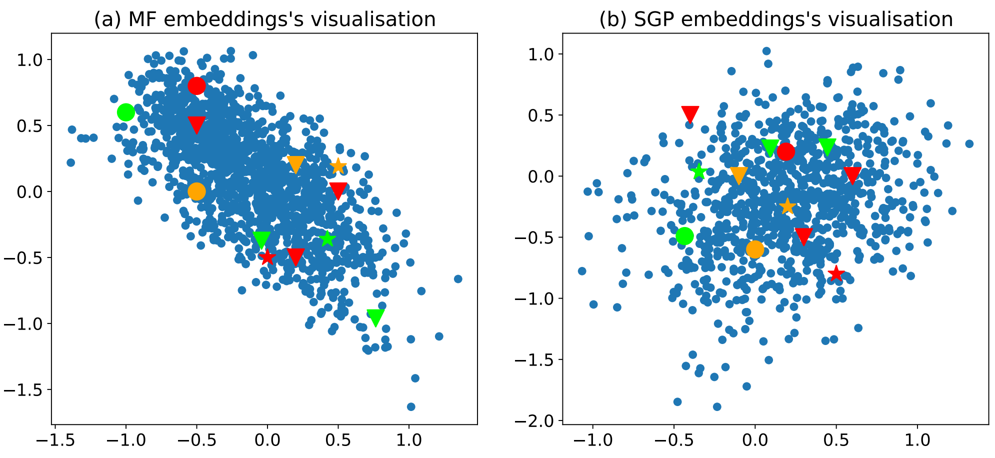


Figure 3.5: The t-SNE plot of all users' embeddings of the Librarything dataset obtained from the MF model (a) and our SGP model (b), where the red dot represents an anchor user, the red triangles are this user's friends and the red start is the target item. The similar configuration is also applied to another two users with their corresponding friends and target items, which are plotted with the colour of green and orange, respectively.

from our SGP model, suggesting that SGP can effectively alleviate the *cold-start* problem introduced in Section 1.1. Moreover, we showed that our SGP model can practically serve *extreme cold-start* users with reasonable recommendations when it only knows about the friend's preferences of these newly registered users. Finally, we used an ablation study in Section 3.5.4 to examine the effect of social relations on our proposed model and a hyperparameter analysis to study the effects of different cut-offs and embedding dimensions, followed by the visualisation of the generated embeddings to further illustrate how our proposed model could benefit recommendations.

Therefore, we can conclude that we have validated the hypothesis of our proposed thesis statement in Section 1.2, namely by capturing the user-user social relation using heterogeneous graph representation learning, a graph-based recommender system can achieve an enhanced performance and benefit *cold-start* users. The possible limitation of our SGP model is linked to the risk of the leakage of sensitive personal information. For instance, one could use a social network graph as an attack vector to expose sensitive information of public figures. Hence, our SGP model should be used with careful consideration if applied to a real application. The success of this chapter raises the question of whether we can generalise the incorporation of social relations to other side information in a graph-based recommender system. Therefore, in Chapter 4, we will explore how to incorporate multiple side information in a more generalised graph pre-training manner.

Chapter 4

Graph Pre-training for Recommendations

4.1 Introduction

In Chapter 3, we validated the hypothesis in our proposed thesis statement (Section 1.2) that by incorporating heterogeneous graph representation learning to leverage the social relations, we can alleviate the *cold-start* problem and improve the overall recommendation effectiveness. The promising performance of our proposed social-aware graph-based recommender system motivates us to further progress their development. Indeed, the effectiveness of social relations in graph-based recommender system poses the follow-up question: *can other side information be used to improve the effectiveness of graph-based recommender systems?* The intuition that social relations can benefit graph-based recommender systems is based on the assumption introduced in Section 1.1 that socially related users might have similar interaction behaviours. A difference between the social relations and other side information of users and items is that social relations can directly link two users, while other side information is usually regarded as the labels or attributes of users/items. Hence, in the scenario of graph-based recommender systems, the social relations can be directly used while other side information may need pre-processing. Although other side information cannot directly link two users, we can still use different types of side information to categorise users into different clusters/groups; hence we assume that users from the same cluster behave similarly. Similarly, we can also categorise different types of items into different groups following how we use the side information to categorise users. Therefore, it is promising to investigate how to leverage multiple types of side information for graph-based recommender systems.

As mentioned in Section 1.5, in this chapter we tackle a more generalised problem (leveraging multiple types of side information) than only leveraging social relations as in Chapter 3 (social relation only). Recall that we have discussed (Section 2.2.2) the limitation of the classic bipartite graph, which can only be extended with two additional relations besides the user-item interactions. Therefore, a simple additive solution is not applicable if we want to leverage more than two types of side information. To tackle this problem, in this chapter, we introduce a novel pre-training scheme to leverage multiple side information in this chapter.

As mentioned in Section 2.1, the goal of recommender systems is to assist users in filtering out non-relevant information and selecting a personalised set of interesting items to maximise the users' satisfaction. Modern recommendation models achieve this goal by learning representation vectors (i.e., embeddings) of the two entities (i.e., users and items) that capture the users' interests and items' attractiveness (Zhang et al., 2019b, 2017), so that the learned embeddings can be used to accurately predict which items a user might choose in the future, e.g., by computing the dot product of the users' and items' embeddings (see Equation (2.1)). Typically, recommendation models are collaborative as described in Section 2.1.1.1, learning the users' interests and items' attractiveness through users' ratings, clicking or other side information. For example, the NGCF model leverages the high-order connectivity between users and items (i.e., the interaction signal in the user-item interactive graph as defined in Section 2.2) to enhance the recommendation performance. However, user-item interactions are typically sparse (Rendle et al., 2009). Therefore, a number of side information-aware recommendation models (Chen and de Rijke, 2018; Liu et al., 2019b; Vasile et al., 2016) have been designed to alleviate the sparsity issue by integrating the rich side information of users and items such as the users' age groups and the items' textual descriptions. Such side information about entities can be used to learn more relations between users and items to further enhance the recommendation performance. For instance, movies with the same features (e.g., same genres and actors) may attract similar users, and such feature relations between movies are a type of knowledge within the side information of movies.

To leverage the side information associated with users and items, many approaches have been proposed, most of which follow the conventional *integration scheme*, which encodes the side information simultaneously with the training of user-item interactions (Chen and de Rijke, 2018; Ning and Karypis, 2012; Park et al., 2013). These integration-based approaches normally optimise a loss function consisting of two components, i.e., the recommendation loss and one (or even more) additional side information-aware loss(es), where a hyperparameter is usually used to control the importance of each loss component (Chen and de Rijke, 2018; Liu et al., 2019b; Wang et al., 2019b; Zhao and Guo, 2017). It is often difficult to find one single adequate solution to optimise all of the loss components since different tasks might conflict with each others (Lin et al., 2019). For example, if two users share an interest in the same type of side information but without having a similar purchase behaviour, the two loss components may have different optimisation directions, making the combined loss hard to optimise and requiring a trade-off between the two objectives. However, it requires tremendous efforts to tune one (or more) hyperparameter(s) to find a good trade-off solution between the two objectives (Liu et al., 2019b; Zhao and Guo, 2017), and they may fall into a scenario where the two loss components strongly conflict during training, resulting in reduced effectiveness (Lin et al., 2019). Moreover, entity side information is typically heterogeneous, meaning that it may consist of many different

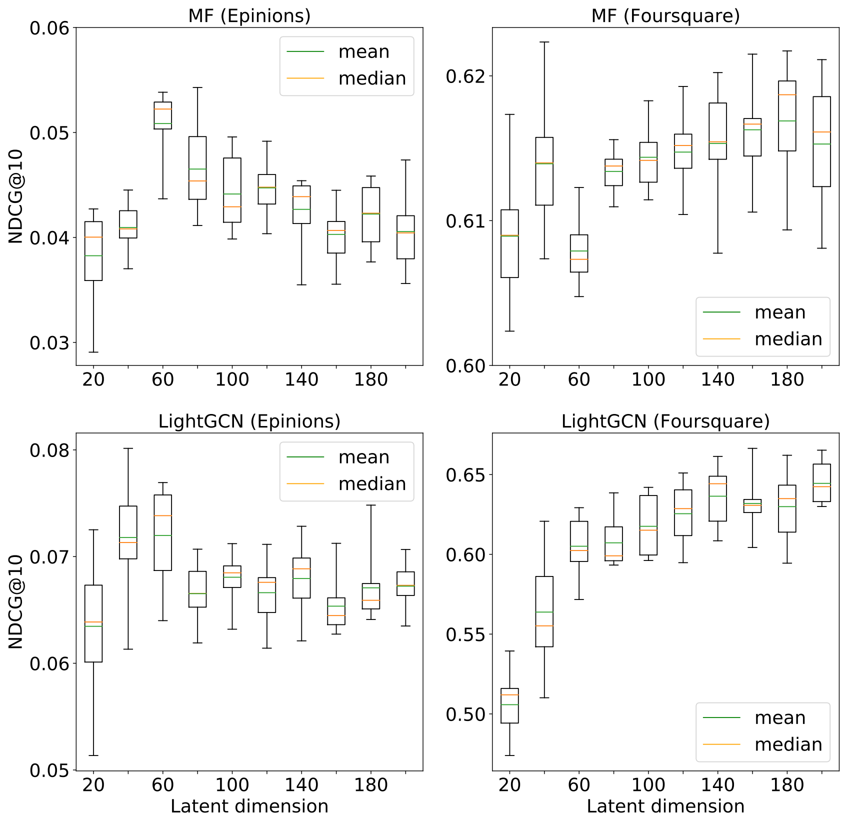


Figure 4.1: Box-and-whisker diagrams for the NDCG performances of the MF (Rendle et al., 2020) and LightGCN (He et al., 2020) models on the Epinions and Foursquare datasets.

feature types (e.g., age, gender and education level) and be represented by different data types (e.g., binary-values, categorical-values or real-values). These types of side information usually play different roles when contributing to the generation of each entity's representation and need to be jointly modelled to capture their heterogeneous semantics. However, existing side information-aware models usually use one or more fixed hyperparameter(s) (Liu et al., 2020, 2019b) to control the importance of all different types of side information. Alternatively, they follow the paradigm of multi-task learning by using a constant value to balance the sum of the main loss and the side information-aware loss (Li et al., 2020a; Wang et al., 2019a), thereby leading to a lack of generalisation and/or reduced performance robustness.

Furthermore, while there have been many powerful neural network-based recommendation models as mentioned in Section 2.1 (He et al., 2020, 2017; Wang et al., 2019c), most of these models are unable to give stable recommendation results. Indeed, as we can see from Figure 4.1, with the different random initialisation of the model's parameters, high variances can be observed in the performances of both a conventional model (i.e., MF (Rendle et al., 2020) as defined by Equation (2.2) in Section 2.1.1) and the graph-based model (i.e., LightGCN (He et al., 2020)) as defined by Equation (3.4) over different embedding dimensions in both the Epinions and Foursquare datasets, demonstrating the lack of stability of these models. In this chapter, we argue that a better initialisation of the model's parameters encapsulating the entity side information could alleviate the low-robustness issue introduced in Section 1.1, and allows the underlying recommendation model to find a stable local optimal recommendation solution.

To address the aforementioned low-robustness issue and enhance the overall recommendation effectiveness, we propose a novel pre-training scheme for leveraging the side information in recommender systems, namely, we first pre-train the embeddings of entities using their side information, and then fine-tune the pre-trained embeddings using an existing recommendation model. Specifically, we explore two types of graph-structured data to capture the interdependent relationships among the entities, and propose two pre-training models using the graph pre-training technique introduced in Section 2.2.3, namely, the **Single-P** model and the **Multi-P** model. The **Single-P** model learns the entity embeddings on single-relational graphs using Graph Convolutional Networks (Kipf and Welling, 2017), while the **Multi-P** model learns entity embeddings on multi-relational graphs using Composition-based Multi-Relational Graph Convolutional Networks (Vashishth et al., 2020). With the expressive power of GNNs that recursively propagate messages and aggregate features over neighbours, our pre-training models are able to encode more relations from the side information of users and items. Note that our proposed pre-training scheme is a general framework to pre-train entity embeddings with side information in recommender systems. Once these embeddings are obtained, they can be applied to existing general representation-based recommenders to enhance their effectiveness and stability. We deploy our pre-trained embeddings into four existing representative general recommender models, i.e., MF (Koren et al., 2009), NCF(He et al., 2017), NGCF (Wang et al., 2019c) and LightGCN (He et al., 2020), to validate the effectiveness of our proposed pre-training scheme.

The contributions of this chapter can be summarised as follows[1]:

(1) We introduce a novel pre-training scheme for leveraging side information by first pre-training the entity embeddings using entity side information and then fine-tuning them using an existing recommender model.

(2) We propose two pre-training models using graph neural networks, namely, the **Single-P** and the **Multi-P** models, which learn entity embeddings based on the single-relational and the multi-relational graphs, respectively, where both types of graphs are constructed from the entity side information. Both of our models can be deployed to fine-tune and enhance existing general representation-based recommender systems.

(3) An extensive empirical evaluation of our pre-training model – through the fine-tuning of four existing recommender models on three real-world datasets – shows that our pre-training scheme can significantly enhance those four models in terms of both the recommendation performance and the model stability.

The remainder of this chapter is organised as follows. In Section 4.2, we further position our work in the literature. Section 4.3 introduces some additional notions specifically used in

[1] Our source code is available at `https://github.com/pretrain/pretrain`.

this chapter and formally defines the task. Section 4.4 describes our pre-training scheme and details the two proposed pre-training models. The experimental setup and the results of our empirical experiments are presented in Section 4.5 and Section 4.6 respectively, followed by the conclusions in Section 4.7.

4.2 Side Information for Recommender Systems

In this section, we give a brief introduction to how existing recommender systems leverage side information.

As we have discussed in Sections 2.1.1.2 and 2.1.1.3 , side information is commonly used to alleviate the *cold-start* issue in different recommendation scenarios. Indeed, before the prevalence of deep neural network-based recommendation methods, there have been many variants of the Matrix Factorisation-based methods, such as the sparse linear methods with side information (SSLIM) (Ning and Karypis, 2012) and the hierarchical Bayesian matrix factorisation method (Park et al., 2013), which adopt the *integration scheme* that incorporates the entity side information by combining the recommendation loss function with an extra side information-aware loss. Even in recent years, this line of research still pervades many works in the literature. For example, xLightFM (Jiang et al., 2021) was proposed to tackle the high memory-consumption issue of factorisation machine (Rendle, 2010) and its variants (Cheng et al., 2016; Guo et al., 2017) by using the quantization-based (Lian et al., 2020b) and neural architecture search (Pham et al., 2018) method. Moreover, the HIRE model proposed by Liu et al. (2019b) uses a weighted matrix factorisation to encode both flat and hierarchical forms of side information into the users' and items' representations, while combining the recommendation loss and two side information-aware losses. Although the HIRE model can effectively incorporate two types of features, it requires different objectives for different types of features. Another line of research examined the integration of side information using deep neural networks, such as the stacked denoising auto-encoder (Wang et al., 2015) and the marginalised denoising auto-encoder (Li et al., 2015). More recently, many recommendation models have explored using Variational auto-encoders (VAEs) (Pang et al., 2019; Wu et al., 2020b; Xie et al., 2021), which jointly encode user ratings and side information during the training, in order to overcome the (often) high-dimensionality of side information. However, most of these methods only consider one type of relation from the entity features, namely they treat all the feature columns equally, thereby ignoring the variance in the feature types' importance to the recommendation performance. Moreover, most of the existing methods (Hui et al., 2021; Jiang et al., 2021; Liu et al., 2019b; Rendle, 2010) adopt the integration scheme, which needs a trade-off between the recommendation loss function and the side information-aware loss, thereby restricting the model design and making it hard to deploy into other more effective general recommender systems. Instead, in this chapter, we propose a general scheme for pre-training entity embeddings using the entity side information, such that these embeddings can be fine-tuned by an existing representation-based recommender system.

4.3 Preliminaries

In this section, we introduce the notations of single-relational and multi-relational graphs, which are only used in this chapter.

To facilitate the description of graph neural networks, we use $\mathcal{V}$ to denote the set of nodes in a graph. If the graph contains only one type of edges, then we call it a *single-graph*, or simply a *graph*; while a graph containing multiple edge types is normally called a *multi-graph*. Differing from the basic graph defined in Section 2.2.1, a graph containing node features is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, where $\mathcal{E}$ is a tuple set with $(u, v) \in \mathcal{E}$ being an edge between nodes $u, v \in \mathcal{V}$ and $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the d-dimensional feature metric of nodes. Then we denote a multi-graph $\mathcal{G}'$ by $\mathcal{G}' = (\mathcal{V}, \mathcal{E}', \mathcal{R}, \boldsymbol{X})$, where $\mathcal{R}$ is the set of edge types, each edge $(u, v, r) \in \mathcal{E}'$ represents that the relation $r \in \mathcal{R}$ exists from node u to v.

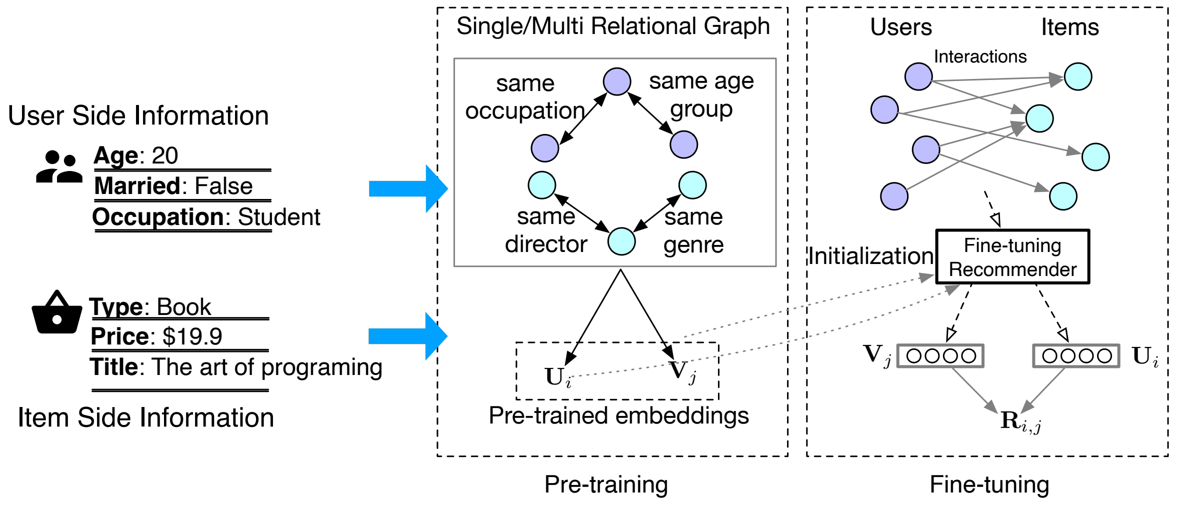


Figure 4.2: An overview of our graph neural pre-training/fine-tuning scheme. Our pre-training model constructs relational graphs based on the feature of entities, and pre-trains the embeddings of entities by using **Multi-P** or **Single-P**.

We use $\boldsymbol{F}_u$ and $\boldsymbol{F}_v$ to denote the feature matrices of the users and items, respectively. We later show that these types of features can be transformed into feature edges in multi-graphs. Then, the corresponding recommendation task is to learn a model θ:

$$\boldsymbol{F}_u, \boldsymbol{F}_v, \boldsymbol{R} \xrightarrow{\theta} \boldsymbol{U}, \boldsymbol{V}. \tag{4.1}$$

4.4 Methodology

We first introduce our pre-training scheme for recommender systems (Section 4.4.1), and detail two pre-training models, namely **Single-P** for single-graph pre-training (Section 4.4.2) and **Multi-P** for multi-graph pre-training (Section 4.4.3). We then describe the fine-tuning process using the existing recommender models (Section 4.4.6) and further elaborate on connections between our scheme and the existing works in Section 4.4.7.

4.4.1 The Pre-training Scheme for Recommendation

In this chapter, we argue that a suitable initialisation of the entity embeddings encapsulating the entity side information is critical to help recommenders learn a stable and enhanced local optimal solution for the existing recommender systems. We propose to learn such an initialisation of entity embeddings by exploiting knowledge from the entity side information. To this end, we propose a general pre-training scheme for leveraging the entity side information using graph neural networks. The overall scheme is illustrated in Figure 4.2. Our pre-training scheme consists of two processes: *pre-training* and *fine-tuning*. During *pre-training*, a graph neural network is used to learn an initialisation of the entity embeddings based on both the side information of users and items (i.e., $\boldsymbol{F}_u$ and $\boldsymbol{F}_v$) and the feedback matrix $\boldsymbol{R}$. On the other hand, in the *fine-tuning* process, an existing recommendation model leverages the pre-trained embeddings as an embedding initialisation and fine-tunes these embeddings by using the feedback matrix $\boldsymbol{R}$ only.

In order to learn the user-item preferences from the interactions between users and items, many recent models, such as NGCF (Wang et al., 2019c) and LightGCN (He et al., 2020), have explored encoding the collaborative signals from the graph-structure interactions, showing promising performances. However, these methods only investigate the interactions between users and items, ignoring the heterogeneous relations among multiple types of side information. To capture such heterogeneous relations, we are motivated by the graph pre-training method (see Section 2.2.3) to generate the pre-trained entity representations using the side information of both the users and items, so that the independent knowledge of each type of entities can be captured from the entity relations. Hence, extracting the relations from the entity side information is a crucial step for the pre-training process, since it determines how much information we can obtain from the entity features and how important such information can help to improve a recommendation model. To extract user-user and item-item relationships from their entity features (i.e., the users' and items' respective features), we propose to build two different types of feature graphs, i.e., the single-relational graphs and the multi-relational graphs, by constructing both the *homogeneous* and *heterogeneous* links between the entity pairs, respectively. We then explore how entity relations from various entity features affect the recommendation performance.

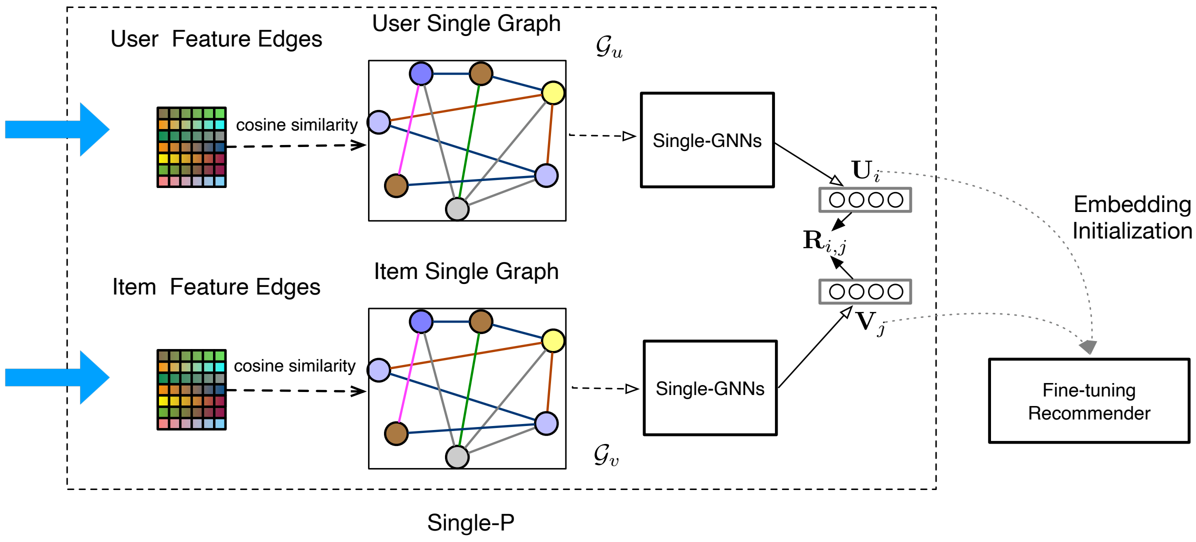


Figure 4.3: An illustration describing the input features, single-relational graphs and output entity embeddings of **Single-P**.

4.4.2 Pre-training on Single-Graphs

We construct two single-graphs, i.e., $\mathcal{G}_u$ and $\mathcal{G}_v$, from the features of users and items respectively, by considering the similarities between users and items as the homogeneous edges, and calculate the edge weights using the cosine similarities between each pair of entities. For example, we construct a *user single-graph* $\mathcal{G}_u = (\mathcal{V}_u, \mathcal{E}_u, \boldsymbol{A}_u, \boldsymbol{X}_u)$ by taking all the users as the set of nodes $\mathcal{V}_u$ and all the user pairs as the set of edges $\mathcal{E}_u$ in the graph. For each $(i, j) \in \mathcal{E}_u$, we calculate the edge weight $\boldsymbol{A}_{ij}$ using the cosine similarity of their feature vectors (i.e., $\boldsymbol{F}_{u_i}$ and $\boldsymbol{F}_{u_j}$): $\boldsymbol{A}_{ij} = \frac{\boldsymbol{F}_{u_i} \cdot \boldsymbol{F}_{u_j}}{\|\boldsymbol{F}_{u_i}\| \|\boldsymbol{F}_{u_j}\|}$. $\boldsymbol{X}_u \in \mathbb{R}^{n \times d}$ is an initial node feature matrix of the graph, the values of which are initialised from the uniform distribution $\mathcal{U}(-0.01, 0.01)$. In the following, we only describe the encoding process for the user single-graph $\mathcal{G}_u$, since the *item single-graph* $\mathcal{G}_v$ is constructed and processed in a similar fashion.

The Single-P model. To obtain the pre-trained embeddings of entities (i.e., users and items) and to exploit the potential correlation among entities based on their single-graphs, three GCN layers (Kipf and Welling, 2017) are applied to encode the entity embeddings according to their relations. The key point of GCN is to propagate the feature information through neighbourhoods of nodes in each iteration during training. The detailed framework of the **Single-P** model is illustrated in Figure 4.3. Based on the basic GNN model defined by Equation (2.20) (see Section 2.2.1), the GCN model adopts the following propagation rule:

$$\boldsymbol{H}_u^{(l)} = \sigma\left(\hat{\boldsymbol{A}}_u \boldsymbol{H}_u^{(l-1)} \boldsymbol{W}_u^{(l-1)}\right), \tag{4.2}$$

where $\hat{\boldsymbol{A}}_u = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}(\boldsymbol{A}_u + \boldsymbol{I})\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$ is the symmetric normalised adjacency matrix[2], $\boldsymbol{W}_u^{(l)}$ is the

[2] $\widetilde{\boldsymbol{D}}$ is defined as $\widetilde{\boldsymbol{D}}_{ii} = \sum_j (\boldsymbol{A}_u + \boldsymbol{I})_{ij}$, where $\boldsymbol{I}$ is the identity matrix.

weight matrix of the l^{th} layer, and $\sigma(\cdot)$ denotes an activation function (e.g., the ReLU function). $\boldsymbol{H}_u^{(l)}$ is the hidden node representation in the l^{th} layer with $\boldsymbol{H}_u^{(0)} = \boldsymbol{X}_u$. As mentioned earlier in this section, we use the uniform distribution to initialise $\boldsymbol{X}_u$, which means that $\boldsymbol{H}_u^{(0)}$ also starts with this uniform distribution. In particular, we initialise the entity embeddings using the entity side information by training over the side information encapsulated in $\boldsymbol{A}$ instead of directly incorporating an initial embedding consisting of the entity side information. Detailed methods to compute the adjacency matrix $\hat{\boldsymbol{A}}_u$ and the corresponding diagonal matrix $\hat{\boldsymbol{D}}$ are defined by Equation (2.16) and Equation (2.17) in Section 2.2.1.

To facilitate the later description of our proposed model, Equation (4.2) can also be formulated in the message passing form (Vashishth et al., 2020):

$$\boldsymbol{H}_{u_i}^{(l)} = \sigma\left(\sum_{(u_i,u_j)\in\mathcal{E}_u\cup(u_i,u_i)} \boldsymbol{W}_u^{(l-1)}\boldsymbol{H}_{u_j}^{(l-1)}\right), \tag{4.3}$$

where $\boldsymbol{W}_u^{(l-1)}$ is the layer-wise parameter and here we only consider the undirected relation and the self-loop relation. The final output embeddings of the maximum depth in the GCN layers, i.e., $\boldsymbol{U} = \boldsymbol{H}_u^{(l)}$, are the pre-training embeddings to be fed into the pre-training loss function. Similarly, the item embeddings are obtained by $\boldsymbol{V} = \boldsymbol{H}_v^{(l)}$, which is aligned with Equation (4.3).

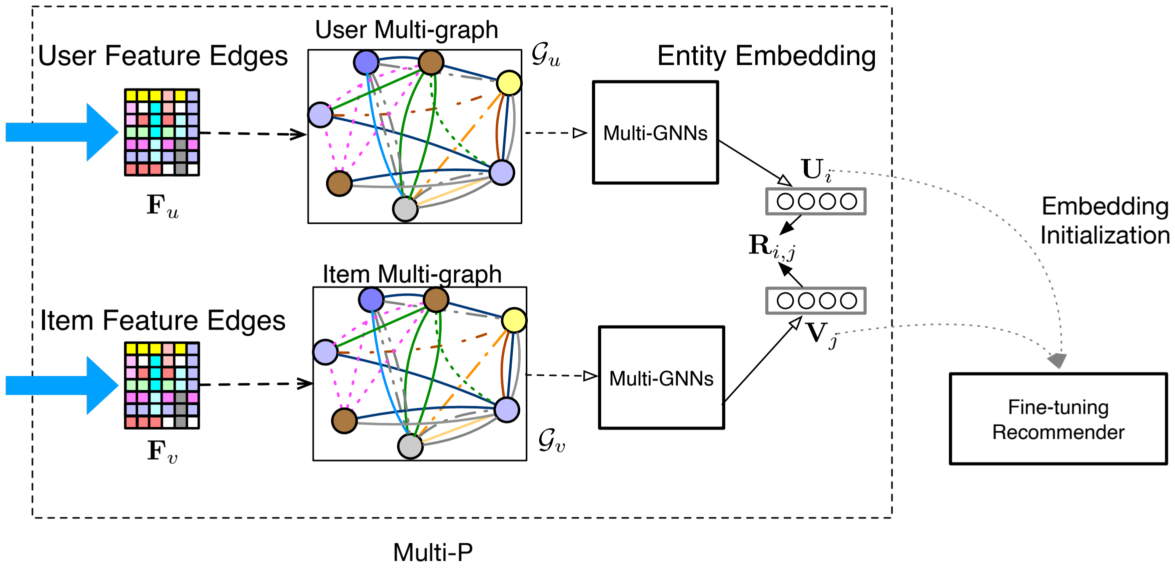


Figure 4.4: An illustration describing the input features, multi-relational graphs and output entity embeddings of **Multi-P**.

4.4.3 Pre-training on Multi-Graphs

In the existing side information-aware models (Chen and de Rijke, 2018; Li et al., 2015), different types of features associated with users and items contribute equally to the latent relationships between entities. However, in the real-world scenario, the features of users and items are typically heterogeneous, with different types of features having different usefulness for enhancing a recommender's performance. For example, users are normally associated with different types of features (e.g., age, gender and education level), which clearly characterise different aspects of the users' preferences (Liu et al., 2019b). To distinguish the importance of different feature types, we pre-train the entity embeddings through message propagation over the different feature types of the entities by using a multi-graph neural network (Vashishth et al., 2020).

Since the entity features can be real-valued, we first need to categorise such real-valued features into some groups, such that all the features of entities are sparsely categorised. Next, we can regard each feature category value as an edge type, and create an edge of this type between a pair of entities if they share the same feature value. In particular, to extract the heterogeneous relations between entities, we construct two multi-graphs for the users and items, i.e., $\mathcal{G}_u^{'}$ and $\mathcal{G}_v^{'}$, respectively. For example, to construct the *user multi-graph* $\mathcal{G}_u^{'} = (\mathcal{V}_u, \mathcal{E}_u^{'}, \mathcal{R}_u, \boldsymbol{X}_u)$, we take all the users as the set of nodes $\mathcal{V}_u$, and the feature category values as the set of relations $\mathcal{R}_u$ in the graph. For any pairs of nodes $i, j \in \mathcal{V}_u$, we create an edge if the two users share the same feature category value (e.g., if both are in the age group 25-35). $\boldsymbol{X}_u$ is set to be the random initialised user embeddings. The *item multi-graph* $\mathcal{G}_v^{'}$ is constructed in a similar fashion. Figure 4.4 illustrates the overall framework of the **Multi-P** model including how to obtain the pre-trained embeddings and how to initialise the fine-tuning recommender.

The Multi-P model. Given a multi-graph, e.g., the user multi-graph $\mathcal{G}_u^{'} = (\mathcal{V}_u, \mathcal{E}_u^{'}, \mathcal{R}_u, \boldsymbol{X}_u)$, we first extend $\mathcal{E}_u^{'}$ and $\mathcal{R}_u$ with the corresponding inverse edges and relations:

$$\begin{aligned}\hat{\mathcal{E}}_u =& \mathcal{E}_u^{'} \cup \left\{ \left(v, u, r^{-1}\right) \mid (u, v, r) \in \mathcal{E}_u^{'} \right\} \cup \{(u, u, \top) \mid u \in \mathcal{V}_u),\\ \hat{\mathcal{R}}_u =& \mathcal{R}_u \cup \mathcal{R}_u^{inv} \cup \{\top\},\end{aligned} \tag{4.4}$$

where $\mathcal{R}_u^{inv} = \{r^{-1} \mid r \in \mathcal{R}_u\}$ denotes the inverse relations (i.e., $(v, u, r^{-1}) = (u, v, r)$) and $\top$ indicates the self-loop. Then, inspired by Composition-GCN (Vashishth et al., 2020), the node embeddings are propagated through edges based on the following propagation rule:

$$\boldsymbol{H}_{u_i}^{(l)} = \sigma \left(\sum_{(u_i, u_j, r) \in \hat{\mathcal{E}}_u} \boldsymbol{W}_{\lambda(r)}^{(l-1)} \phi \left(\boldsymbol{H}_{u_j}^{(l-1)}, \boldsymbol{Z}_r^{(l-1)} \right) \right), \tag{4.5}$$

where l is layer number, $\boldsymbol{Z}_r = \sum_{k=1}^{b} \alpha_{kr} \boldsymbol{B}_k$ is the relation embedding with $\{\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_b\}$ being a set of learnable basis vectors and α_{kr} is the basis-specific learnable scalar weight. b is a hyperparameter corresponding to the number of basis vectors. $\phi(\cdot)$ is the composition operator defined as: $\phi(\boldsymbol{e}_s, \boldsymbol{e}_r) = \boldsymbol{e}_s - \boldsymbol{e}_r$, which is inspired by the TransE model (Bordes et al., 2013). $\boldsymbol{W}_{\lambda(r)}^{(l-1)} \in \mathbb{R}^{d_1 \times d_0}$ is a relation-type specific parameter, where $\boldsymbol{W}_{\lambda(r)}$ are given below:

$$\boldsymbol{W}_{\lambda(r)} = \begin{cases} \boldsymbol{W}_O, & r \in \mathcal{R}_u \\ \boldsymbol{W}_I, & r \in \mathcal{R}_u^{inv} \\ \boldsymbol{W}_S, & r = \top \quad (\text{self} - \text{loop}) \end{cases} \tag{4.6}$$

where $\boldsymbol{W}_O$, $\boldsymbol{W}_I$ and $\boldsymbol{W}_S$ are all trainable weights.

Then the output embeddings of the final layer are taken as pre-training embeddings (i.e., $\boldsymbol{U} = \boldsymbol{H}_u^{(l)}$). In the next section, we adapt the Binary Cross-Entropy (BCE) loss (see Section 2.2.1), which is defined in in Equation (2.21) for our pre-training task. The item embeddings are obtained similarly to Equation (4.5).

4.4.4 Pre-training Loss

For most graph neural networks, the embeddings can be learned by the reconstruction of the graph structure or the labels of the corresponding entities (Kipf and Welling, 2017). Therefore, for both our Single-P and Multi-P models, we pre-train the user and item embeddings with a rating/interaction-based loss, in order to encapsulate the potential information between entities for recommendation. Specifically, with the embeddings of both users and items learned from the multi-relational or single-relation graphs, we construct our pre-training loss using the Binary Cross-Entropy (BCE) loss function introduced in Section 2.2.1:

$$\mathcal{L}_{PT} = -\sum_{\boldsymbol{R}_{i,j} \in \boldsymbol{R}} \boldsymbol{R}_{i,j} \cdot \log\left(\hat{\boldsymbol{R}}_{i,j}\right) + (1 - \boldsymbol{R}_{i,j}) \cdot \log\left(1 - \hat{\boldsymbol{R}}_{i,j}\right) + \eta \left\|\boldsymbol{\Theta}\right\|^2, \tag{4.7}$$

where $\hat{\boldsymbol{R}}_{i,j}$ is the predicted score calculated by the embedding dot product $\hat{\boldsymbol{R}}_{i,j} = \boldsymbol{U}_i^{\text{tr}} \cdot \boldsymbol{V}_j$, $\boldsymbol{\Theta}$ denotes all parameter embeddings and η denotes the regularisation weight. Two explicit entity biases are also used for calculating the scores, following (Rendle et al., 2020).

4.4.5 Complexity Analysis

The complexity of our pre-training scheme highly depends on the complexity of the underlying graph neural models. For example, our proposed **Single-P** method holds a complexity of $O\left(ld|\mathcal{E}| + ld^2|\mathcal{V}|\right)$ (Kipf and Welling, 2017), which is the same as the GCN model. The complexity of our **Multi-P** model is $O\left((ld^2 + bd + b|\mathcal{R}|)|\mathcal{E}|\right)$, which is similar to the complexity of the Composition-GCN model (Vashishth et al., 2020), where l denotes the number of layers, b is the number of learnable basis vectors, d is the embedding dimension, and $|\mathcal{R}|$ is the number of relation types. The number of edges $|\mathcal{E}|$ in the entity graphs (multi-graphs) typically accounts for the largest complexity in the **Multi-P** model, and if the entities contain dense features, the number of edges $|\mathcal{E}|$ could be much larger than the number of interactions (e.g., the generated user multi-relational feature graph of the MovieLens-1M dataset contains around 2.7 million edges). Some variants of these neural models, such as FastGCN (Chen et al., 2018) and ClusterGCN (Chiang et al., 2019), might enhance the efficiency of our pre-training scheme but at the possible cost of reducing effectiveness. Once the entity embeddings are pre-trained, they can be reused and fine-tuned by many existing recommenders to enhance their effectiveness and stability.

4.4.6 Fine-tuning with Existing Recommenders

Most of the modern recommenders are trained based on gradient-based optimisation methods, which usually obtain locally-optimal solutions. Due to the non-convexity of their objective functions, parameter initialisation plays an important role for the convergence and performances of these recommendation models (Ebesu et al., 2018; He et al., 2017). Most of the existing recommendation models initialise their embeddings using a uniform distribution (Wan et al., 2018), a normal distribution (Rendle et al., 2020) or the Xavier uniform distribution (He et al., 2020; Wang et al., 2019c). Due to the randomness of the generated embeddings and the lack of prior knowledge, these models often fall into some poor locally-optimal solutions, resulting in high instabilities, as illustrated in Figure 4.1.

To address this issue, we propose to initialise the entity embeddings of an existing recommendation model from the output embeddings of our one of pre-training models, then we further fine-tune these embeddings with the recommendation model's own optimiser. Specifically, we first pre-train both the embeddings of users and items by our proposed **Multi-P** or **Single-P** models until convergence, then feed these pre-trained embeddings into an existing recommendation model as the parameter initialisation to train the recommendation model with the interactions/ratings only. The training frameworks of the pre-training and fine-tuning processes are summarised in Algorithm 3 and Algorithm 4, respectively. During the fine-tuning stage, the training objective is chosen depending on the underlying base model. For example, in this chapter, we integrate our pre-trained embeddings into four recommender systems that have been described in Section 3.4.1, i.e., MF, NCF NGCF and LightGCN. Two training objective functions are used in these models, i.e., the BCE loss and the Bayesian Personalised Ranking (BPR) loss. Specifically, the MF, NGCF and LightGCN models optimise the pairwise BPR loss (see Section 2.1.1.1), which is formulated as follows:

$$\mathcal{L}_{FT} = -\sum \ln \sigma(\hat{\boldsymbol{R}}_{i,j} - \hat{\boldsymbol{R}}_{i,z}) + \eta \left\|\boldsymbol{\Theta}\right\|^2, \tag{4.8}$$

while the NCF model is trained based on the BCE loss:

$$\mathcal{L}_{FT} = -\sum \boldsymbol{R}_{i,j} \cdot \log\left(\hat{\boldsymbol{R}}_{i,j}\right) + (1 - \boldsymbol{R}_{i,j}) \cdot \log\left(1 - \hat{\boldsymbol{R}}_{i,j}\right) + \eta \left\|\boldsymbol{\Theta}\right\|^2, \tag{4.9}$$

where $\hat{\boldsymbol{R}}_{i,j}$ denotes the predicted scores for the observed interactions, and $\hat{\boldsymbol{R}}_{i,z}$ denotes the predicted scores for the unobserved interactions.

Algorithm 3: The pre-training process

```
Input: Rating matrix $\boldsymbol{R}$; Feature matrices $\boldsymbol{F}_u$ and $\boldsymbol{F}_v$; Multi-graphs $\mathcal{G}_u^{'}$ and $\mathcal{G}_v^{'}$ or
  Single-graphs $\mathcal{G}_u$ and $\mathcal{G}_v$.
Output: Pre-trained embeddings $\hat{\boldsymbol{U}}$, $\hat{\boldsymbol{V}}$.
Initialise embeddings $\boldsymbol{U}$, $\boldsymbol{V}$ and other learnable parameters $\Theta$;
while not early-stopped do
    $\mathcal{L}_{PT} = 0$;
    for each training instance in $\mathcal{G}_u$ and $\mathcal{G}_v$ (or $\mathcal{G}_u^{'}$ and $\mathcal{G}_v^{'}$) do
        Propagate information according to Equation (4.5);
    end
    for each training instance in $\boldsymbol{R}$ do
        Compute epoch loss $\nabla\mathcal{L}$ according to Equation (4.7);
    end
    $\mathcal{L}_{PT} \leftarrow \mathcal{L}_{PT} + \nabla\mathcal{L}$;
    Update $\Theta$, $\boldsymbol{U}$, $\boldsymbol{V}$;
end
```

Algorithm 4: The fine-tuning process

```
Input: Rating matrix $\boldsymbol{R}$; Pre-trained embeddings $\hat{\boldsymbol{U}}$, $\hat{\boldsymbol{V}}$; A general recommender $q$.
Output: The recommended list $\mathcal{S}$ for each user.
Inherit $\hat{\boldsymbol{U}}$ $\hat{\boldsymbol{V}}$ to initialise $q$;
Initialise other learnable parameters $\hat{\Theta}$;
while not early-stopped do
    $\mathcal{L}_{FT} = 0$;
    for each training instance in $\boldsymbol{R}$ do
        Compute epoch loss $\nabla\mathcal{L}$ according to Equation (4.8) or Equation (4.9);
    end
    $\mathcal{L}_{FT} \leftarrow \mathcal{L}_{FT} + \nabla\mathcal{L}$;
    Update $\hat{\Theta}$, $\hat{\boldsymbol{U}}$, $\hat{\boldsymbol{V}}$;
end
Do recommendation to find the recommended list $\mathcal{S}$ based on $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$;
```

4.4.7 Discussion

The idea of pre-training embeddings for recommender systems has already been investigated in the literature. For example, to avoid saddle points and poorly performing local minima, both NCF (He et al., 2017) and CMN (Ebesu et al., 2018) apply the Generalised Matrix Factorisation (GMF) as a pre-training model to initialise the embedding weights of users and items. For both cases, the applied GMF function is a generalised version of the MF model defined in Section 2.1.1.1 (Equation (2.2)). In particular, the embedding vectors of users and items in

GMF are simply obtained by training from the weighted output of the embedding dot product using the interaction matrix:

$$\hat{\boldsymbol{R}}_{u,v} = \boldsymbol{W}\sigma\left(\boldsymbol{U}_u \odot \boldsymbol{V}_V\right), \tag{4.10}$$

where $\odot$ denotes the element-wise product of vectors, σ is an activation function and $\boldsymbol{W}$ is the trainable parameter. However, these models are unable to leverage the relations from the heterogeneous entity features. We note that our pre-training model using the graph neural network can be seen as a generalisation of the GMF model, since our **Multi-P** model can be reduced to the GMF model by removing all the links of the constructed multi-graphs and setting the maximum depth of the multi-graph neural network to be 1. More recently, Hao et al. (2021b) exploited how to use GNN to conduct pre-training for downstream tasks, inspired by the GNN pre-training (Hu et al., 2020a). However, their proposed models only leverage the graph structure, which lacks the ability to incorporate heterogeneous side information about the entities, compared with our proposed scheme.

In relation to the graph-based recommender systems, the most relevant works to our **Multi-P** model is the Multi-GCCF model (Sun et al., 2019), which, similarly, considers the user-to-user and item-to-item relations as graphs and uses a graph convolution network to train the embeddings of the two entities. However, the entity graphs in Multi-GCCF are constructed from the rating/click matrix, rather than from the entity side information. Hence, the heterogeneous relations from the entity side information cannot be captured by the Multi-GCCF model.

4.5 Experimental Setup

In this section, we first introduce the research questions that we aim to answer in this chapter (Section 4.5.1). Next, we present the datasets used for conducting the experiments as well as the relevant pre-processing procedures to prepare the datasets including all interaction data and different types of side information (Section 4.5.2). Finally, we present the experimental settings and describe the used baselines (section 4.5.3).

4.5.1 Research Questions

We aim to answer the following research questions:

RQ4.1. Do our pre-trained models help existing recommendation systems obtain better performances?

RQ4.2. Do our pre-trained models outperform the existing state-of-the-art recommenders that use side information?

RQ4.3. Are the performance improvements gained through our pre-training models due to the multiple types of side information?

RQ4.4. Does the pre-training process help to improve the stability of the existing models?

RQ4.5. Does the pre-training process help to alleviate the classical *cold-start* problem?
RQ4.6. How do the embeddings dimension and different ranking cut-offs affect the recommendation performances of the pre-trained recommenders?

Table 4.1: Statistics of the datasets.

Dataset	# Users	# Items	# Interactions	# User/Item Features
Foursquare	2,060	2,876	27,149	2,108/47
MovieLens-1M	6,040	3,704	1,000,209	21/18

4.5.2 Datasets

To evaluate the effectiveness of our introduced pre-training scheme, we use three datasets, namely Foursquare, MovieLens-1M, and Epinions. We do not use the Librarything and Yelp datasets (see Table 3.2) because they lack different types of side information for building the single-relational and multi-relational graphs. Table 4.1 shows the statistics of the Foursquare and MovieLens-1M datasets and the statistic of the Epinions dataset can be found in Table 3.2. For the MovieLens-1M dataset, the users' features (i.e., side information) are "gender", "age" and "occupation", while the items' features are the 18 different genres. For the Foursquare dataset, we use the tags provided by online users as features for the restaurants, while we represent each user as a bag-of-words feature vector from his/her own reviews (stopwords are removed). Similarly, for the Epinions dataset, we represent both users and items with bag-of-words feature vectors from their associated reviews with stopwords removed, selecting the 10 most frequent words to represent both users and items. Among the three datasets, there is only one real-valued feature, namely ages in the MovieLens-1M dataset, which needs to be pre-processed into one-hot representations by a categorisation operation. Specifically, the users' age feature in the MovieLens-1M dataset is categorised into 8 age groups, each with a step of 10 years. Then a one-hot vector is used to represent the users' age feature. Below, we summarise how to construct the single-relational graph and the multi-relational graph for our proposed **Multi-P** and **Single-P** models, respectively.

Single-relational graph construction. To construct a single-relation graph for **Single-P**, we first need to build the one-hot vectors for each user and item. Taking the MovieLens-1M dataset as an example, we know that there are 18 different genres for all movies; therefore, each movie is represented as a binary vector of size 1$\times$18. If one movie is labelled as a comedy movie, its vector will have 1 at the corresponding column of "comedy" and 0 elsewhere if no other labels are given. Hence, the feature matrix $\boldsymbol{F}_v$ of the MovieLens-1M dataset has a size of 3704$\times$18. Next, we compute cosine similarities between each pair of the users' one-hot vectors or each pair of the items' one-hot vectors so that we obtain cosine similarities between each user pair and each item pair. Such a single relational graph will have a size of 3704$\times$3704 containing all similarity values. Finally, this single-relational graph can be used as an input for the proposed **Single-P** model. We follow a similar procedure to compute the single-relational graph for users.

Multi-relational graph construction. To construct a multi-relational graph for **Multi-P**, we also follow a similar procedure to build the one-hot vectors for each user and item so as to compute the feature matrices $\boldsymbol{F}_v$ and $\boldsymbol{F}_u$. Instead of using the cosine similarity to capture the similarity values between each pair of entities, the input of **Multi-P** is a series of graphs, where each graph contains all entities possessing one specific type of side information. Therefore, if we take again the MovieLens-1M dataset as an example, the input of **Multi-P** for items will be 18 different graphs, where each graph has a size of $m \times m$. Here, the value of m depends on how many entities are involved in the multi-relational graph.

4.5.3 Experimental Settings

Following the leave-one-out splitting used in Section 3.4.2, we split the interactions of each dataset into training, validation and 10 different testing sets. Three ranking evaluation metrics, namely the Normalised Discounted Cumulative Gain (NDCG), Recall and Mean Average Precision (MAP) metrics, are applied for evaluating the performances of our evaluated models. Detailed definitions of all used evaluation metrics can be found in Section 2.1.2.2.

We evaluate the effectiveness of our pre-training scheme by comparing it with ten existing state-of-the-art recommendation models. Among the baselines, four are general (representation-based) recommender systems, which are also used for the subsequent fine-tuning. Specifically, these four baselines i.e., MF, NCF, NGCF and LightGCN, have been described in Section 3.4.1.

Differing from these four general recommenders, the other three baselines can be categorised as recommender systems that incorporate content information (Section 2.1.1.3). These baselines use an integration scheme to incorporate the side information of both users and items:

- **HIRE** (Liu et al., 2019b): This is a side information-aware recommendation model, which combines the flat and hierarchical side information to alleviate the challenge brought by the heterogeneity of the side information.

- **cVAE** (Chen and de Rijke, 2018): cVAE is a side information-aware recommendation model that uses the variational auto-encoder to encode the entity side information into entities for enhancing the performance.

- **SSLIM** (Ning and Karypis, 2012): This is a classical sparse linear recommender, which can use both the users and items' side information. In particular, we choose the binary representation to remain consistent with our feature representations.

In addition, we incorporate three other baselines, which use different methods to enhance their corresponding graph-based recommenders:

- SGL (Wu et al., 2021): As described in Section 3.4.1, SGL uses self-supervised learning, including the node dropout, edge dropout and random walk augmentation techniques to generate multiple representations of users and items based on the structure of the graph.

- PT-GNN (Hao et al., 2021b)[3]: PT-GNN uses the pre-training technique of the GNN model to enhance the embeddings of the cold-start users or items, which is similar to the technique introduced in Section 2.2.3. The pre-training task of PT-GNN consists in directly reconstructing the cold-start user/item embeddings by mimicking the meta-learning setting via episode-based training (Vinyals et al., 2016). In (Hao et al., 2021b), many state-of-the-art baseline models (including LightGCN (He et al., 2020), GraphSAGE (Hamilton et al., 2017a) and GAT (Veličković et al., 2018)) have been shown to be enhanced by PT-GNN. We choose the variant using GraphSAGE as one of our baselines due to its competitive overall performance. In the following, for simplicity, we use PT-GNN to denote the variant using GraphSAGE.

- SimGCL (Yu et al., 2022a): This is a recently proposed graph contrastive recommender with a high effectiveness and flexibility. SimGCL can enhance the contrastive representations of users and items by introducing a regulated noise sampled from the uniform distribution instead of relying on those graph augmentation techniques introduced in Section 2.2.4.

Furthermore, we compare our proposed pre-training scheme with variants, consisting of pre-training only and multi-task learning:

- Single-P (pre. only) & Multi-P (pre. only): These two models only use the users and items' embeddings learned during the **Single-P** or **Multi-P** pre-training stages to make recommendations. By comparing these two variants with the models pre-trained by our proposed scheme, we can directly observe the obtained improvements of using pre-training and fine-tuning together.

- $\text{MTL}_{+\textbf{Single-P}}$ & $\text{MTL}_{+\textbf{Multi-P}}$: These two variants use multi-task learning to train **Single-P** or **Multi-P** together with a baseline recommender, instead of following our proposed scheme. These two variants use $\mathcal{L}_{PT} + \beta * \mathcal{L}_{FT}$ as the overall training loss, where β is a constant and the detailed equations of $\mathcal{L}_{PT}$ and $\mathcal{L}_{FT}$ can be found in Section 4.4.3 and Section 4.4.6, respectively.

We apply our two pre-training models on the four existing widely used representation-based baselines, namely MF, NCF, NGCF and LightGCN. We use **Single-P** and **Multi-P** to denote the two pre-training models, respectively. $\text{MF}_{+\textbf{Multi-P}}$ stands for a model, pre-trained by **Multi-P** and fine-tuned with MF.

[3] In (Hao et al., 2021b), no explicit name has been given for the proposed graph pre-training model. For simplicity, we use PT-GNN to denote the model.

We adopt the Adam (Kingma and Ba, 2015) optimiser in both **Multi-P** and **Single-P** as well as the four representation-based baselines. To determine the values of all hyperparameters, we randomly sample one interaction for each user as the validation set and tune the hyperparameters on it for all of the models. In particular, we tune the pre-training models (i.e., **Multi-P** and **Single-P**) by varying the learning rate in $\{10^{-2}, 10^{-3}, 10^{-4}\}$ and the regularisation weight η in $\{10^{-2}, ..., 10^{-5}\}$. The learning rates of the baseline models are also tuned according to the suggested ranges from the original papers. The depths for all GNNs of the graph-based recommenders (i.e., NGCF, LightGCN, SGL and SimGCL) and the pre-training models are kept to 3 with each layer having a size of 64, while the dropout ratios of all GNNs vary among $\{0.3, 0.4, ..., 0.8\}$ as suggested in the existing literature (He et al., 2020). We set the maximum number of training epochs to 500; the batch size to 1000 and the latent dimension to 64 for all models. Moreover, we use an early stopping strategy, i.e., we apply a premature stopping if NDCG@10 on the validation data does not increase for 50 successive epochs. Note that the embedding dimension d is a hyperparameter for both the pre-training models and the fine-tuning models (i.e., the baseline models). For a fair comparison, we set this hyperparameter to 64, since most of our experimental models can almost achieve their best performances for this dimension size across our three datasets. To make a fair comparison between our proposed scheme and those three baselines, i.e., HIRE, cVAE and SSLIM, which also incorporate the side information of both users and items, we train the baselines using the same side information used by our proposed scheme for pre-training. For our **Multi-P** model, the number of learnable basis vectors b is set to 10, which is empirically tuned from the set {5, 7, 10, 20} by using the validation set for all datasets.[4] For those two multi-task learning model variants, i.e., $\text{MTL}_{+\textbf{Single-P}}$ & $\text{MTL}_{+\textbf{Multi-P}}$, we tune the hyperparameter β among {0.1, 0.5, 1} following (Wu et al., 2021).

4.6 Results and Analysis

In this section, we report the results obtained from four main experiments aimed at answering the research questions listed in Section 4.5.1. In particular, we first address **RQ4.1** by analysing whether both **Single-P** and **Multi-P** can help to improve the four existing representative recommenders. Then, we further compare the performances of these models with three competitive recommenders, which incorporate different graph-based techniques and three recommenders that leverage both the users' and items' side information (**RQ4.2**). We provide an ablation study where we randomly drop {20%, 40%, 60%, 80%} of the entity features during the pre-training process in order to seek an answer to **RQ4.3**. To answer **RQ4.4**, we conduct experiments over different random seeds, and analyse the standard deviations of these models' performances. To

[4] Note that a more thorough tuning of this parameter may further improve the recommendation performances, but we did not observe a clear performance trend over different b values in our experiments, and under this setting we have already obtained excellent performances that can be used to draw our conclusions.

address **RQ4.5**, we conduct an analysis, where we compare the best observed performing pre-trained models (i.e., LightGCN$_{+\textbf{Single-P}}$ and LightGCN$_{+\textbf{Multi-P}}$) with PT-GNN and LightGCN across different groups of users to examine whether our proposed scheme can help to alleviate the *cold-start* problem. Finally, to answer **RQ4.6**, we provide a detailed analysis of the performances of the pre-training models on different embedding dimensions and different cut-offs for the recommended items.

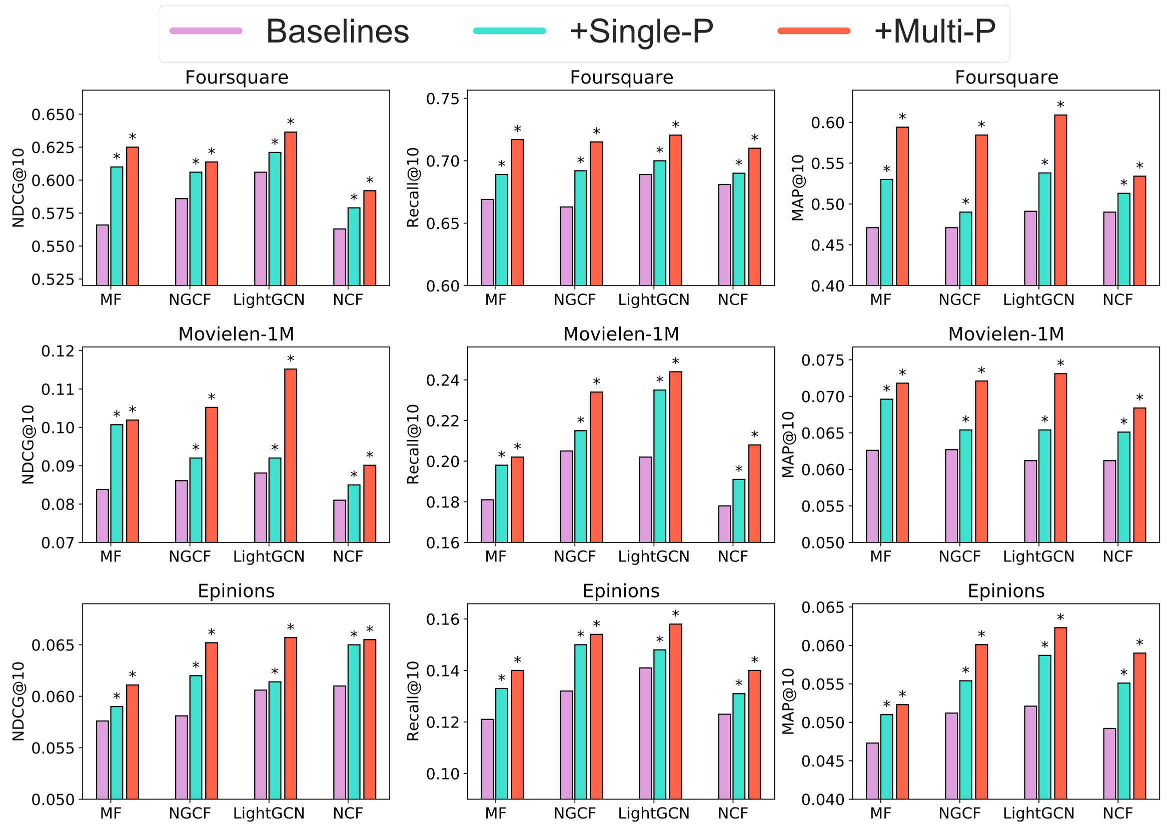


Figure 4.5: Performance comparison of the four selected existing recommenders with the **Single-P** and **Multi-P** pre-training processes. We use $*$ to denote a significant difference between the performances of the baselines and their pre-trained variants, according to the paired t-test with the Holm-Bonferroni correction for $p<0.01$.

4.6.1 Effectiveness of Pre-training

To validate the effectiveness of our pre-training scheme, we compare the performances of the four selected recommender models (i.e., MF, NCF, NGCF and LightGCN) with their pre-trained variants under the pre-training processes defined by our **Multi-P** and **Single-P** schemes. Figure 4.5 reports the recommendation performances comparison in terms of the NDCG, Recall and MAP metrics at a rank cut-off of 10. From Figure 4.5, we can clearly observe that, over the three used datasets, all the selected 4 recommender models exhibit significantly improved performances when **Multi-P** is applied. Moreover, we can also see that a baseline pre-trained with our **Multi-P** scheme can always outperform the baseline pre-trained with **Single-P**. This

Table 4.2: Performance comparison of recommenders with side information. The best and second best performances are marked in boldface or underlined, respectively. We use $*$ to denote a significant difference between the performances of the side information-aware baselines and the best proposed model i.e., LightGCN$_{+\textbf{Multi-P}}$, according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$.

Model	Foursquare		MovieLens-1M		Epinions	
	NDCG	MAP	NDCG	MAP	NDCG	MAP
HIRE	0.5232*	0.4575*	0.0997*	<u>0.0692</u>*	0.0667*	0.0518*
cVAE	0.5326*	0.4438*	0.0678*	0.0453*	0.0532*	0.0410*
SSLIM	0.5894*	0.4691*	0.0655*	0.0441*	0.0533*	0.0401*
SGL	0.6051*	0.5691*	0.0739*	0.0485*	0.0710*	0.0484*
PT-GNN	0.6048*	0.5687*	0.0755*	0.0501*	0.0697*	0.0503*
SimGCL	0.6121*	0.5891*	0.0758*	0.0541*	<u>0.0723</u>*	0.0481*
MF$_{+\textbf{Single-P}}$	0.6206	0.5901	0.0979	0.0646	0.0527	0.0434
MF$_{+\textbf{Multi-P}}$	<u>0.6249</u>	<u>0.5944</u>	<u>0.1019</u>	**0.0718**	0.0587	0.0498
NGCF$_{+\textbf{Single-P}}$	0.6016	0.5683	0.0713	0.0450	0.0708	0.0454
NGCF$_{+\textbf{Multi-P}}$	0.6138	0.5844	0.0752	0.0461	0.0719	0.0485
LightGCN$_{+\textbf{Single-P}}$	0.6162	0.5940	0.0952	0.0631	0.0717	<u>0.0594</u>
LightGCN$_{+\textbf{Multi-P}}$	**0.6364**	**0.6089**	**0.1068**	0.0689	**0.0792**	**0.0623**
NCF$_{+\textbf{Single-P}}$	0.5677	0.4939	0.0870	0.0551	0.0620	0.0531
NCF$_{+\textbf{Multi-P}}$	0.6021	0.5340	0.0913	0.0584	0.0691	0.0583
Single-P (pre. only)	0.5758	0.5093	0.0700	0.0431	0.0517	0.0421
Multi-P (pre. only)	0.5912	0.5235	0.0812	0.0481	0.0601	0.0510
MTL$_{+\textbf{Single-P}}$	0.6012	0.5093	0.0725	0.0453	0.0522	0.0431
MTL$_{+\textbf{Multi-P}}$	0.6100	0.5337	0.0810	0.0482	0.0615	0.0530

is somewhat expected, as **Multi-P** is trained using the multi-graphs constructed from the entities' side information. Therefore, **Multi-P** is capable of capturing the heterogeneous relations between entities within the side information, in contrast to **Single-P**, which can only leverage the similarity between each feature vector.

To conclude on **RQ4.1**, we have shown that our proposed **Multi-P** model can effectively leverage different types of users and items' side information, thereby enhancing the existing representation-based recommenders with significant performance improvements, consistent across the three used datasets, three measures and four baselines. Therefore, based on the consistent performance improvements achieved by the graph pre-training technique, we can consolidate the hypothesis in our proposed thesis statement.

4.6.2 Effectiveness of Integrating Side Information

Having shown that our proposed **Multi-P** is effective at enhancing the performances of the existing recommenders through the leveraging of side information, we next examine whether these recommenders with our pre-training scheme perform better than the existing state-of-the-art models. To answer this, we further compare the pre-trained models (i.e., MF$_{+\textbf{Multi-P}}$, NCF$_{+\textbf{Multi-P}}$,NGCF$_{+\textbf{Multi-P}}$ and LightGCN$_{+\textbf{Multi-P}}$) with their corresponding baselines pre-trained with **Single-P** as well as three state-of-the-art recommenders where both side information of users and items are used. Table 4.2 reports the recommendation performances in terms of the NDCG and MAP metrics at rank cut-off 10 for each model across all three datasets. From Table 4.2, we observe that although the relative performance ranking of systems is different across the used datasets, the three baselines (i.e., HIRE, cVAE and SSLIM) do not achieve the highest performances on any of the used datasets. Specifically, the LightGCN$_{+\textbf{Multi-P}}$ model performs the best on both the Epinions and Foursquare datasets, outperforming the NDCG@10 score of the HIRE side information-aware baseline by 18.7% (0.0667 $\rightarrow$ 0.0792) and 21.6% (0.5232 $\rightarrow$ 0.6364), respectively. For the MovieLens-1M dataset, MF$_{+\textbf{Multi-P}}$ achieves the best performance on the MAP metric, outperforming the HIRE model by 3.76% (0.0692 $\rightarrow$ 0.0718). Furthermore, we compare the pre-trained models with four variants that also use side information: Single-P (pre. only), Multi-P (pre. only), MTL$_{+\textbf{Single-P}}$ and MTL$_{+\textbf{Multi-P}}$. Recall that **Single-P** (pre. only) and **Multi-P** (pre. only) are two variants that only use the pre-training models without fine-tuning; MTL$_{+\textbf{Single-P}}$ and MTL$_{+\textbf{Multi-P}}$ are two variants that use multi-task learning to train the models. From Table 4.2, we find that the variants using multi-task learning always outperform the two pre-training only variants. However, none of these variants can reach the best or second best performances on all three used datasets, further demonstrating our proposed scheme's superiority. We also notice that models pre-trained by our proposed scheme cannot consistently outperform the pre-training only variant. For example, the NCF$_{+\textbf{Single-P}}$ model is less effective than the pre-training only variant **Single-P** (pre. only) on the Foursquare dataset. This suggests that the performance of **Single-P** has decreased after being fine-tuned by the NCF model. Such an observation is related to the well-known issue of training a deep neural network with a warm restart (Loshchilov and Hutter, 2016) when a pre-trained model cannot even achieve the previous local optima during the fine-tuning process.

In addition to comparing our proposed scheme with side information-based baselines, we incorporate another three baselines, which use different graph-based methods to enhance the recommendation performance instead of relying on side information. Specifically, the SGL model uses the graph self-supervised learning method, PT-GNN uses the graph pre-training method, and SimGCL uses the graph contrastive learning method. By comparing the SGL, PT-GNN and SimGCL models with our proposed scheme in Table 4.2, we observe that our proposed scheme can achieve the best performances for all cases in terms of NDCG@10. This observation shows that using our proposed pre-training scheme to integrate side information outperforms baselines that use other competitive graph-based techniques.

Overall, to answer **RQ4.2**, we have shown that the four selected representative recommendation models with our **Multi-P** pre-training scheme can outperform the other three baseline models that also leverage the side information using the integration scheme on all the three used

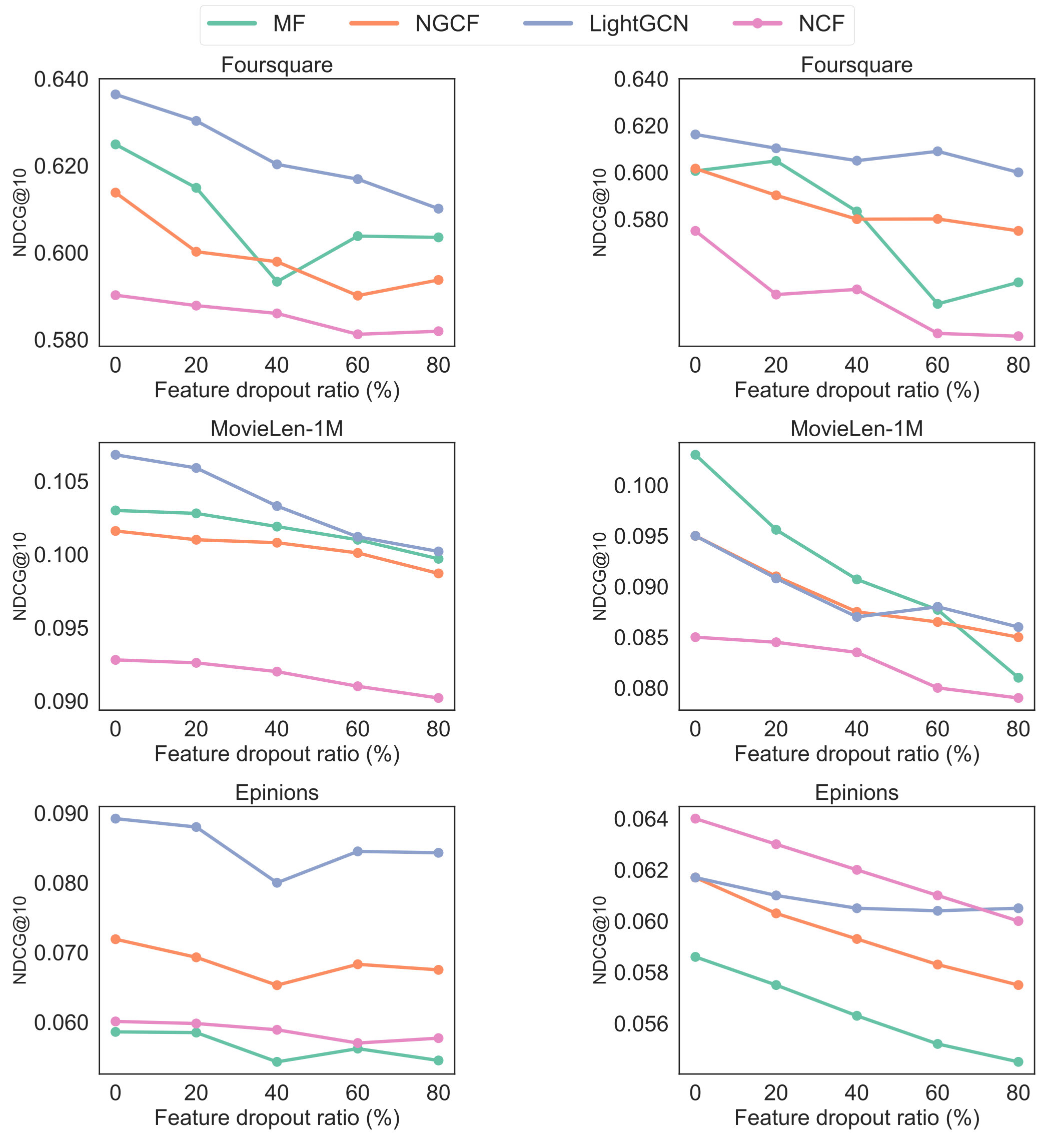
Table 4.3: Standard deviations (denoted std.) and means of the NDCG@10 performances over 50 random seeds. Lower standard deviation (in bold) means a better stability.

Model	Foursquare		Epinions	
	std.	mean	std.	mean
MF	0.0197	0.5163	0.0127	0.0501
MF$_{+\textbf{Single-P}}$	0.0046	0.6206	0.0049	0.0527
MF$_{+\textbf{Multi-P}}$	**0.0029**	0.6249	**0.0043**	0.0587
NGCF	0.0237	0.5162	0.0108	0.0681
NGCF$_{+\textbf{Single-P}}$	0.0053	0.6016	0.0042	0.0708
NGCF$_{+\textbf{Multi-P}}$	**0.0044**	0.6138	**0.0038**	0.0719
LightGCN	0.0209	0.5365	0.0092	0.0706
LightGCN$_{+\textbf{Single-P}}$	0.0050	0.6262	0.0039	0.0717
LightGCN$_{+\textbf{Multi-P}}$	**0.0039**	0.6264	**0.0031**	0.0792
NCF	0.0283	0.4621	0.0159	0.0591
NCF$_{+\textbf{Single-P}}$	0.0091	0.5777	0.0079	0.0620
NCF$_{+\textbf{Multi-P}}$	**0.0086**	0.6021	**0.0064**	0.0691

datasets. Therefore, since our pre-trained models can outperform not only general baselines but also baselines using the side information, we become more confident in our proposed statement that the graph pre-training technique can enhance the performance of recommender systems.

4.6.3 Ablation Study of Side Information

Having observed that all the evaluated existing baseline models are significantly improved by our pre-training scheme, we now check whether these improvements are actually the result of using the heterogeneous relations among the multiple types of side information, captured by our pre-training models (**RQ4.3**). To answer this question, we conduct an ablation study to examine the effect of randomly removing entity features, thereby revealing the connection between the performance improvements and the side information. Specifically, we randomly drop different proportions ({20%, 40%, 60%, 80%}) of entity features during the pre-training process, and evaluate the recommendation performances of the fine-tuned models given these pre-trained embeddings. Figure 4.6 reports the obtained results. From Figure 4.6, we can see that all the NDCG@10 performances of all the fine-tuned models decrease as the features dropout ratio increases from 0% (i.e., no dropped features) to 80% (80% of features are dropped) in all the three datasets. This result suggests that randomly dropping entity features does hurt the overall recommendation performance. This result also suggests that the performance improvements are indeed gained from the entity features and our **Single-P**. In addition, **Multi-P** models are able to accurately capture the heterogeneous relations from the multiple types of side information, which validates the hypothesis in our thesis statement.



(a) **Multi-P** (b) **Single-P**

Figure 4.6: Results of the baselines deploying the **Multi-P** and **Single-P** models over different dropout ratio of entity features.

4.6.4 Stability Analysis

In the previous sections, we have demonstrated the general applicability and effectiveness of our introduced pre-training scheme. To address **RQ4.4**, we calculate the standard deviations of the NDCG@10 performances of the baseline models (i.e., MF, NCF, NGCF and LightGCN) and their enhanced variants by our **Multi-P** and **Single-P** models. To avoid repetition, we only present the experimental results on the Foursquare and Epinions datasets (see Table 4.3) because similar results are observed on the MovieLens dataset. From the table, we observe that **Multi-P** markedly improves the performances and stabilities of all the used baselines, with much smaller observed standard deviations in each paired comparison of a baseline model with and without the use of the pre-training scheme. Noticeably, although less effective than our

Multi-P model, the **Single-P** model can also bring a marked stability enhancement to all baselines, which demonstrates the superiority of the graph pre-training scheme for recommender systems. To conclude, our proposed pre-training scheme can enhance the performance of each of the representation-based recommender systems as well as their stability, thereby alleviating the low-robustness issue introduced in Section 1.1 and observed in Figure 4.1. Therefore, our stability analysis consolidate the postulation in our proposed thesis statement by showing that the graph pre-training technique can be used to enhance the recommendation performance and address the low-robustness issue at the same time.

4.6.5 *Cold-start* Analysis

To answer **RQ4.5**, we evaluate the performances of LightGCN, PT-GNN, LightGCN$_{+\textbf{Single-P}}$ and LightGCN$_{+\textbf{Multi-P}}$ on the Foursquare and Epinions datasets[5] over different groups of users, respectively. Specifically, following existing work (Huang et al., 2021a; Liu et al., 2020), we consider that the *cold-start* users are those users with fewer than 10 interactions and the *regular* users are those users with more than 10 interactions. In Table 4.4, we report the NDCG@10 performances of the overall (i.e., all users included), *cold-start* (i.e., *cold-start* users only) and *regular* groups (i.e., *regular* users only) using all the evaluated models, respectively. In particular, we specifically choose PT-GNN as a baseline since it is especially designed to improve the *cold-start* recommendation (Hao et al., 2021b). In addition, we choose LightGCN and its two pre-trained variants LightGCN$_{+\textbf{Single-P}}$ and LightGCN$_{+\textbf{Multi-P}}$ for their excellent overall performances as shown in Table 4.2. From Table 4.4, we observe that our LightGCN$_{+\textbf{Single-P}}$ and LightGCN$_{+\textbf{Multi-P}}$ models consistently outperform the original LightGCN model and the pre-training baseline, i.e., PT-GNN, on both used datasets across different groups of users. It is worth noting that although PT-GNN markedly outperforms the LightGCN baseline for the *cold-start* users on the Foursquare dataset (0.5117 $\rightarrow$ 0.5334) and on the Epinions dataset (0.0532 $\rightarrow$ 0.0603), the improvements for the *regular* users are relatively marginal, especially for the Foursquare dataset, where NDCG@10 is only improved from 0.6238 to 0.6239. On the contrary, our proposed scheme can boost the recommendation performance for different groups of users instead of only focusing on the *cold-start* users. The reason why our proposed scheme can consistently improve the performance of the original model is that we use abundant side information to construct the relational graphs. Since the side information is available for different groups of users (sometimes the *regular* users have even more attributes), in general our proposed scheme can enhance the representations of users. In comparison, PT-GNN relies on the interaction graph to construct the embeddings of the *cold-start* users, which might only benefit these *cold-start* users instead of all users.

[5] We only use the Foursquare and Epinions datasets because the MovieLens-1M dataset has only users with more than 20 interactions. However, typically, users with more than 20 interactions can hardly be called as *cold-start* users.

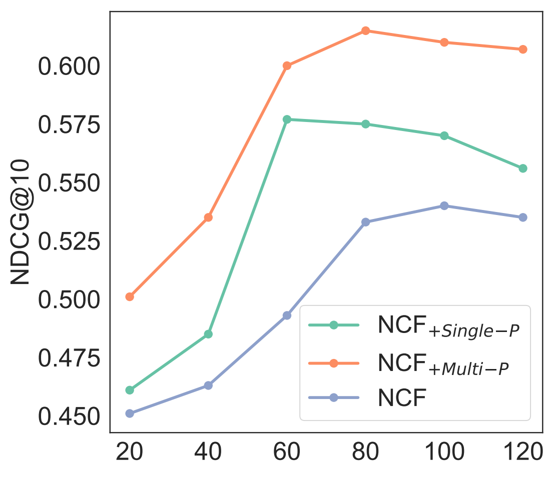
Table 4.4: NDCG@10 performances of our proposed scheme and those of the baselines across different groups of users on the two used datasets.

Model	Foursquare			Epinions		
	Overall	Cold-start	Regular	Overall	Cold-start	Regular
LightGCN	0.6012	0.5117	0.6238	0.0611	0.0532	0.0673
PT-GNN	0.6048	0.5334	0.6239	0.0697	0.0603	0.0700
LightGCN$_{+\textbf{Single-P}}$	0.6162	0.5458	0.6397	0.0717	0.0653	0.0731
LightGCN$_{+\textbf{Multi-P}}$	0.6364	0.5529	0.6458	0.0792	0.0701	0.0802

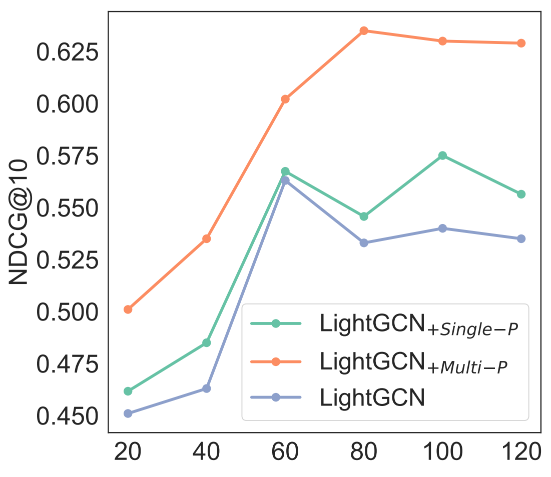
To conclude, we have shown that our proposed scheme is able to alleviate the *cold-start* problem, while still ensuring an effective recommendation for all other users in comparison to strong baselines. In addition, we further validate our proposed thesis statement by demonstrating that the graph pre-training technique can alleviate the *cold-start* problem when incorporating different types of side information.

4.6.6 Hyperparameter Analysis

Finally, to answer **RQ4.6**, we study how the embedding dimension affects the recommendation performance. Figure 4.7 shows the performance comparison results of our pre-trained models (i.e., LightGCN$_{+\textbf{Single-P}}$, NCF$_{+\textbf{Single-P}}$, NCF$_{+\textbf{Multi-P}}$ and LightGCN$_{+\textbf{Multi-P}}$) with their baselines (i.e., LightGCN and NCF). From Figure 4.7, we observe that the size of the embedding dimension does affect the final recommendation performances of all these evaluated models. We can also observe that when the size of the latent dimensions is $\leq$40, all three models show relatively poor performances, which can be further boosted when the dimension size increases; almost all the models' performances reach their highest points when the dimensions are between 60 to 80. Recall that, for a fair comparison, we fixed the embedding dimension to 64 for all the implemented models, which can be further justified from these obtained results. Moreover, Figure 4.7 demonstrates that our pre-training scheme can bring consistent improvements to the exiting models across different embedding dimensions. We also plot the performances evaluated by NDCG over different cut-offs (k) of the items ranking in Figure 4.8. We can also see that both LightGCN$_{+\textbf{Multi-P}}$ and NCF$_{+\textbf{Multi-P}}$ consistently improve over their baseline models for different cut-offs. In particular, we see that even for some low rank cut-offs (e.g., $k = \{1, 3, 5\}$) or for deep cut-offs (e.g., $k = 50$) our **Multi-P** model can still enhance the LightGCN and NCF models with a marked improvement, which means that our per-training scheme can help improve the recommendation performance under different circumstances when different amount of items (i.e., cut-off values) are chosen to be exposed to the users. Therefore, this hyperparameter analysis consolidates our proposed thesis statement by showing that the improvements brought by the graph pre-training technique are consistent under different circumstances.

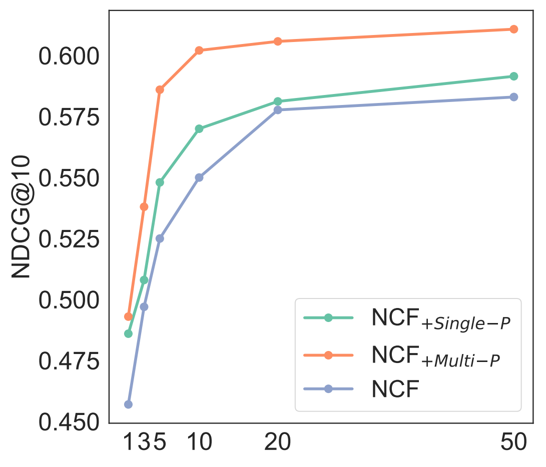


(a) NCF

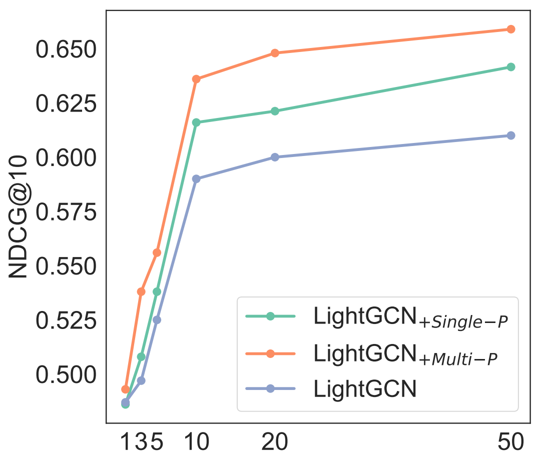


(b) LightGCN

Figure 4.7: The performance comparison over different dimensions on the Foursquare dataset.



(a) NCF



(b) LightGCN

Figure 4.8: The performance comparison over different cut-off values on the Foursquare dataset.

4.7 Conclusions

In this chapter, we introduced a novel pre-training scheme for recommender systems to leverage the entity side information in a general manner. In particular, we proposed two models for pre-training the entity representations to leverage multiple types of side information, based on the graphs constructed from the entity side information. In answer to **RQ4.1** and **RQ4.4**, the extensive evaluation of our pre-training scheme with the fine-tuning of four existing representation-based recommenders showed that effectively pre-training the embeddings with both the users and items' side information improved these existing models in terms of both effectiveness (always significantly, see Figure 4.5) and stability (see Table 4.3). Furthermore, compared to the existing state-of-the-art recommender baselines, which integrate the same side information, our **Multi-P** model exhibited up to 7% improvement in NDCG@10 for the MovieLens-1M dataset, 21% improvement on the Foursquare dataset and 48.6% improvement on the Epinions dataset (see Table 4.2). In addition, our **Multi-P** model consistently and significantly outperformed recent baselines that incorporate self-supervised learning, graph contrastive learning or graph pre-training techniques. Moreover, we have also shown through an in-depth analysis that by leveraging the side information through pre-training, our **Single-P** and **Multi-P** models can successfully alleviate the classical *cold-start* problem while ensuring effective recommendations for all other users (see Table 4.4). We also showed that the **Multi-P** model, pre-trained using multi-graphs, can always outperform the **Single-P** model, which suggests that more information is captured through the use multi-graphs. Our pre-training scheme provides a general framework for leveraging side information, which can be used to enhance a general representation-based recommendation model.

In summary, we have validated the hypothesis in our proposed thesis statement in Section 1.2, namely, by leveraging the graph pre-training to incorporate multiple side information, a graph-based recommender system can achieve enhanced performance and alleviate the *cold-start* problem as well as the low-robustness issue. In particular, we tackled a more generalised problem in this chapter compared with the SGP model proposed in Chapter 3, where only the social relations are incorporated. Specifically, our proposed **Single-P** scheme can be regarded as an extension of our SGP model (Chapter 1.2) towards the scenario of multiple side information. In addition, the **Multi-P** scheme is more advanced than **Single-P** given that it can capture the multi-relations from the graph.

In the next chapter, we aim to tackle the issue of less effective negative items mentioned in Section 1.1. Specifically, we focus on using the graph contrastive learning technique to improve the graph-based recommender systems from the sampling perspective. In particular, first, we will investigate how to sample more informative negative items in Chapter 5. Then we will focus on combining additional positive and negative samples in Chapter 6.

Chapter 5

Graph Contrastive Negative Sampling for Recommendations

5.1 Introduction

In Chapter 4, we proposed two pre-training schemes namely, **Single-P** and **Multi-P** the effectiveness of which validated our hypothesis in the proposed thesis statement (Section 1.2) that the recommendation performance can be enhanced by leveraging multiple side information using the graph pre-training technique. However, users are becoming less willing to share their personal information due to privacy issues, making it challenging and risky to solely rely on such side information (Badsha et al., 2016; Wang et al., 2021a). To tackle this issue, contrastive learning (Chopra et al., 2005; Robinson et al., 2020) is a commonly used technique for its promising performance. As we have mentioned in Section 2.2.4, we can leverage the graph contrastive learning technique to generate or sample more informative samples in the training process. In particular, graph contrastive learning can be used to generate more informative negative items to improve the recommendation performance. Therefore, in this chapter, we aim to study how to leverage the graph contrastive learning technique to sample more informative negative items to enhance the randomly sampled (user, positive item, negative item) triplets such that the final recommendation performance of ranking-based models can be improved.

As introduced in Section 2.1, under the training and optimisation process of BPR (see Equation 2.5), each input instance is formed as a triplet consisting of one user and a pair of positive & negative items, where the negative items are randomly sampled from the whole corpus of each user's uninteracted with items (i.e., random Negative Sampling (NS)). Ideally, all negative items need be used to construct the training triplets. However, given that users only interact with a few items, it is infeasible to pair each (user, positive item) with all negative items (Wang et al., 2020). Hence, BPR and its variants use the random negative sampling approach to assign a single negative item to each (user, positive item) pair. Although efficient, the random negative sampling approach lacks the ability to search for more informative negative items, which are important for an enhanced recommendation performance (Chuang et al., 2020). Specifically, we consider those items that are unlikely to be interacted with by a user as the informative negative items for this user. Furthermore, the random negative sampling approach might induce false-negatives and the so-called exposure bias (Khenissi et al., 2020) as described in Section 2.1.1.1. To alleviate the issues of the random negative sampling approach, some recent works have been proposed to modify the sampling strategy by considering the items' popularity (Zheng et al., 2020b) or the items' side information (Khenissi et al., 2020; Manotumruksa et al., 2017) (e.g., geographical locations, tags). However, these negative sampling approaches are either less effective or heavily rely on conditional items selection based on sparse categorisation of items Ding et al. (2019). Due to the fact that the categorisations of items are not always available, those negative sampling approaches cannot be deployed in general interaction datasets. Therefore, a more generalised negative sampling approach that does not rely on side information is highly desirable.

To alleviate the difficulty of choosing appropriate negative items without side information, we aim to incorporate the embeddings learned by a general recommender system. Given that the performance of a recommender system is expected to increase along the training process before saturation, one can become more and more confident that the lower-ranked items generated by the trained model are less likely to be interacted with by each user. Specifically, with the learned embeddings generated after each training epoch, we can better predict which item is an informative negative item in comparison to randomly sampled items. Hence, our underlying intuition is that dynamically sampling negative items using the learned embeddings during the training process can help general recommender systems to fetch more informative negative samples.

Based on the aforementioned idea, in this chapter, we propose a general sampling scheme, called Dynamic Negative Sampling (DNS). Our DNS scheme can dynamically select negative items solely based the embeddings of users and items learned during the training of a general recommender without requiring any side information nor any extra data preprocessing. Furthermore, to leverage these additional negative items sampled at each checkpoint, our DNS scheme also includes an objective function motivated by the Information Noise Contrastive Estimation (InfoNCE) loss (Oord et al., 2018), a contrastive loss function originally devised to push multiple negative samples away from the positive sample. Based on information theory (Carter, 2007), we know that a distribution with a higher entropy contains richer information. To further motivate our proposed DNS scheme, we directly inspect the dynamic change of information during the training by introducing an entropy measure. Specifically, we use the proposed measure to approximate the information contained in different groups of negative items in order to compare the informativeness of random negative items with the dynamically sampled ones.

To summarise, this chapter makes the following contributions:

• We propose a dynamic negative sampling (DNS) scheme, which can dynamically select the negative items and improve the recommendation performance of general recommender systems.
• We propose a novel contrastive objective function inspired by InfoNCE to work collaboratively

within the DNS scheme in order to leverage multiple negative items.
• We propose an entropy measure to observe the change of information by approximating the dynamic information gain obtained from negative samples.

5.2 Negative Sampling in Recommender Systems

In this section, we briefly position our work in the context of negative sampling approaches as well as recommender systems building on advanced negative sampling approaches.

Negative sampling (NS) approaches are widely applied in pairwise and listwise learning recommendation models to sample negative items for (user, positive item) pairs. As argued in Section 2.1.1.1, most of the items have not been interacted with by each user and hence many items are negative items for each user. Therefore, negative sampling is the process of selecting a portion of negative items from the whole corpus of the user's uninteracted with items. In general, there are mainly two types of negative sampling approaches, namely heuristic (Manotumruksa et al., 2017; Rendle et al., 2009) and model-based negative sampling approaches (Zhang et al., 2013). The heuristic negative sampling approaches set the sampling distribution according to some prior knowledge (Manotumruksa et al., 2017) or heuristic assumption (Rendle et al., 2009), while the model-based negative sampling approaches are based on the embeddings learned from models (Zhang et al., 2013). Since our proposed DNS scheme only leverages the learned embeddings, we classify it as a model-based approach.

Among different types of model-based negative sampling approaches, generative adversarial networks (GAN)-based approaches (Lian et al., 2020a; Park and Chang, 2019; Wang et al., 2017; Yu et al., 2020) have been shown to be effective. However, training an adversarial network is often time-consuming, making it hard to efficiently deliver precise recommendations to users (Bowles et al., 2018). Another model-based negative sampling approach, known as dynamic negative sampling (Ding et al., 2019; Wang et al., 2020), uses the embeddings learned during the dynamic training of a recommendation system. Similar to Zhang et al. (2013), our proposed scheme includes a dynamic negative sampling approach to update negative items. However, different from existing work (Wang et al., 2020; Zhang et al., 2013), which used dynamic negative sampling iteratively after each epoch, our DNS scheme is only activated after a number of epochs, to improve the efficiency of the dynamic negative sampling approaches. In addition, most of the existing works use *in-batch* negatives (Karpukhin et al., 2020; Yih et al., 2011), meaning that their negative samples are selected from each training batch instead of being selected from all available samples. This *in-batch* negative sampling requires lower computational cost but it is suboptimal in terms of the effectiveness compared with sampling the global negatives from all available samples (Xiong et al., 2020). Hence, differently from those negative sampling approaches using the less effective *in-batch* negatives (Karpukhin et al., 2020; Yih et al., 2011), our proposed DNS scheme leverages the *global* negative items, meaning that we search among all negative items of each user instead of searching in a single batch of data.

5.3 Methodology

In this section, we first define contrastive recommendation and its relationship with BPR (Section 5.3.1), followed by our dynamic negative sampling (DNS) approach (Section 5.3.2). Finally, we demonstrate how to incorporate our proposed scheme and objective function for a general recommender system (Section 5.3.3). To provide a clear overview of our proposed scheme, we use Figure 5.1 to illustrate the architecture of DNS in comparison with BPR (Rendle et al., 2009). The task definition and notations are the same as in Chapter 3.

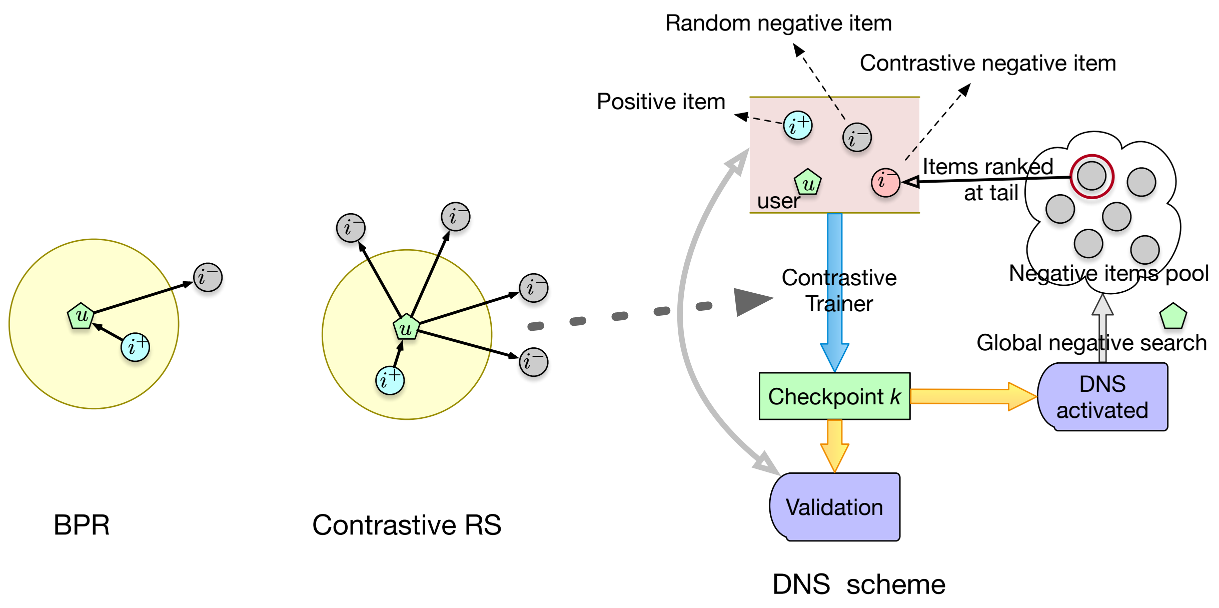


Figure 5.1: Comparison of our proposed DNS scheme with BPR.

5.3.1 Contrastive Recommendation

Contrastive recommenders (Liu et al., 2021b; Wu et al., 2021; Yu et al., 2022a) are designed to learn contrastive representations for users and items such that the distance between users and their positive items are minimised while the distance between users and their negative items are maximised. Among existing contrastive recommenders, InfoNCE is the most commonly adopted objective function (Chen et al., 2020c). In general, contrastive recommenders adapt the original InfoNCE loss (Equation (2.25)) by replacing the image samples with the user and positive/negative samples, where the adapted objective function is shown in the following equation:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp\left(\text{sim}(\mathbf{e}_u, \mathbf{e}_{i_u^+})/\tau\right)}{\exp\left(\text{sim}(\mathbf{e}_u, \mathbf{e}_{i_u^+})/\tau\right) + \sum_1^j \exp\left(\text{sim}(\mathbf{e}_u, \mathbf{e}_{i_u^-})/\tau\right)}, \tag{5.1}$$

where sim$(\cdot)$ is a similarity measure and τ is a constant denoting the *temperature parameter*; $\mathbf{e}_u$, $\mathbf{e}_{i_u^+}$ and $\mathbf{e}_{i_u^-}$ represent the embeddings of user u and his/her interacted and uninteracted with items, respectively.

In Equation (5.1), we note that there are j different negative samples. In the existing literature (Liu et al., 2021b; Zhou et al., 2021), such negative samples are obtained by leveraging different data augmentation methods introduced in Section 2.2.4. For example, graph perturbations have been used to sample negative samples from the multi-hop graph neighbourhood (Liu et al., 2021b). In addition, the node dropout, edge dropout, and random walk approaches have also been used to generate negative samples (Wu et al., 2021). Given that existing contrastive recommender systems generally follow a similar loss function, we can conclude that the approach used to generate the negative samples is the underlying factor in differentiating contrastive recommenders.

Indeed, we can bridge the BPR pairwise ranking approach (see Section 2.1.1.1) with contrastive recommenders by defining the objective function of BPR in a similar fashion. For example, given each training instance of BPR constructed as:

$$D_u := \{\langle i_u^+, i_u^- \rangle \,|\, i_u^+ \in \mathcal{I}_u^+ \wedge i_u^- \in \mathcal{I} \setminus \mathcal{I}_u^+\}, \tag{5.2}$$

a BPR recommender optimises the following objective function, which is similar to Equation (2.5):

$$\mathcal{L}_{\text{BPR}} = \sum_{(u,i^+,i^-)\in \boldsymbol{D}} -\log \frac{\exp(\mathbf{e}_u \cdot \mathbf{e}_{i_u^+})}{\exp(\mathbf{e}_u \cdot \mathbf{e}_{i_u^-}) + \exp(\mathbf{e}_u \cdot \mathbf{e}_{i_u^+})}, \tag{5.3}$$

where λ is the regularisation parameter; and θ represents all parameter embeddings in the trained model.

By comparing Equation (5.1) and Equation (5.3), we notice that there are two main differences: (1) BPR uses $\exp(\cdot)$ to model the interaction between users and items while the InfoNCE loss uses $\exp(\cdot)$ together with a similarity measure; (2) in the BPR setup, only a static randomly sampled negative item is incorporated for each training instance, while for contrastive recommenders, multiple negative items are incorporated. A detailed comparison between BPR with a contrastive recommender is illustrated in Figure 5.1. Since both approaches aim to repel negative items away from users and positive items, we can consider a contrastive recommender as a more generalised extension of BPR with additional negative items. The difference between contrastive recommenders and BPR is that contrastive recommenders use different data augmentation methods to create or search for more informative negative samples.

Interestingly, a recent study (Yu et al., 2022a) has theoretically demonstrated that the existing data augmentation methods, which are used by contrastive recommenders cannot provide robust performance improvements. In their study, Yu et al. (2022a) showed that the data augmentation methods, including node dropout and random walk, cannot significantly improve the performance of a contrastive recommender, with the variant without augmentations sometimes performing better. Although those data augmentation methods can effectively create different variants of embeddings (known as different views in contrastive learning) to serve as contrastive negatives, these variants are still rooted in the same user/item. Hence, the whole process can only be treated as static because they keep a constant sampling distribution across the training process.

Therefore, to further progress the development of contrastive recommenders, we propose to sample contrastive negative items dynamically. During the dynamic sampling process, we expect the model to gradually learn which negative items are more informative than random negative items. Thus, our contrastive negative items are updated accordingly during this learning process. In the following section, we formally define our proposed Dynamic Negative Sampling (DNS) scheme by detailing how to search for contrastive negative items and the subsequent loss function.

5.3.2 Dynamic Negative Sampling

As mentioned in Section 5.1, using the static approach (e.g., the one used in BPR) to randomly sample negative items might lead to an increase of false-negatives and can result in the exposure bias issue. We propose to use a dynamic approach instead of the typical static one and its variants (He et al., 2020; Manotumruksa et al., 2017).

The amount of each user u's negative items (i.e. $|\mathcal{I} \setminus \mathcal{I}_u^+|$) is large since $|\mathcal{I}| \gg |\mathcal{I}_u^+|$. This shows why a sampling strategy is required to handle the huge amount of negative items because otherwise the size of the training set $\boldsymbol{D}$ will become a bottleneck. Although the random sampling strategy adopted by the pairwise ranking approach is simple and generally effective, these randomly sampled items can accidentally hurt the recommendation performance. For example, those randomly sampled negative items might actually be positive items in the test set, since the sampling is suppose to be conducted solely based on the training set $\boldsymbol{D}$. These false-negative items will likely degrade the recommendation performance. In addition, users might be paired with those negative items that have not been exposed to them. This might introduce a bias against unpopular items by assuming that they are not preferred by users.

To alleviate the aforementioned issues of the random negative sampling, we devise a novel dynamic negative sampling scheme, abbreviated as DNS. Our DNS scheme can sample contrastive negative items and reduce the false-negatives simultaneously. Specifically, the DNS scheme can automatically update negative items at each checkpoint, solely relying on our proposed similarity-based method defined by Equation (5.4), without any side information about the users and items.

Following existing contrastive models (Chen et al., 2020c,d; Robinson et al., 2020; Xiong et al., 2020), we use the cosine similarity to measure the similarity between each user and his/her negative items. In the field of computer vision and natural language processing, a negative sample is usually defined as a contrastive negative when it has a high similarity score with the targeted image/sentence. This is a commonly used definition for most of the contrastive models (Chen et al., 2020c; Robinson et al., 2020). Distinctively, our DNS scheme defines contrastive negative items as those items, which have lower similarity scores with the target user because items with higher similarity scores are likely to be the items of the user's interest. As a consequence, using the exact definition of hard negatives from other fields might actually increase the false-negatives in a recommender system framework instead of benefiting the representation learning and the prediction performance. Our choice of dissimilar negative items will be further justified in the subsequent experiments (see Section 5.5.1). We use the following equation to search for the contrastive negative items:

$$i_u^{-\prime} \in f_{\text{topk}}\left(-\cos(\mathbf{e}_u, \mathbf{E}_{I_u^-})\right), \tag{5.4}$$

where $i_u^{-\prime}$ denotes the contrastive negative item for user u; $f_{\text{topk}}(\cdot)$ means that k elements with higher similarity scores will be selected ; $\cos(\cdot)$ is the cosine similarity; and $\mathbf{E}_{I_u^-}$ is the embedding table containing the embeddings of all negative items for user u.

After sampling k contrastive negative items for each user, our DNS scheme will randomly assign one or more $i_u^{-\prime}$ for each user. Note that if a user has m positive items in the training set $\boldsymbol{D}$, m contrastive negative items $i_u^{-\prime}$ will be assigned to this user correspondingly. Therefore, the new training set $\boldsymbol{D}'$ will have the same number of training instances as the original set $\boldsymbol{D}$ but with the sampled negative items in each instance.

For training the recommender, we propose to leverage both statically and dynamically sampled negative items to combine the benefit of both the static and dynamic negatives during contrastive learning. In particular, our contrastive recommender aims to optimise the following objective function:

$$\mathcal{L}_{\text{con}} = \sum_{(u,i^+,i^-,i^{-\prime})\in\boldsymbol{D}'} -\log \frac{f_{\text{s}}(\mathbf{e}_u, \mathbf{e}_{i_u^+})}{f_{\text{s}}(\mathbf{e}_u, \mathbf{e}_{i_u^-}) + f_{\text{s}}(\mathbf{e}_u, \mathbf{e}_{i_u^{-\prime}}) + f_{\text{s}}(\mathbf{e}_u, \mathbf{e}_{i_u^+})}, \tag{5.5}$$

where $f_{\text{s}} = \exp(\cos(\cdot)/\tau)$ and τ is a constant denoting the *temperature parameter* (Oord et al., 2018); the cosine similarity $\cos(\cdot)$ can also be replaced by other similarity measures but as mentioned below Equation (5.4), we use $\cos(\cdot)$ following the default setup of contrastive learning; $\mathbf{e}_u$, $\mathbf{e}_{i_u^+}$ and $\mathbf{e}_{i_u^-}$ have been defined under Equation (5.1), while $\mathbf{e}_{i_u^{-\prime}}$ is the embedding of a contrastive item.

By using the contrastive objective function $\mathcal{L}_{\text{con}}$, we aim to repel not only the static negative items but also the contrastive negative items away from each user and his/her corresponding positive items in the latent space. Indeed, by comparing Equation (5.5) and Equation (5.3), we can also interpret $\mathcal{L}_{\text{con}}$ as an extended version of $\mathcal{L}_{\text{BPR}}$, where $\mathcal{L}_{\text{BPR}}$ is dedicated to leverage the single negative case.

Since retrieving contrastive negative items will lead to a lower efficiency, we only update contrastive negative items iteratively after a fixed number of epochs to accelerate the overall training process. Specifically, we retrieve k contrastive negative items for each user at every checkpoint. Next, these updated contrastive negative items will be stored and used during the following n epochs till the next checkpoint. In addition, contrastive negative items from the previous checkpoints are dropped for the memory-efficiency reason. Indeed, they are no longer required.

5.3.3 Model Prediction

When incorporating the proposed DNS scheme into a general model, an existing recommender system still retains its original model initialisation and no additional data pre-processing is required. Our DNS scheme is activated only at each checkpoint during the training process according to a pre-defined update interval, which is an integer. Therefore, our proposed scheme is straightforward enough to be incorporated by general recommender systems. Furthermore, our contrastive objective function $\mathcal{L}_{\text{con}}$ does not change the prediction layer, hence no extra modification is required for the recommendation generation. In particular, for our used general recommender systems, namely BPR (Rendle et al., 2009), LightGCN (He et al., 2020), SGL (Wu et al., 2021) and SimGCL (Yu et al., 2022a), we still use the dot product (BPR) or the dot product with their corresponding activation layers (LightGCN, SGL and SimGCL) as their prediction functions. We have described the BPR, LightGCN and SGL models in Section 3.4.1, while SimGCL was introduced in Section 4.5.3. In particular, we use SGL and SimGCL to replace NGCF (Wang et al., 2019c) and NCF (He et al., 2017) due to their better performance, as shown in Table 4.2.

5.3.4 Entropy Analysis

As mentioned in Section 5.1, we aim to show the benefit of our proposed scheme from the perspective of information theory. In information theory, entropy is the measure of a variable's average level of uncertainty or its informational value (Carter, 2007). In fact, entropy is strongly related to many critical concepts in deep learning. For example, the cross-entropy loss (Martinez and Stiefelhagen, 2018; Zhang and Sabuncu, 2018) is widely used for classification tasks. Moreover, information bottleneck (Tishby et al., 2000), mutual information (Kraskov et al., 2004) and Kullback-Leibler divergence (Hershey and Olsen, 2007) can all be considered as a comparison of entropy between the observed and predicted distributions.

In this section, we aim to determine the informational value of the negative items sampled by DNS for a recommender system. In particular, to show the added-value of our DNS scheme, we need to compute such an informational value and compare the value of DNS-sampled negative items against randomly sampled negative items. Therefore, inspired by information theory, we leverage the entropy measure. Since the embeddings of users and items are of the same length and trained in the same latent space, we can roughly assume that they are sampled from the same distribution. Thus, to measure the entropy of a certain group of items and users, we propose to merge the embeddings of these users and items and compute the entropy as follows:

$$H = -\frac{\text{sum}\Big(\sigma(\text{concat}(\mathbf{E}_u, \mathbf{E}_i)) \cdot \log(\sigma(\text{concat}(\mathbf{E}_u, \mathbf{E}_i)))\Big)}{d}, \tag{5.6}$$

where $\sigma(\cdot)$ is the softmax function used to normalise all embeddings to positive values to avoid computing a $\log$ of negative values; concat$(\cdot)$ stands for the concatenation operation of embeddings; and d is the latent dimension of the users and items' embeddings i.e. $\mathbf{E}_u$ and $\mathbf{E}_i$, respectively.

Using Equation (5.6), we can compute the entropy of the users and items' embeddings. Our proposed entropy measure aims to approximate the information or uncertainty encapsulated in the sampled negative items. To validate our measure, we also examine it along with the performances of the examined models on the validation set during training. In principle, when the entropy measure saturates, the validation performance should also stabilise because there is no more information to be learned.

Table 5.1: Statistics of the Amazon-Instant-Video dataset.

	Amazon
Users	21,596
Items	11,167
Interactions	2,092,329
Density	0.868%

5.4 Datasets and Experimental Setup

We use three public datasets, i.e. Yelp, MovieLens-1M and Amazon-Instant-Video[1], to evaluate our proposed DNS scheme, where Yelp and MovieLens-1M have been used before and their statistics can be found in Table 3.2 and Table 4.1, respectively. Amazon-Instant-Video is the Amazon review dataset containing the interactions between the users and instant videos. We choose this new dataset to examine whether our proposed scheme can be applied to a larger dataset, where the Amazon-Instant-Video dataset is almost $\times 2$ larger than the largest dataset (i.e., Yelp) used in Section 3.4 and Section 4.5. Table 5.1 provides the statistics of the Amazon-Instant-Video dataset. For the rest of the chapter, we use 'MovieLens' and 'Amazon' as short-hands for 'MovieLens-1M' and 'Amazon-Instant-Video', respectively. We aim to answer the following research questions:

[1] https://jmcauley.ucsd.edu/data/amazon/

RQ5.1 Do dissimilar negative items lead to a lower false-negative rate and enhanced recommendation performances?

RQ5.2 Can our proposed DNS scheme generally improve the recommendation performances for existing embedding-based recommender systems and outperform state-of-the-art negative sampling-enhanced approaches?

RQ5.3 How do the introduced hyperparameters affect the effectiveness of DNS?

RQ5.4 Can negative items sampled by DNS provide a higher information gain than those random ones according to our proposed entropy measure?

In the next sections, we present four general baselines and two state-of-the-art NS-enhanced approaches that we use to evaluate the performance of our proposed DNS scheme, followed by the used evaluation methodology, and the corresponding experimental setup.

5.4.1 Baselines

We evaluate the effectiveness of our DNS scheme on four general recommendation models and two NS-enhanced approaches. Specifically, four general recommenders include BPR, LightGCN, SGL and SimGCL, where BPR, LightGCN and SGL were introduced and used as baselines in Section 3.4.1. In addition, details of SimGCL (Yu et al., 2022a) can be found in Section 4.5.3.

Furthermore, we compare DNS with two other state-of-the-art NS-enhanced approaches to demonstrate the superiority of our proposed scheme:

- **MixGCF** (Huang et al., 2021b): MixGCF is a recent general negative sampling (NS) approach that can be directly used to train GNN-based recommender systems. It creates synthetic negative samples through the idea of neighbourhood mixing. Among all of its variants combined with LightGCN, NGCF and PinSage, the LightGCN variant consistently outperforms the other two variants on all of the benchmark datasets with a large margin (at least 40% in terms of the recall@20 measure). Therefore, we only compare DNS with the best variant (i.e. MixGCF+LightGCN) and we use MixGCF to denote this best variant for the rest of the chapter.

- **PRIS** (Lian et al., 2020a): Personalised Ranking loss based on Importance Sampling (PRIS) is an effective recommender system for items recommendation. It incorporates a novel negative sampling approach based on importance sampling, where PRIS can assign larger weights to more informative negative items.

5.4.2 Evaluation Methodology

Following the commonly used evaluation setup for these datasets (He et al., 2020; Wu et al., 2021), we split each dataset into a training set, validation set and test set with a ratio of 8:1:1. Similar to Section 3.4.2, we construct 10 different testing sets with different sampled negative items for each dataset using different random seeds. This allows to reduce the evaluation bias on some specific testing negatives (Krichene and Rendle, 2022). For a fair comparison, we keep the batch size as 1000 for all models and each model is trained by up-to 500 epochs unless it is early-stopped when the validation performance does not increase for 50 successive epochs. To tune all hyperparameters, we apply a grid search, where the learning rate is tuned in $\{10^{-2}, 10^{-3}, 10^{-4}\}$; the latent dimension in $\{32, 64, 128\}$ and the L_2 normalisation in $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.

To answer **RQ5.1**, recall that in Section 5.3.2 we defined the dissimilar negative items to be the contrastive negative items. Similarly, we can also follow ANCE (Xiong et al., 2020) and other contrastive models to define a similar negative item as a contrastive sample, which is commonly known as the hard negative. In addition, we also include a naive baseline sampler that dynamically samples random negative items. To justify our choice of dissimilar negatives, we compare the recommendation performance of DNS when adopting different types of negative items, namely dynamic dissimilar negatives, dynamic similar negatives and dynamic random negatives. Furthermore, we calculate the corresponding false-negative rates for all the aforementioned three cases, to further illustrate why choosing the dynamic dissimilar negatives is more appropriate in the recommendation scenario. The false-negative rate is computed by dividing the number of training instances whose negative items are false negatives according to the testing set, by the total number of training instances.

To answer **RQ5.2**, we need to validate the effectiveness of our proposed DNS scheme over two groups of comparative experiments, namely comparing our DNS-enhanced recommenders with their corresponding baseline recommenders and comparing our DNS-enhanced recommenders with other NS-enhanced recommenders. Three metrics (NDCG@10, Recall@10 and MAP@10) are applied to evaluate the effectiveness of all evaluated models and baselines. Details of these metrics have been introduced in Section 2.1.2.2.

Note that our DNS scheme contains two new hyperparameters, i.e. n and k, which are the update interval and the number of retrieved negative items at each checkpoint, respectively. To answer **RQ5.3**, we tune n and k from $\{1, 5, 10, 20, 30, 40, 50\}$ and $\{10, 50, 100, 150, 200, 250, 300\}$, respectively. We evaluate the recommendation performance of the BPR model with our DNS scheme. When $n = 1$, the trained model will have its negative items refreshed after every epoch and we denote this case as the exhaustive update.

Finally, to answer **RQ5.4**, after each training epoch, we calculate the entropy measure of BPR and BPR_{DNS}, respectively. For a fair comparison, we explicitly measure the entropy of contrastive negative items only for BPR_{DNS} to exclude the possible bias of having additional negative items over BPR. Meanwhile, we report the NDCG@10 performance of these two models in the same plot to demonstrate the validity of our proposed measure. For the latter case, we disable the early-stopping strategy to illustrate the whole training curve.

For both **RQ5.3** and **RQ5.4**, we only show the results of BPR and BPR$_{\text{DNS}}$ but we note that the experiments with other baselines result in the same conclusions. Moreover, for **RQ5.4**, we need to report the validation performance iteratively after every epoch, which is time-consuming if deep neural models are used. Therefore, we also choose these two models for both efficiency reason and conciseness, since they sufficiently illustrate the added-value of our proposed DNS scheme.

5.5 Results Analysis

Table 5.2: Performances and false-negative rates of BPR when using a static random sampler and three dynamic samplers on the three used datasets. The NDCG metric is computed at rank cutoff 10. F-N rate denotes the false-negative rate and * denotes a significant difference between the performance of the dynamic dissimilar sampler and that of other samplers according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$. The best result and the lowest F-N rate are highlighted in bold.

Sampler	**MovieLens**		**Yelp**		**Amazon**	
	F-N Rate (%)	NDCG	F-N (%) Rate	NDCG	F-N Rate (%)	NDCG
Static random sampler	0.5139	0.2805*	0.1289	0.0887*	0.6001	0.0834*
Dynamic random sampler	0.5125	0.2910*	0.1201	0.0889*	0.604	0.0820*
Dynamic similar sampler	1.0562	0.2013*	0.2884	0.0656*	1.1796	0.0633*
Dynamic dissimilar sampler	**0.4853**	**0.3394**	**0.1033**	**0.1513**	**0.5787**	**0.1092**

In this section, we report the experimental results and answer our four research questions.

5.5.1 Contrastive Negative Items

In order to answer **RQ5.1**, we compare three possible dynamic negative samplers with the static random sampler i.e., a plain BPR on the three used datasets. First, from Table 5.2, we find that there is no noticeable difference between the static random negative sampler and the dynamic random negative sampler in terms of the false-negative rate and NDCG@10. This suggests that solely refreshing negatives will not help in decreasing the false-negative rate and improving the recommendation performance. Second, by comparing the static random sampler and the dynamic similar sampler, we observe a significant performance decrease when using the similar negative sampler across all datasets. Furthermore, the false-negative rate is almost doubled on each dataset when the dynamic similar negative sampler is used, compared with the static BPR. This is because the dynamic similar negative sampler always selects highly ranked negative items during the training process. Although these highly ranked negative items might be the typical hard negatives, they are more likely to be the users' interacted with items, which leads to a higher false-negative rate. Alternatively, the dynamic similar negative sampler will sample some interesting items to the users that have not been presented to them yet, leading to an exposure bias. Therefore, as argued in Section 5.3.2, sampling negative items with high similarity scores is not appropriate for recommender systems and it can hurt the recommendation performance by boosting the false-negative rate. On the contrary, the dynamic dissimilar negative sampler, which is used by DNS can consistently and significantly outperform all other negative samplers across all used datasets with a lower false-negative rate.

As a summary, our reported results suggest that contrastive negative items should be defined as those negative items with lower similarity scores with users, which indeed corroborates our choice of dynamic dissimilar negative sampler for DNS as discussed in Section 5.3.2. In answer to **RQ5.1**, dissimilar negative items do lead to a lower false-negative rate and enhanced recommendation performances. Furthermore, the reported results guide us to consolidate our proposed thesis statement by revealing that the contrastive negative sampling approach should select dissimilar negative items.

Table 5.3: Performances of two advanced NS-based baselines and four general recommenders with their DNS variants on the three used datasets. All metrics are computed at rank cutoff at 10. The best result is highlighted in bold and the largest percentage improvement is underlined; * denotes a significant difference between the performance of a DNS variant and that of the baseline, while † denotes a significant difference among the four DNS-based models, according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$.

	MovieLens			**Yelp**			**Amazon**		
	NDCG	Recall	MAP	NDCG	Recall	MAP	NDCG	Recall	MAP
PRIS	0.3298*	0.2301*	0.1105*	0.2205*	0.2897*	0.1214*	0.2578*	0.3806*	0.1970*
MixGCF	0.3300*	0.2351*	0.1179*	0.2100*	0.2705*	0.1104*	0.2308*	0.3698*	0.1806*
BPR	0.2805*	0.1917*	0.0886*	0.1230*	0.1683*	0.0600*	0.0834*	0.1632*	0.0526*
BPR$_{\text{DNS}}$	0.3394†	0.2281†	0.1090†	0.1513†	0.2019†	0.0726†	0.1092†	0.2089†	0.0678†
Improvement	20.9%	19.0%	23.0%	23.0%	20.1%	21.0%	30.9%	28.0%	28.9%
LightGCN	0.3425*	0.2504*	0.1252*	0.2285*	0.3067*	0.1255*	0.2599*	0.4111*	0.1980*
LightGCN$_{\text{DNS}}$	0.3938†	0.2905†	0.1488†	0.2696†	0.3690†	**0.1472**†	0.3119†	0.4805†	**0.2356**†
Improvement	15.0%	16.1%	18.8%	18.0%	20.3%	17.3%	20.0%	16.9%	18.9%
SGL	0.3575*	0.2716*	0.1366*	0.2198*	0.2867*	0.1204*	0.2664*	0.3983*	0.2092*
SGL$_{\text{DNS}}$	**0.3942**†	**0.3042**†	**0.1537**†	0.2528†	0.3340†	0.1360†	0.3143†	0.4680†	0.2489†
Improvement	10.3%	12.0%	12.5%	15.1%	16.5%	12.9%	18.1%	17.5%	19.0%
SimGCL	0.3505*	0.2689*	0.1301*	0.2328*	0.3278*	0.1298*	0.2754*	0.4229*	0.2187*
SimGCL$_{\text{DNS}}$	0.3870†	0.2998†	0.1496†	**0.2707**†	**0.3731**†	0.1467†	**0.3175**†	**0.4910**†	**0.2524**†
Improvement	10.4%	11.5%	15.0%	16.3%	13.5%	13.0%	15.3%	16.1%	15.4%

5.5.2 Effectiveness of DNS

Table 5.3 shows that our four used baselines are improved with large performance margins compared to their DNS variants across all used metrics and datasets. This shows that our proposed DNS scheme can remarkably improve the recommendation performance of the general recommender systems. Noticeably, our BPR$_{\text{DNS}}$ model can even outperform both NS-enhanced approaches (MixGCF, PRIS) on the MovieLens dataset. This suggests that the performance of a classic non-neural embedding-based recommender system can be largely boosted and can even outperform state-of-the-art neural recommenders when negative items of high quality are provided. Among all DNS-based models, the SimGCL$_{\text{DNS}}$ variant is shown to achieve the best performance for the majority of cases. We also find that a better performing general recommender usually gives a better DNS variant. For example, SGL performs the best among all general recommenders on the MovieLens dataset, while SimGCL performs the best on the Yelp and Amazon datasets. As a result, their DNS variants achieve the overall best performances on the various datasets. This is expected because a better performing general recommender can also provide negative items that are unlikely to be interacted with by users with a higher probability, compared with those worse performing baseline recommenders.

Overall, in answer to **RQ5.2**, our results show that our proposed DNS scheme can generally improve the recommendation performances of four embedding-based general recommenders with a large margin. Meanwhile, most of our DNS-based recommenders can outperform strong baselines including MixGCF and PRIS, which use other advanced negative sampling approaches. Therefore, our results support our hypothesis in the proposed thesis statement that graph contrastive learning can be leveraged to enhance the effectiveness of recommender systems.

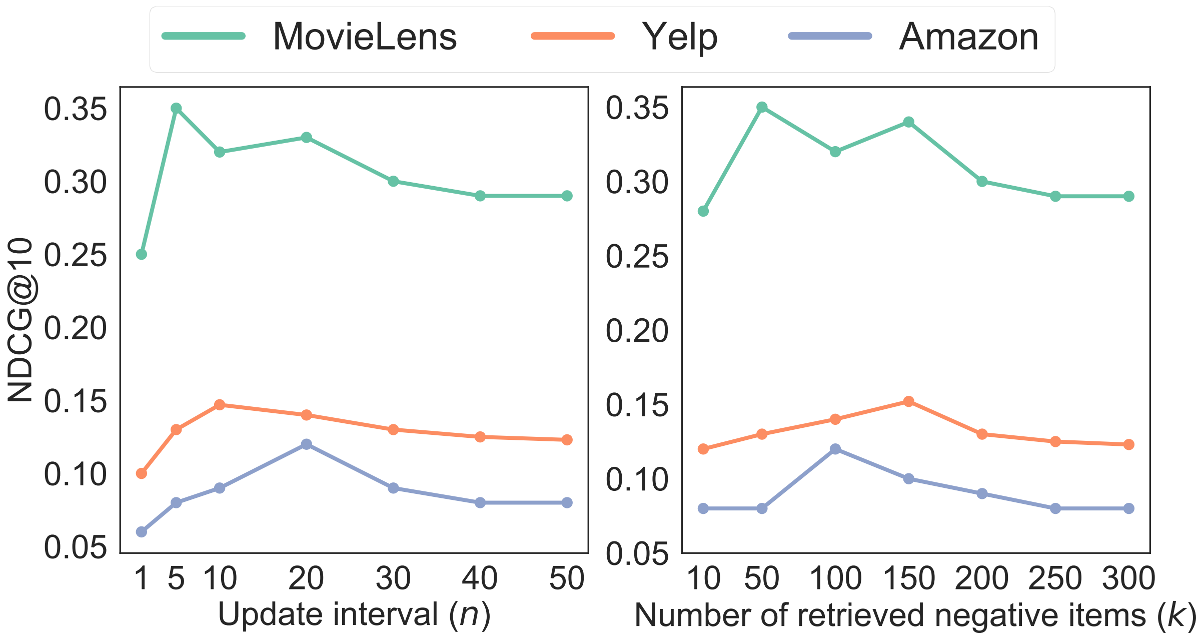


Figure 5.2: NDCG@10 performances of the BPR$_{\text{DNS}}$ model over (a) different update intervals (n) and (b) different number of retrieved negative items (k) on three used datasets.

5.5.3 Hyperparameter Analysis

To answer **RQ5.3**, recall that our proposed DNS scheme introduces two hyperparameters, namely the update interval (n) and the number of retrieved negative items (k), where n defines how frequently DNS will be activated to update the negative items and k controls the number of negative items to be retrieved for each user. We plot these two hyperparameters versus the NDCG@10 performance of the BPR$_{\text{DNS}}$ model on three used datasets, where n varies from $\{1, 5, 10, 20, 30, 40, 50\}$ and k is tuned within $\{10, 50, 100, 150, 200, 250, 300\}$. We choose the maximum of n as 50 because as mentioned in Section 5.4.2, we train all models with an early-stopping strategy, where each model will be early-stopped if its validation performance is not improved within 50 successive epochs. Hence, DNS might not be activated at all when n is larger than 50. k is tuned with a maximum of 300 because when the number of retrieved negative items is too large, these sampled negative items will tend to be the same as the random items. From Figure 5.2 (a), we can see that at the exhaustive update i.e., n=1, BPR$_{\text{DNS}}$ will underperfom the plain BPR model for all datasets (NDCG@10=0.2479, 0.0987 and 0.0611 for the MovieLens, Yelp and Amazon datasets, respectively). This indicates that updating negative items too frequently will in turn hurt the recommendation performance of the general recommender. Indeed, training recommender systems on a relatively large dataset almost always takes more than 1 epoch to converge, hence this performance decline is due to the updating of the negative items with a high frequency. We also find that BPR$_{\text{DNS}}$ peaks at n= 5, 10 and 20 for the MovieLens, Yelp and Amazon datasets, respectively. BPR$_{\text{DNS}}$ peaks with a large n value on the Amazon dataset because the Amazon dataset has far more interactions than the other two datasets (e.g., about 4.5$\times$ size of the Yelp dataset), hence the longer convergence period is also expected, which means a frequent update is not desired. From Figure 5.2 (b), we find that sampling between 50 to 150 contrastive negative items help the BPR$_{\text{DNS}}$ model to achieve the best performance on the three used datasets. When only 10 contrastive negative items are sampled, DNS is likely to cause repeated samples. For example, if a user has more than 10 interactions in the training set, repeated samples are unavoidable since only 10 contrastive negative items are selected. These repeated samples might degrade the recommendation performance. On the other hand, if a large amount of negative items are sampled, these sampled items tend to become as informative as those randomly sampled negative items. For example, when the top 1,000 negative items are sampled by DNS, the probability that each item is selected is $\frac{1}{1000} = 0.001$. This is not substantially larger than randomly sampling negative items from the MovieLens dataset (3,533 items in total) if we exclude all positive items. Furthermore, the effectiveness of DNS lies in that it can explore the users' actual negative items and estimate what these users dislike. Therefore, sampling from a large pool of negative items is not far from a random guess since not many users have a large amount of items they dislike.

Our hyperparameter analysis consolidates our proposed thesis statement by showing that using graph contrastive learning to sample contrastive negative items can consistently improve the effectiveness of a general ranking-based recommender systems i.e., BPR.

5.5.4 Entropy Analysis

To answer **RQ5.4**, we plot the entropy measure of the contrastive negative items against the random negative items together with the performances of BPR and BPR$_{\text{DNS}}$ on the validation set. First, we want to examine the validity of our proposed measure. We hypothesise that entropy can approximate the amount of information contained in the embeddings. From Figure 5.3, we can observe that when the entropy of the two models stabilises after epoch 50, the validation performances also stop fluctuating. The proposed measure is not an instant indicator because although we can approximate the level of information, it still takes a few more epochs to learn the information thoroughly. To summarise, our proposed entropy measure can generally approximate the level of information contained in the embeddings. After examining the validity of our proposed measure, we can compare the entropy of BPR and BPR$_{\text{DNS}}$ shown as green and orange lines, respectively, in Figure 5.3. The figure clearly shows that the contrastive negative items provided by our DNS scheme continuously contain more information than the random negative items when using the entropy measure. Furthermore, the smoother entropy line of BPR$_{\text{DNS}}$ (green) indicates that our DNS scheme also provides a more constant information input, which is beneficial for a faster convergence, as suggested by Zhao et al. (2021).

In answer to **RQ5.4**, we can conclude that our proposed entropy measure can generally approximate the level of information contained in the embeddings. Finally, our results show that the contrastive negative items sampled by our proposed DNS scheme indeed do provide a higher information gain than those random negative items. Therefore, this entropy analysis helps us to consolidate our hypothesis in the thesis statement by demonstrating that DNS can provide more informative negative items indicated by the information gain.

5.6 Conclusions

In this chapter, we proposed DNS, a dynamic negative sampling scheme for contrastive recommender systems, which can be deployed in many existing general recommender systems to improve their effectiveness. In particular, DNS can dynamically sample contrastive negative items at each checkpoint solely based on the cosine similarity scores between the users and items' embeddings without using any side information. Besides, we proposed a contrastive objective function to leverage these sampled negative items. This objective function is able to enhance the effectiveness of general recommenders. To answer **RQ5.1**, we first examined different types of negative samplers in Section 5.5.1, where results showed that the dynamic dissimilar sampler is the best-performing one. In order to answer **RQ5.2**, we conducted an extensive evaluation of the DNS-based models on three datasets in comparison to six baselines showing that our proposed

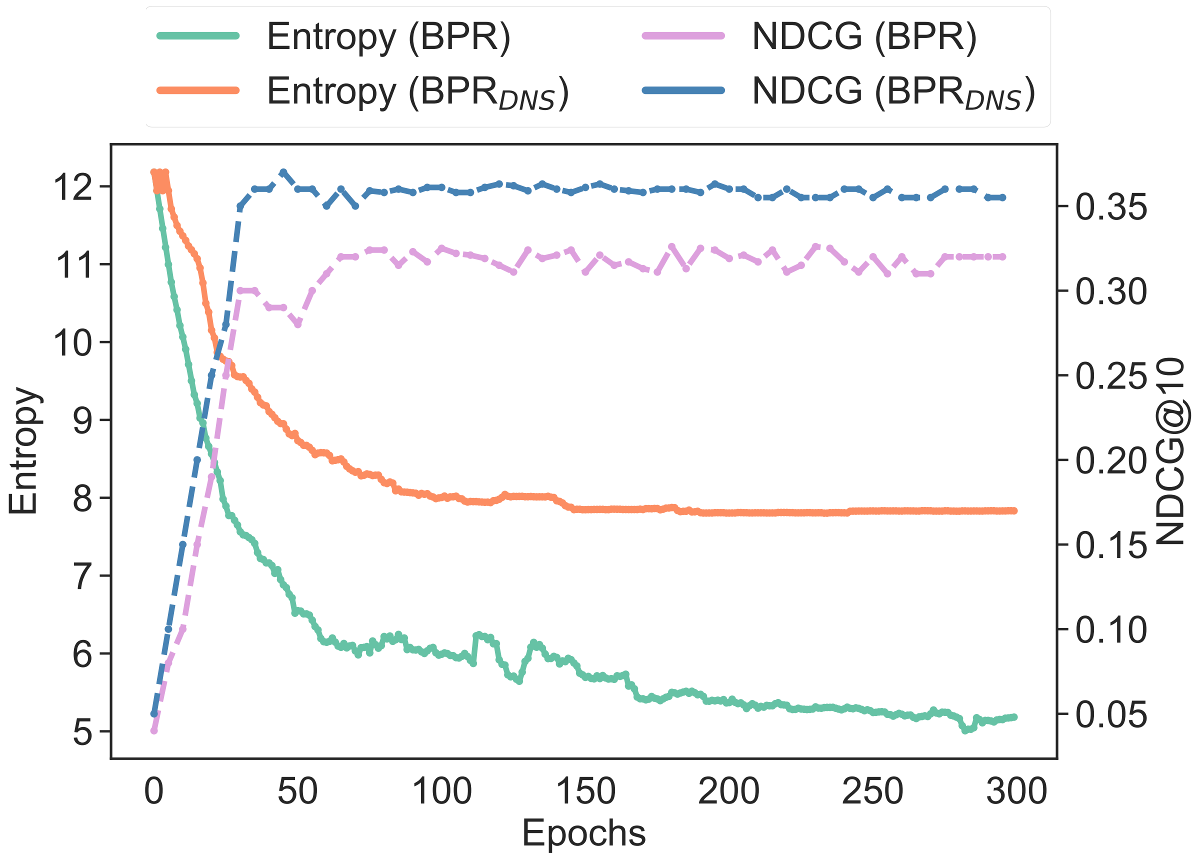


Figure 5.3: Entropy comparisons of BPR$_{\text{DNS}}$ and BPR on the MovieLens-1M dataset.

scheme can generally improve the performance of four general recommenders and can significantly outperform two strong baselines using different advanced negative sampling approaches. Specifically, our DNS scheme can improve the performance of a general recommender by up-to 30.9% in terms of the NDCG@10 metric (see Table 5.3). Afterwards, we conducted a detailed hyperparameter analysis in Section 5.5.3 to examine the effects of our introduced hyperparameters (**RQ5.3**). In Section 5.5.4, we found that using DNS to provide contrastive negative items can bring a continuously higher and more constant information gain (**RQ5.4**).

In summary, we have validated the hypothesis of our proposed thesis statement in Section 1.2, namely that graph contrastive learning can enhance the effectiveness of graph-based recommender systems. The contribution of this chapter can be differentiated from the contributions of Chapters 3 & 4 because DNS improves the recommendation effectiveness by leveraging a more effective negative sampling approach while the previous two chapters focused on how to effectively incorporate the side information. The next chapter will still centre on graph contrastive learning but it will also concentrate more on sampling both pseudo-positive and negative samples simultaneously in order to enhance the efficiency of graph-based recommender systems. Furthermore, in the next chapter, we will additionally focus on the efficiency of a graph-based recommender system .

Chapter 6

Contrastive Multiple Sampling for Graph Recommendations

6.1 Introduction

In Chapter 5, we proposed a dynamic negative sampling approach that effectively samples negative items to be paired with a user and his/her positive items. The extensive experiments in Section 5.5.2 and the entropy-based analysis in Section 5.5.4 have demonstrated that our proposed approach can generally improve the recommendation performance of 4 ranking-based general recommenders. In this chapter, we further explore how to use graph contrastive learning to improve the efficiency of graph-based recommender systems. In particular, we are interested in whether positive samples can benefit graph-based recommender systems.

In an undirected graph $\mathcal{G}$ as defined in Section 2.2.1, a node n linked with a node v is defined as the 1-hop neighbour of node v. At the same time, the 1-hop neighbours of node n excluding node v, are defined as the 2-hop neighbours of node v. Similarly, the n-hop neighbours of a node are defined as the 1-hop neighbours of the $(n-1)$-hop neighbours of the targeted node. In particularly, those neighbours including the 1-hop and multi-hop neighbours, are considered as positive samples of the target node in the scenario of graph representation learning (Defferrard et al., 2016). Indeed, graph neural networks benefit from the message passing mechanism introduced in Section 2.2.1. Motivated by this graph-based intuition, we propose to consider the multi-hop neighbours of the users and items in an interaction matrix as positive samples in the recommendation scenario. In particular, we define these multi-hop neighbours as pseudo-positive samples to differentiate from the real positive samples i.e., the positive items as indicated by the users. Therefore, our research aim in this chapter is to propose a more effective sampling approach that samples both pseudo-positive and negative samples to enhance the recommendation performance. In particular, we focus on alleviating the low-efficiency issue of the graph-based recommender systems mentioned in Section 1.2. Moreover, in this chapter, we will propose a multiple sampler that reuses the dynamic negative sampling method proposed in

Chapter 5.

As explained in Section 2.2.1, the advantages of applying a GNN, especially multi-layer or heterogeneous GNNs, for the recommendation task lie in two pivotal functions, namely *neighbourhood aggregation* and *message passing*. By applying a neighbourhood aggregation function, the interactive information between users and items can be captured in their representations through message passing over the edges of a user-item bipartite graph. In addition, stacking multiple layers of GNNs can help recommender systems pass a message over multi-hop neighbours. However, the neighbourhood aggregation and message passing of the existing graph-based recommender systems heavily rely on the pre-computed adjacency matrix $\mathbf{A}$ of the user-item bipartite graph (see Section 2.2), which is time- and memory-consuming to compute. Although $\mathbf{A}$ is normally sparse, many epochs of iterative matrix multiplications and gradient updates over the concatenation of the users and items' embeddings is still inefficient, especially when multiple stacked layers of GNNs are applied (He et al., 2020; Wang et al., 2019c) We argue that the inefficiency of the existing graph-based recommender systems is mainly caused by the matrix multiplications over unnecessary neighbours. Indeed, some frequently interacted items would be aggregated to most users, causing more time- and memory-consumption. Given that the effectiveness gained by the graph-based recommender systems is attributed to neighbourhood aggregation and message passing (Gao et al., 2021), the challenge here is to exploit a more efficient manner to perform similar functionalities without using the full neighbourhood graph convolution.

In this chapter, we propose a Multilayer Perceptron (MLP) based Contrastive Graph Recommender, abbreviated as MLP-CGRec, which uses an MLP-Mixer (Tolstikhin et al., 2021) to encode the users and items' representations and to conduct efficient contrastive learning. The key idea of our approach is that, instead of aggregating and passing a message over all the neighbours, we sample each user and item's neighbours to form each pseudo-positive pair. Our proposed MLP-CGRec offers multiple pseudo-positive samples, which will lead to an unbalanced number of pseudo-positive and negative items because the (user, positive item, negative item) triplet will be expanded by these pseudo-positive samples. Instead of sampling more random negative items, we reuse the DNS method proposed in Chapter 5, which uses an approximate nearest neighbour search method to efficiently sample an equal amount of contrastive negative items to rebalance learning.

To summarise, this chapter makes the following contributions

- We incorporate a contrastive loss and a novel graph sampling method to simplify the *neighbourhood aggregation* and *message passing* of graph-based recommender systems.

- We employ an efficient MLP-based learning algorithm to enhance the expressive power and recommendation accuracy.

- We conduct extensive experiments on three public datasets, and show that our proposed MLP-

CGRec can achieve a high efficiency in terms of both the memory and time consumption compared with state-of-the-art graph-based recommender systems without a significant loss of effectiveness.

6.2 Multilayer Perceptron-based Models

This section briefly introduces recent progress in MLP-based models. In particular, we position our proposed model among the existing MLP-based recommender systems.

In principle, an MLP architecture could be regarded as a universal approximator (Kratsios, 2021; Pinkus, 1999). Recently, MLPs have attracted attention in the field of computer vision and other classification tasks with more general, efficient and effective architectures. For example, the Graph-MLP (Hu et al., 2021) method is proposed to perform similar functions with GNNs without explicit message passing. The high efficiency and competitive performance of Graph-MLP on the node classification task suggest that the classic message passing and neighbourhood aggregation in GNNs may not be necessary to convey information from neighbours. In addition, MLP-Mixer (Tolstikhin et al., 2021) is a novel all-MLP architecture for the image classification task. The simple design and competitive accuracy of MLP-Mixer bring us to pay attention to the trade-off between a simple design and an effective but complicated design based on convolution or self-attention. Other simple yet effective MLP-based models, including ResMLP (Touvron et al., 2021), gMLP (Liu et al., 2021a) have been proposed for the image classification task, motivating us to design an MLP-based model for the recommendation task. Although the MLP module has been leveraged in many recommender systems (Deng et al., 2019; Lu et al., 2018; Zhang et al., 2019b), it is limited as an alternative to the prediction layer, i.e., the dot product function and the idea is to use the MLP module to learn the non-linear relations between users and items. Different from existing MLP-based recommenders, our proposed MLP-CGRec model extends existing work from only modelling the relations between users and items to learning non-linear multi-hop relations from neighbours in order to learn enhanced representations of the users and items. By doing so, we can benefit from the simple architecture of MLP and its universal representational power.

6.3 Methodology

In this section, we present the details of our proposed MLP-CGRec model. Specifically, in Section 6.3.1, we detail how we sample pseudo-positive samples in an efficient manner. In Section 6.3.2, we present how to speed up the negative sampling approach. In Section 6.3.3 we present how to adapt the MLP-Mixer model for our recommendation task followed by our proposed objective function in Section 6.3.4. The task definition and preliminaries have already been introduced in Section 3.3.1.

6.3.1 Graph Neighbourhood Construction

From the user-item interaction matrix $\mathbf{R}$, we have introduced how to obtain its corresponding adjacency matrix $\mathbf{A}^{(M+N)\times(M+N)}$ in Section 2.2.1. The elements of $\mathbf{A}$ indicate whether the pairs of users and items are adjacent or not in the interaction graph. In the existing graph-based recommender systems, the multi-hop graph embeddings of users and items are usually computed by:

$$\mathbf{E}^{(l+1)} = (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{E}^{(l)} \tag{6.1}$$

where $\mathbf{E}^{(l)}$ contains the embeddings of all users and items at the l-th layer of the GNN, and $\mathbf{D}$ is the diagonal degree matrix of $\mathbf{A}$.

Starting from the initial embeddings $\mathbf{E}^{(0)}$, we compute Equation (6.1) for $(l-1)$ times to obtain the l-hop embeddings. Although the adjacency matrix and the diagonal degree matrix are both sparse, the users and items' embeddings are however dense and the whole process not only involves the matrix multiplication but also the iterative gradient update for the GNNs at each training epoch. This is why existing graph-based recommender systems have a relatively unsatisfactory training efficiency. To avoid this costly operation, we derive how to use the adjacency matrix to explore the l-hop neighbours by raising $\mathbf{A}$ to the power of l below with an induction process:

Lemma 6.3.1. *The $(i,j)^{th}$ entry $a_{ij}^{(l)}$ of $\mathbf{A}^l$, where $\mathbf{A}$ is the adjacency matrix of **R**, counts the number of walks of length l having the start and end nodes i and j, respectively.*

Proof. **Base case**: When l = 1, $\mathbf{A}^l$ = $\mathbf{A}$, and there is a walk between node i and j if and only if a_{ij}=1, thus the result holds. **Induction step**: Assume the proposition holds for $l = n$ and consider the case when $l = n+1$, i.e., the matrix $\mathbf{A}^{n+1}$ = $\mathbf{A}^n\mathbf{A}$. From the induction hypothesis, the value of (i,j) of the matrix $\mathbf{A}^n$ is the count of walks of length *n* from i to j. Now, the number of walks of length $n+1$ between node i and node j equals the number of walks of length n from node i to each node v, which is adjacent to node j. Therefore, the number of walks of length $n+1$ from node i to node j, i.e., the $(i,j)^{th}$ entry of $\mathbf{A}^{n+1}$, is the non-zero entries of $\mathbf{A}$, which corresponds exactly to the first neighbours of v. Thus the result holds for $l = n+1$ as well. **Conclusion**: By the principle of induction, Lemma 6.3.1 is true for all $l \in \mathbb{Z}^+$. □

From the induction above, we can draw the conclusion that the l-hop neighbours of $\mathbf{R}$ can be obtained by raising $\mathbf{A}$ to the power of l. For example, we can refer to the u-th row of the matrix A^2 to find user u's 2-hop neighbours. Therefore, by pre-defining how many neighbourhoods we would like to use, we can pre-compute the corresponding graph neighbourhood matrices. For the most commonly adopted case in graph-based recommender systems (He et al., 2020; Wang et al., 2019c), where 3-hop neighbours are included, we can pre-compute $\mathbf{A}^2$ and $\mathbf{A}^3$ of $\mathbf{R}$ for the subsequent sampling.

With the graph neighbourhood construction method, we obtain pseudo-positive samples in the recommendation scenario but without a balanced number of negative samples, which might lead to the over-repelling of some negative samples from other data points during learning (Chen et al., 2021). For example, the embedding of a negative item is usually repelled from the embeddings of the user and the corresponding positive item in the latent space. Now that more pseudo-positive samples are obtained, the embedding of this negative item will be repelled from the embeddings of user and multiple positive samples, which is unbalanced. Therefore, in the next section, we describe how to use the contrastive negative sampling approach in Section 6.3.2 to select negative samples to balance the number of pseudo-positive and negative samples.

6.3.2 Contrastive Negative Sampling

Given that typical training triplets (u, i^+, i^-) are extended to a training instance with multiple pseudo-positive samples, we need to obtain more negative samples to balance the number of positive and negative samples. Similar to how we get the randomly sampled negative item i.e., i^-, the most straightforward way is to randomly sample more negative items. However, inspired by the effectiveness of our dynamic negative sampling approach (DNS) proposed in Chapter 5, we decide to reuse this approach to sample negatives for MLP-CGRec.

To further increase the training efficiency, we adopt Faiss (Johnson et al., 2019), the fast neighbour search library, to reduce the search time. Given the number of users and items in the used datasets, we follow the advice in (Johnson et al., 2019) to use the flat indexes and the brute-force search [1] to avoid an accuracy loss. In particular, we select those negative items with lower similarity scores to avoid the false-negative items. As the recommendation effectiveness increases along the model training, we can become more confident that the target users are unlikely to prefer items with lower similarity scores, which will benefit the subsequent contrastive training.

6.3.3 MLP-based Learning

Although the graph neighbourhood construction method proposed in Section 6.3.1 and contrastive negative sampling (Section 6.3.2) are efficient, the model's expressive power is limited without a nonlinear activation. Recently, some advanced variants of Multilayer Perceptron (MLP) (Liu et al., 2021a; Tolstikhin et al., 2021; Touvron et al., 2021) have been proposed for more efficient deep model training. Inspired by their high efficiency and competitive accuracy on the image classification and natural language processing tasks, we propose to adopt MLP-Mixer to enhance our model's expressive power without a high computational cost. Our MLP module is defined as:

$$\mathbf{E}_{n+1} = \mathbf{E}_n + \mathbf{W}_2\sigma\Big(\mathbf{W}_1\mathrm{LayerNorm}(\mathbf{E}_n)\Big) \tag{6.2}$$

[1] Using flat indexes and the brute-force search is mathematically the same with the commonly used exact search. However, the overall search process is accelerated by optimising the General Matrix Multiply (GEMM) routines in the cuBLAS library for the GPU acceleration.

where $\mathbf{E}_{n+1}$ contains the embeddings for all users and items at the $n+1$ epoch; $\sigma(\cdot)$ is the GELU (Hendrycks and Gimpel, 2016) activation; $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable weight matrices; and $\text{LayerNorm}(\cdot)$ denotes the layer normalisation (Ba et al., 2016), which is used to enhance the training stability.

In Equation (6.2), we use a self-addition loop i.e., adding the original $\mathbf{E}_n$ to $\mathbf{E}_{n+1}$, which is inspired by the tying parameter technique in MLP-Mixer (Tolstikhin et al., 2021). The self-addition loop can help the model to preserve the information from the previous epoch and avoid overfitting. To justify our choice, we also implement a variant without the self-addition loop, which is identical to the MLP architecture used by the Graph-MLP model. We term this variant as plain MLP and we will compare our model to this plain MLP in an ablation study.

6.3.4 Contrastive Training of MLP-CGRec

Contrastive learning introduced in Section 2.2.4 aims to encourage similar pairs to stay close to each other while dissimilar ones are far apart in the latent space (Chen et al., 2020d; Chopra et al., 2005). In a contrastive recommendation scenario (Wu et al., 2019b; Zhou et al., 2021), we target learning representations for users and items, where interacted users and items stay closer in the learnt space. Inspired by the performance of InfoNCE loss (see Equation (2.25)), our proposed MLP-CGRec model uses the objective function as follows:

$$\mathcal{L} = \sum -\log \frac{f_{\text{s}}(\mathbf{e}_u, \mathbf{e}_{i_u^+}) + \sum_2^n f_{\text{s}}(\mathbf{e}_u, \mathbf{e}_{n^+})}{f_{\text{s}}(\mathbf{e}_u, \mathbf{e}_{i_u^+}) + f_{\text{s}}(\mathbf{e}_u, \mathbf{e}_{i_u^-}) + \sum_2^n f_{\text{s}}(\mathbf{e}_u, \mathbf{e}_{i_{con}^-})} \tag{6.3}$$

where $f_{\text{s}}(\cdot) = e^{\cos(\cdot)}$; i_u^+, n^+, i_u^- and i_{con}^- denote a positive item, a n-hop neighbour (i.e., a pseudo-positive sample), a random negative item and a contrastive negative item of user u, respectively; n determines how many hops of neighbours to incorporate.

With Equation (6.3), we aim to encourage users to stay closer with their positive items and pseudo-positive samples while maximising the distance between users with their random and contrastive negative items. To predict each user's preferred items, we use the dot products between the embeddings of users and items to compute the corresponding probabilities of interactions.

6.4 Datasets and Experimental Setup

We use the same datasets (i.e., MovieLens, Yelp and Amazon) as used in Section 5.4 to evaluate our proposed MLP-CGRec model and all used baselines. In the following, we aim to answer the following research questions:

RQ6.1. Can MLP-CGRec achieve higher efficiency compared with the existing graph-based recommender systems without significantly degrading its recommendation effectiveness?

RQ6.2. What is the impact of these pseudo-positive samples?
RQ6.3. What is the impact of the MLP-based learning method?

To answer **RQ6.1**, we compare our MLP-CGRec model with the following baselines: BPR, NCF, NGCF, LightGCN, UltraGCN, SGL (Wu et al., 2021) and MixGCF (Huang et al., 2021b). The detailed descriptions of BPR, NCF, NGCF, LightGCN, UltraGCN and SGL can be found in Section 3.4.1, while MixGCF has been described in Section 5.4.1. In particular, BPR and NCF are standard baselines; NGCF, LightGCN, UltraGCN and SGL are effective graph-based baselines; and MixGCF is a recent baseline based on an advanced sampling method. In this chapter, to avoid repetition, we do not consider the dynamic negative sampling approach proposed in Chapter 5 as a baseline since we will systematically compare the combinations of DNS, MLP-CGRec and other proposed models in Section 7.1.

Since the training efficiency and effectiveness are both critical in our study, we compare our MLP-CGRec with all baselines in terms of Normalised Discounted Cumulative Gain@10 (NDCG), Hit Ratio@10 (HR), max memory consumption, average epoch time and total training time, where the max memory consumption is monitored by Tensorboard[2]. Following Section 5.4.2, we split each dataset into a training set, validation set and test set with a ratio of 8:1:1. In addition, we construct 10 different testing sets with different sampled negative items for each dataset using different random seeds. Hence, the reported performance of each run is based on the average of the 10 testing sets. For a fair comparison, we conduct all experiments on the same machine with a GeForce RTX 2080Ti GPU. For significance testing, we apply a two one-sided equivalence test (TOST) (Liu et al., 2019a; MacAvaney et al., 2020; Mackenzie et al., 2018; Mohammad et al., 2018). The purpose of the TOST test is to examine if MLP-CGRec is significantly equivalent to a baseline with the acceptable range of inequality being $\pm 5\%$. We define success as outperforming a baseline in terms of effectiveness and efficiency or outperforming a baseline in terms of efficiency while not significantly decreasing its effectiveness.

To answer **RQ6.2**, we compare the following variants of MLP-CGRec: (i) 2-hop neighbours with contrastive negative items and (ii) 3-hop neighbours with contrastive negative items. Hence, we can clearly examine the effect of each type of neighbours. Recall that contrastive negative items are sampled only when multi-hop neighbours are incorporated hence we can neglect the condition when only contrastive negative items are sampled. To answer **RQ6.3**, we use an ablation study to examine the effectiveness of the proposed MLP-CGRec model when: (i) the MLP module is not applied; (ii) a plain MLP is applied; (iii) the proposed MLP-mixer is applied.

The latent dimension and batch size are fixed to 64 and 1000, respectively, for all models. For each dataset, we use 20% of the interactions as a test set; of the remaining, we use 10% as a validation set, and the remainder for training. For the trainable matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ used in MLP-CGRec, we closely follow the implementation details in (Tolstikhin et al., 2021) and set $\mathbf{W}_1$ and $\mathbf{W}_2$ to 32, which is half of the input latent dimension. For the $f_{\mathrm{top-k}}$ function, we empirically set k to 100 according to our early stage experiments. To tune all hyper-parameters, we

[2] https://www.tensorflow.org/tensorboard

apply a grid search, where the learning rate is tuned in $\{10^{-2}, 10^{-3}, 10^{-4}\}$; and the L_2 normalisation in $\{10^{-1}, 10^{-2}, ..., 10^{-5}\}$. The range of node dropout ratio for NGCF, SGL, LightGCN and UltraGCN can be found in Section 3.4.2. We use 3-hop neighbours in MLP-CGRec following other graph-based recommender systems (He et al., 2020; Wang et al., 2019c).

6.5 Results

In this section, we report the experimental results and answer our three research questions.

6.5.1 Effectiveness and Efficiency

Table 6.1 reports the overall performances of our MLP-CGRec model and all other baselines. The results in Table 6.1 show that MLP-CGRec achieves a competitive performance on all these used datasets. On MovieLens, although LightGCN and SGL slightly outperform our MLP-CGRec model, both models do not surpass MLP-CGRec by a significant difference according to the TOST (two one-sided test). In particular, our model generally performs better on larger datasets than on smaller datasets. For example, MLP-CGRec achieves the second-best and the best performance on the larger Yelp and Amazon datasets in terms of NDCG@10, respectively. This observation is consistent with the findings in (Hu et al., 2021; Tolstikhin et al., 2021), where both the Graph-MLP and MLP-Mixer models only outperform state-of-the-art models on larger datasets for both node and image classification tasks. We notice that SGL and UltraGCN do not always outperform LightGCN as reported by Mao et al. (2021) and Wu et al. (2021). This is due to the difference in the experimental setup where we additionally use 10 different testing sets to reduce the evaluation bias. Therefore, our results further demonstrate the promising generalisability of a simple MLP-based learning architecture, whose scalability is better than complicated networks. Furthermore, our proposed MLP-CGRec model consistently achieves the best training efficiency among all neural recommenders in terms of GPU memory consumption and training time. We owe this high efficiency of MLP-CGRec to the simple design of the MLP module and the neighbourhood construction, which can be accomplished quickly using the sparse matrix multiplication. Therefore, in answer to **RQ6.1**, we conclude that our MLP-CGRec model can achieve competitive recommendation effectiveness with a higher efficiency compared with existing graph-based recommender systems.

6.5.2 Impacts of Samples and Components

Figure 6.1 plots the NDCG@10 performances of all different variants of MLP-CGRec on all used datasets. In order to answer **RQ6.2**, we compare the variants of MLP-CGRec without using encoders over different hops of neighbours (green bars). Here, 1-hop neighbours are

Table 6.1: Experimental results of MLP-CGRec and other baselines on the three used datasets w.r.t. HR@10, NDCG@10, max GPU memory consumption (gigabytes), average epoch time (seconds) and total training time (minutes). The best effectiveness is highlighted in bold and the second best result is highlighted with underline. * denotes a significant difference compared to the result of MLP-CGRec using the TOST (two one-sided test) with p<0.05.

	MovieLens				
	NDCG	HR	Memory	Epoch Time	Training Time
BPR	0.2031*	0.1274*	2.91	45.8	61.2
NCF	0.2114*	0.1398*	3.88	58.7	100.5
NGCF	0.2517*	0.1665*	4.13	65.7	161.0
LightGCN	**0.3303**	**0.2329**	3.96	59.8	118.5
UltraGCN	0.2646*	0.1862*	3.98	61.3	147.5
MixGCF	0.2737*	0.1989*	4.38	60.3	132.2
SGL	<u>0.3116</u>	<u>0.2260</u>	4.08	69.3	135.6
MLP-CGRec	0.3004	0.2176	3.01	48.2	78.5
Diff (%)	-6.02	-6.57	-24.0	-19.4	-33.7
	Yelp				
	NDCG	HR	Memory	Epoch Time	Training Time
BPR	0.1221*	0.1520*	5.42	84.5	96.7
NCF	0.1330*	0.1598*	6.07	98.9	123.3
NGCF	0.1420*	0.1736*	7.64	112.4	168.6
LightGCN	**0.2233**	<u>0.2597</u>	6.43	102.3	136.6
UltraGCN	0.2111	**0.2619**	6.70	108.9	166.8
MixGCF	0.2003*	0.2378*	6.91	107.8	158.4
SGL	0.1673*	0.2098*	6.61	101.9	145.8
MLP-CGRec	<u>0.2164</u>	0.2501	5.71	91.3	121.2
Diff (%)	-3.09	-4.51	-11.2	-10.1	-12.7
	Amazon				
	NDCG	HR	Memory	Epoch Time	Training Time
BPR	0.0745*	0.1138*	5.78	95.1	147.6
NCF	0.1093*	0.1678*	6.31	132.2	188.5
NGCF	0.1297*	0.1927*	7.91	168.9	241.4
LightGCN	<u>0.1519</u>	0.2177	7.01	149.2	206.0
UltraGCN	0.1302*	0.2021*	7.11	145.2	208.2
MixGCF	0.1403*	0.2107	8.03	143.9	211.4
SGL	0.1581	**0.2201**	6.58	151.5	184.8
MLP-CGRec	**0.1593**	<u>0.2190</u>	5.90	101.3	152.2
Diff (%)	+4.87	-0.50	-10.3	-33.1	-17.6

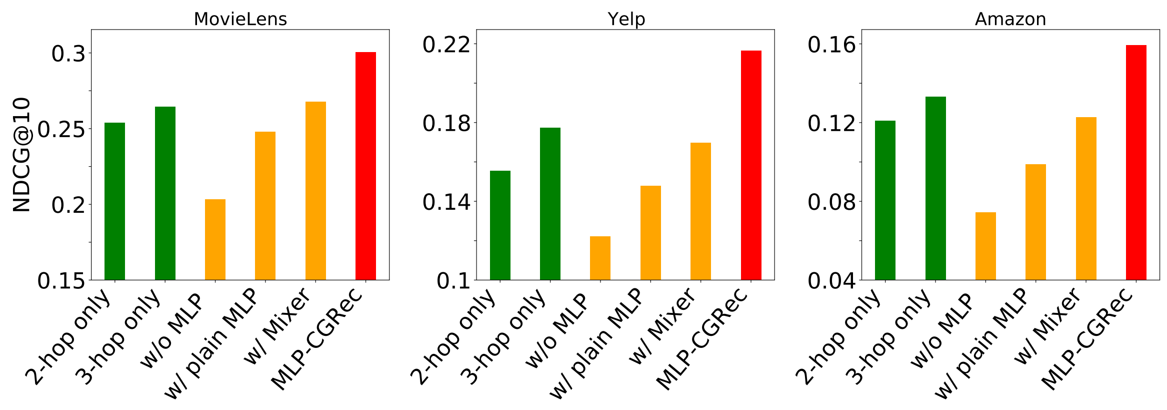


Figure 6.1: Comparison between different variants of MLP-CGRec, where the green bars are variants with multiple neighbours, orange bars are variants with different MLPs and the red bar is the final MLP-CGRec model.

neglected because they are already included in the interaction graph. By comparing the performances of variants with 2-hop and 3-hop neighbours, we find that the 3-hop neighbours bring more gains over the 2-hop neighbours, which means that users sharing one interacted item may not necessarily share the same overall interests. This explains why the 3-hop neighbourhood aggregation becomes the most common setup of the graph-based recommender systems. Therefore, in answer to **RQ6.2**, we can conclude that both types of pseudo-positive samples i.e., 2-hop and 3-hop neighbours, can improve the effectiveness of a graph-based recommender system.

In answer to **RQ6.3**, we compare the effectiveness of different variants of MLP-CGRec, which use different MLP-based approaches including the plain MLP and the MLP-Mixer. Figure 6.1 shows that the variant of MLP-CGRec using the MLP-Mixer constantly outperforms the one with a plain MLP and the plain MLP surpasses the one with no MLP on all datasets. This observation justifies our choice of incorporating the MLP architecture inspired by the MLP-Mixer as our representation learning module. Lastly, none of the variants can outperform MLP-CGRec, which means the integration of all proposed modules can achieve the best performance.

6.6 Conclusions

In this chapter, we proposed MLP-CGRec, an MLP-based recommender that uses a neighbourhood construction method and a contrastive objective to replace the classic neighbourhood aggregation and message passing to achieve a competitive recommendation effectiveness with up to 33.7% running time reduction (see Table 6.1). Our proposed MLP-CGRec model has been shown to achieve the best efficiency and a comparable effectiveness with state-of-the-art graph-based and contrastive baselines on three public datasets, namely MovieLens, Yelp and Amazon. Furthermore, our ablation study (see Figure 6.1) reveals the effects of the different proposed modules. In particular, we found that the proposed MLP-Mixer is more effective than the plain

MLP and that 3-hop neighbours are more effective than 2-hop neighbours (see Figure 6.1).

To summarise, we further validated the hypothesis of our proposed thesis statement in Section 1.2 that using graph contrastive learning to select both the contrastive negative items sampled by DNS and the pseudo-positive samples, can alleviate the low-efficiency issue of graph-based recommender systems. Differing from the dynamic negative sampling approach proposed in Chapter 5, MLP-CGRec relies on a multiple sampling approach that incorporates both pseudo-positive and negative samples. Specifically, although MLP-CGRec does not consistently outperform all baselines in terms of the recommendation effectiveness, it alleviates the low-efficiency issue of graph-based recommender systems which has not been studied in Chapter 5.

In the next chapter, we will investigate how to combine the various techniques and approaches proposed in this chapter as well as Chapters 3, 4 and 5. Specifically, we will first recall all our proposed approaches and list all possible combinations. Next, we will present how to combine different approaches followed by the corresponding experiments. Finally, we will report if some of the incorporated techniques , namely heterogeneous graph representation learning (Chapter 3), graph pre-training (Chapter 4) and graph contrastive learning (Chapters 5 and 6), can be combined such that the integrated model can provide better recommendations.

Chapter 7

Integrated Graph Recommender Systems

7.1 Introduction

In the previous chapters, we have presented all of the essential building blocks for our thesis statement proposed in Section 1.2. In particular, we first proposed the Social Graph Pre-training (SGP) model (Chapter 3) by leveraging heterogeneous graph representation learning to encode the user's social relations and enhance the final recommendations. Next, we presented two pre-training schemes, namely **Single-P** and **Multi-P** in Chapter 4, where both schemes have been shown to be effective after the fine-tuning of general recommender systems. Finally, we proposed dynamic negative sampling (DNS) and the MLP-CGRec model to provide more effective negative items in Chapter 5 and efficient pseudo-positive samples in Chapter 6, respectively.

Although our proposed approaches/models/schemes have been devised to enhance the effectiveness, robustness and efficiency of ranking-based recommender systems, different components have different advantages. For example, the proposed SGP model and the two pre-training schemes, i.e., **Single-P** and **Multi-P** can improve both the effectiveness and robustness of the recommendation as well as alleviate the *cold-start* problem. In addition, DNS can enhance the effectiveness of general recommenders by providing more informative negative items instead of relying on the side information. Moreover, MLP-CGRec as proposed in Chapter 6, can reduce the training computational cost and memory consumption of the graph-based recommenders. However, at this stage, it is unclear if our proposed approaches can be integrated into one model such that this integrated model can have more comprehensive abilities. Therefore, in this chapter, we investigate which proposed approaches can be combined and how such integrated models perform compared with their components. To have a better understanding on the focus of each proposed approach, we split the overall recommendation framework into three different phases, namely the *Pre-processing*, *Model Training* and *Evaluation* phases. Specifically, we regard all of the steps before the actual training of a recommender system as the *Pre-processing* phase. For the case of a two-phase training i.e., pre-training + fine-tuning, we treat the pre-training stage as the *Pre-processing*, while the fine-tuning is treated as the *Model Training*. Furthermore,

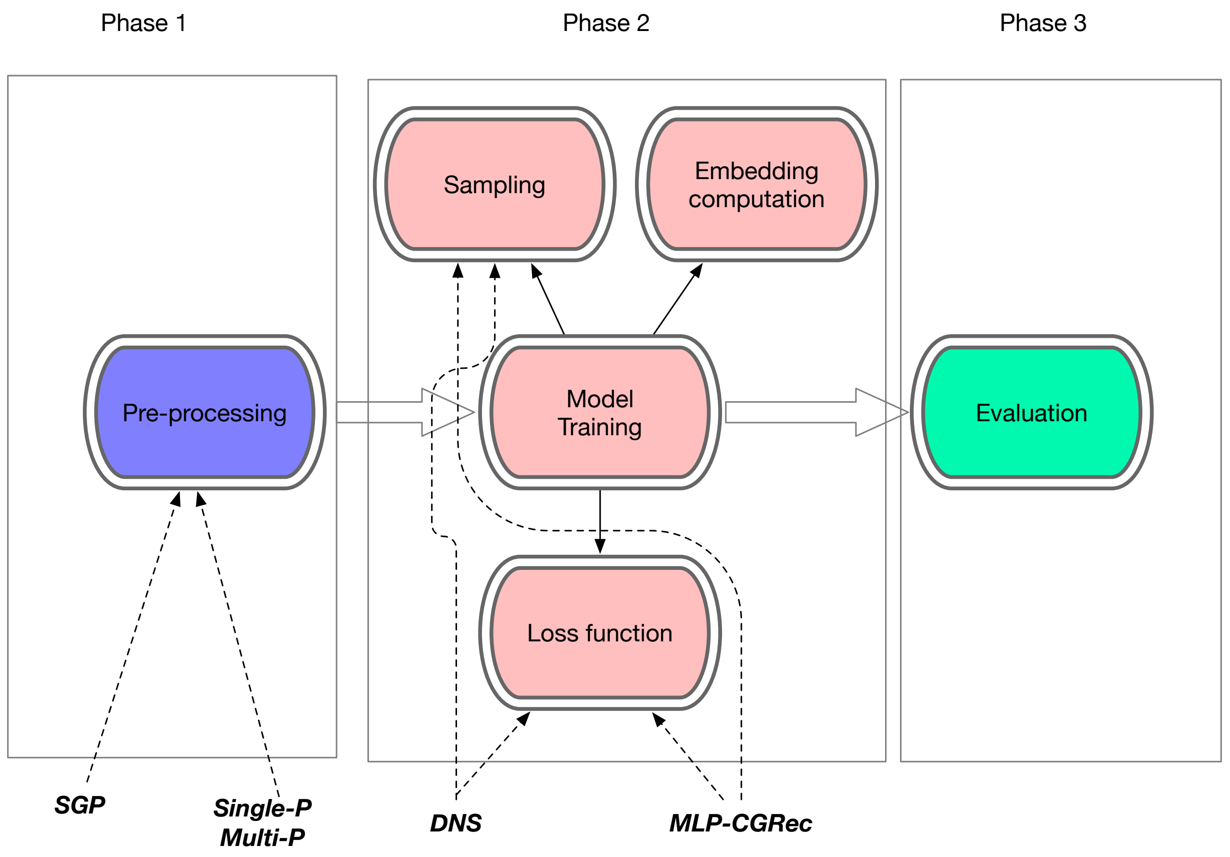


Figure 7.1: An overview of the recommendation framework.

the *Evaluation* phase includes different evaluation strategies such as the leave-one-out (see Sections 3.4.2 and 4.5.3) and random splitting (see Sections 5.4.2 and 6.4) evaluations. We use Figure 7.1 to visualise the focus of each of our proposed approaches. In principle, approaches that focus on the same phase can only be substituted by each other. In Section 7.2, we show the evaluated combinations in details.

In addition, the experimental setups used across different chapters are not unified, which also explains why some baselines performed differently on the same dataset. For example, we used a leave-one-out strategy to evaluate the SGP model (Chapter 3) and the two pre-training schemes (Chapter 4). On the other hand, we used the random split method to assess DNS (Chapter 5) and the MLP-CGRec model (Chapter 6). Although there is no particular advantage of one evaluation method over another, it is difficult to compare different models and techniques without a fair and unified evaluation. Therefore, we adapt a unified setup to evaluate all the combinations of our proposed methods. Last but not least, we used 10 different testing sets to evaluate all of our models in order to reduce the evaluation bias on some specific testing negatives. Although our evaluation method is fairer than those only using one testing set, Krichene and Rendle (2022) have recently argued that the sampled metrics are still inconsistent. Specifically, to speed up the evaluation step, existing recommender systems (He et al., 2020, 2017; Wang et al., 2019c) are usually evaluated based on a sampled testing set where only a certain amount of negative

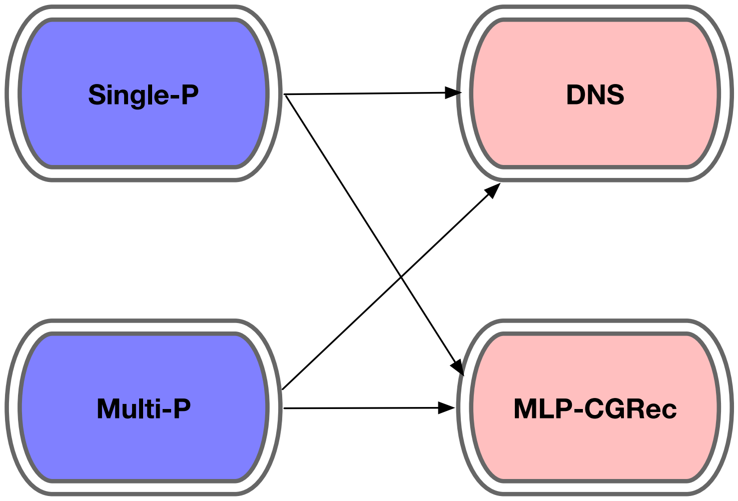


Figure 7.2: An overview of 4 possible combinations between different methods.

items are sampled and used. Although the evaluation efficiency is improved, it can induce an evaluation bias by accidentally sampling some easy negatives or omitting some hard negatives in the testing set Krichene and Rendle (2022). For the case of ranking-based recommender systems, easy negatives are defined as those items commonly placed at the tail of a ranking list and hard negatives are defined as those items commonly placed at the head of a ranking list by recommender systems. Therefore, to alleviate such an evaluation bias when the testing set is sampled in this chapter, we also use the full set of negative items.

This chapter is structured as follows: Section 7.2 presents all possible combinations of the proposed approaches; Section 7.3 describes our unified experimental setup; Section 7.4 and Section 7.5 present the experimental results and conclusions, respectively.

7.2 Method

In this section, we aim to study the different combinations of our proposed approaches among the SGP model, the **Single-P/Multi-P** schemes, the dynamic negative sampling approach and the MLP-CGRec model. As mentioned in Section 7.1, only models that address different phases can be combined. For example, one model can only use one pre-training scheme; hence the pre-training methods of SGP and **Single-P/Multi-P** cannot be integrated into one model. Similarly, one model can only have one sampling approach i.e., either DNS or the multiple sampling approach used by MLP-CGRec. Furthermore, as mentioned in Section 1.3, **Single-P** is regarded as a generalised version of SGP on the social relations. Therefore, to avoid unnecessary and redundant combinations, we can skip the SGP model when choosing components to be combined with other approaches. In Figure 7.2, we summarise 4 different possible combinations that will be examined in this chapter. In the following two sections, we will discuss how to combine the pre-training scheme with each sampling approach.

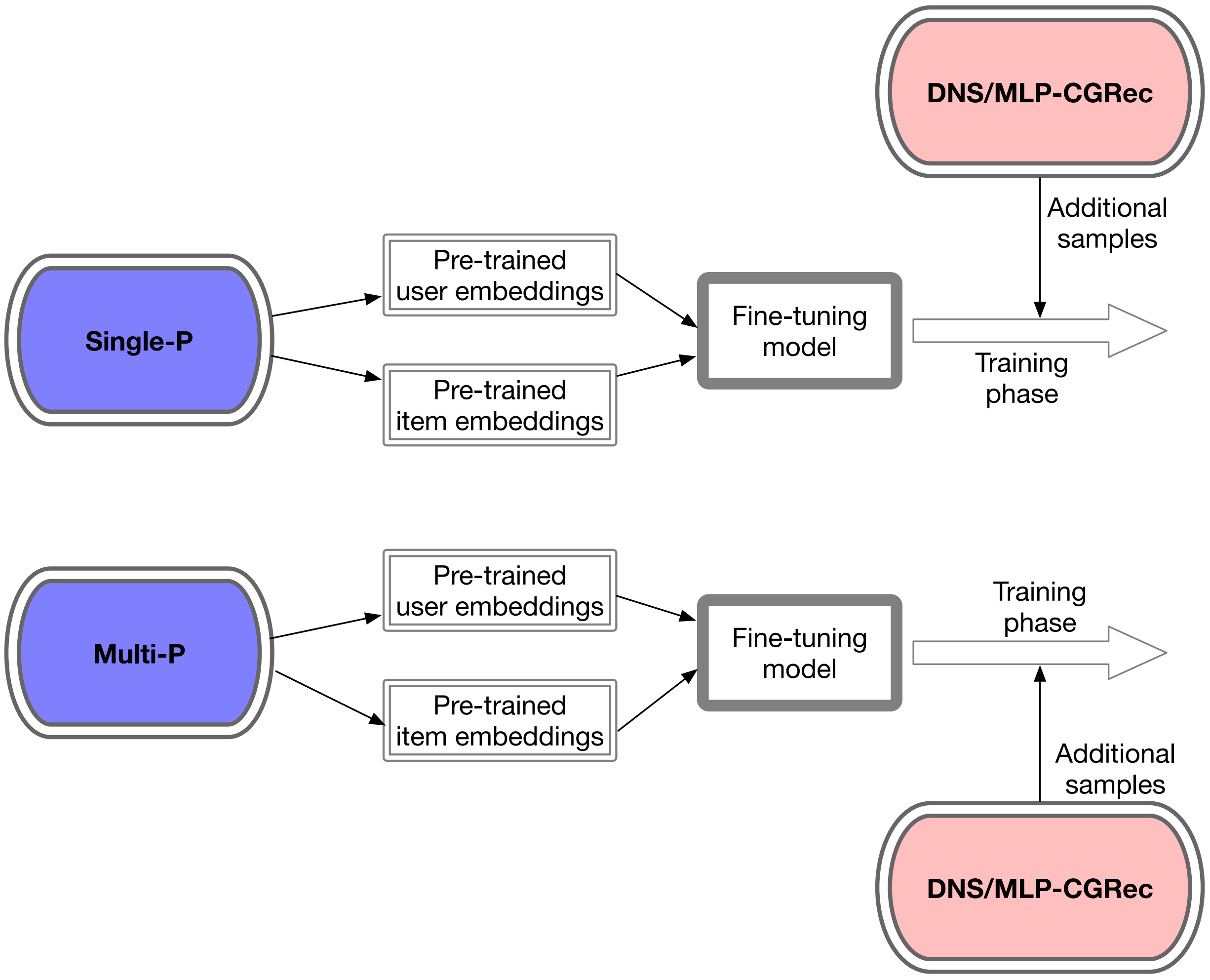


Figure 7.3: An illustration of 4 possible integrated models.

7.2.1 Pre-training with Dynamic Negative Sampling

As shown in Figure 7.2, the **Single-P** and **Multi-P** schemes can be combined with the dynamic negative sampling method. Specifically, we aim to incorporate the side information of both users and items using **Single-P** or **Multi-P**. Afterwards, we use the pre-trained embeddings obtained from **Single-P** or **Multi-P** to initialise the fine-tuning model, where this fine-tuning model is trained with more informative negative items sampled by DNS.

More specifically, we first obtain the pre-trained embeddings of users ($\hat{\boldsymbol{U}}$) and items ($\hat{\boldsymbol{V}}$) from Equation (4.3) (**Single-P**) or from Equation (4.5) (**Multi-P**). Next, we use LightGCN (Equation 3.4) as the fine-tuning model for its consistently good performance on different datasets and under different evaluation methods as shown in Chapters 3, 4, 5, 6. After being initialised by $\hat{\boldsymbol{U}}, \hat{\boldsymbol{V}}$, LightGCN is trained with more informative negative items obtained from DNS. Details on how to use DNS to sample informative negative items can be found in Section 5.3.2 (e.g., Equation (5.4) and Equation (5.5)). In particular, we illustrate the configuration of this integrated model that uses either **Single-P** or **Multi-P** for pre-training and DNS for sampling in Figure 7.3.

7.2.2 Pre-training with Multiple Sampling

Similarly, the **Single-P** and **Multi-P** schemes can be combined with the multiple sampling approach used by the MLP-CGRec model. After obtaining the pre-trained embeddings of users ($\hat{\boldsymbol{U}}$) and items ($\hat{\boldsymbol{V}}$), we use LightGCN for the fine-tuning again. Differing from the integrated model proposed in Section 7.2.1, pre-training with multiple sampling requests both negative and pseudo-positive samples. Specifically, these pseudo-positive samples are defined as graph neighbours in Section 6.3.1. Recall that these pseudo-positive samples are selected using the graph neighbourhood construction method introduced in Section 6.3.1. Furthermore, the negative samples are defined as contrastive negative items, where we use DNS to sample these negative items.

In the next section, we describe how we design experiments to evaluate these four different integrated models.

7.3 Experiments

To evaluate the effectiveness of our integrated models, we use the same three datasets as we have used in Chapter 4, namely Foursquare, MovieLens-1M and Epinions. This is because our pre-training schemes rely on multiple types of side information to generate the pre-trained embeddings. Table 4.1 showed the statistics of all used datasets. The pre-processing of each dataset has been described in Section 4.5.2. In addition, we follow the same graph construction process to build the single-relational and multi-relational graphs as mentioned in Section 4.5.2.

Since we aim to examine different abilities of our integrated models, we need to compare them with baselines from different perspectives. Hence, in addition to the baselines used in Chapter 4, we also add the MixGCF (Huang et al., 2021c) and PRIS (Lian et al., 2020a) models from Chapter 5 and UltraGCN (Mao et al., 2021) from Chapter 6 as baselines. Following the most recent work by Krichene and Rendle (2022), we split each dataset into a training set, validation set and testing set with a ratio of 8:1:1. However, differently from the multiple testing sets used in all previous chapters, this testing set contains the whole corpus of negative items. Therefore, we aim to rank the ground-truth item(s) against all negative items of each user instead of ranking the ground-truth item(s) against a portion of randomly sampled negative items. In particular, we use NDCG@10 and MAP@10 to measure the effectiveness of all proposed models and baselines. In addition, we evaluate the recommendation effectiveness of the four integrated models and their constituent approaches proposed in previous chapters across different groups of users. Next, we report the robustness of the examined models by applying different initialisations followed by a training efficiency comparison where we report the memory consumption, average/total training epoch time of the examined models.

In the following, we aim to answer the following research questions:

RQ7.1. Do the integrated models outperform models that they are built on as well as other

baselines in terms of the recommendation effectiveness?
RQ7.2. Do the integrated models retain the abilities of the models that they are built on in alleviating the *cold-start* problem as well as the low-robustness and low-efficiency issues?

7.4 Results

In this section, we report the experimental results and answer our two aforementioned research questions.

7.4.1 Effectiveness of the Integrated Models

Table 7.1 reports the NDCG@10 and MAP@10 measures of four integrated models compared with all baselines and the best performing proposed model from each chapter. For example, in Table 4.2, LightGCN$_{+\textbf{Single-P}}$ and LightGCN$_{+\textbf{Multi-P}}$ are observed to be the best performing models for each pre-training scheme. From Table 7.1, we observe that **Multi-P**+DNS achieves the best effectiveness on the Foursquare and MovieLens datasets and the second best on the Epinions dataset in terms of the MAP metric. This suggests that our proposed models/schemes can be integrated to further improve the recommendation effectiveness. At the same time, if we compare the four integrated models, we find that **Single-P**+DNS and **Single-P**+multiple sampling underperform **Multi-P**+DNS and **Multi-P**+multiple sampling, respectively. Indeed, since the integrated models using the multiple sampling approach (**Single-P**+multiple sampling and **Multi-P**+multiple sampling) always underperform the ones using DNS (**Single-P**+DNS and **Multi-P**+DNS), we can conclude that the multiple sampling approach is less effective than DNS in terms of the recommendation effectiveness. This is reasonable since the multiple sampling approach used by MLP-CGRec has been proposed to improve the training efficiency of graph-based recommendation as mentioned in Section 6.3.1. Specifically, we improve the efficiency by using only sampled positive neighbours leading to a sacrifice in effectiveness. By comparing the integrated models with LightGCN$_{\text{DNS}}$ and MLP-CGRec, we find that incorporating the pre-trained embeddings can further enhance these two models i.e., LightGCN$_{\text{DNS}}$ and MLP-CGRec, that use our proposed sampling approaches.

Therefore, in answer to **RQ7.1**, we can conclude that an integrated model can outperform its corresponding pre-trained model and sampling-based model. In particular, the combination of **Multi-P** (Chapter 4) with that the dynamic negative sampling approach (Chapter 5) can achieve the best effectiveness on the Foursquare and MovieLens datasets and significantly outperform the used baselines as well as the other integrated models. This conclusion consolidates the hypothesis of our proposed thesis by further improving the effectiveness of graph-based recommender systems using a combination of our proposed approaches.

Table 7.1: An effectiveness comparison of the integrated models with all baselines. The best and second best effectiveness are marked in boldface or underlined, respectively. We use $*$ to denote a significant difference between the best performing model and the rest, according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$. In addition, we use † to denote a significant difference between the best performing integrated model and other integrated models, according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$.

Model	Foursquare		MovieLens-1M		Epinions	
	NDCG	MAP	NDCG	MAP	NDCG	MAP
HIRE	0.4932^{*}	0.4275^{*}	0.0967^{*}	0.0563^{*}	0.0636^{*}	0.0488^{*}
cVAE	0.5026^{*}	0.4138^{*}	0.0648^{*}	0.0425^{*}	0.0501^{*}	0.0382^{*}
SSLIM	0.5586^{*}	0.4371^{*}	0.0627^{*}	0.0413^{*}	0.0501^{*}	0.0371^{*}
SGL	0.5762^{*}	0.5381^{*}	0.0708^{*}	0.0455^{*}	0.0682^{*}	0.0451^{*}
PT-GNN	0.5738^{*}	0.5377^{*}	0.0728^{*}	0.0471^{*}	0.0663^{*}	0.0473^{*}
SimGCL	0.5831^{*}	0.5541^{*}	0.0723^{*}	0.0511^{*}	0.0697^{*}	0.0452^{*}
MF	0.5006^{*}	0.4701^{*}	0.0619^{*}	0.0446^{*}	0.0527^{*}	0.0384^{*}
NGCF	0.5138^{*}	0.4844^{*}	0.0652^{*}	0.0491^{*}	0.0639^{*}	0.0435^{*}
LightGCN	0.5362^{*}	0.5040^{*}	0.0692^{*}	0.0501^{*}	0.0677^{*}	0.0464^{*}
NCF	0.5107^{*}	0.4899^{*}	0.0630^{*}	0.0481^{*}	0.0590^{*}	0.0401^{*}
MixGCF	0.5421^{*}	0.5291^{*}	0.0708^{*}	0.0503^{*}	0.0653^{*}	0.0422^{*}
PRIS	0.5621^{*}	0.5291^{*}	0.0688^{*}	0.0491^{*}	0.0623^{*}	0.0431^{*}
UltraGCN	0.5321^{*}	0.5091^{*}	0.0638^{*}	0.0481^{*}	0.0603^{*}	0.0453^{*}
LightGCN$_{+\textbf{Single-P}}$	0.6162^{*}	0.5840^{*}	0.0952^{*}	0.0631^{*}	0.0717^{*}	0.0594^{*}
LightGCN$_{+\textbf{Multi-P}}$	$\underline{0.6264}$	$\underline{0.6089}^{*}$	$\underline{0.1068}^{*}$	$\underline{0.0689}^{*}$	$\underline{0.0792}$	**0.0623**
LightGCN$_{\text{DNS}}$	0.6182^{*}	0.5940^{*}	0.0902^{*}	0.0619^{*}	0.0728^{*}	0.0603^{*}
MLP-CGRec	0.5228^{*}	0.4897^{*}	0.0618^{*}	0.0473^{*}	0.0593^{*}	0.0449^{*}
Single-P+DNS	$0.6259^{*\dagger}$	$0.5937^{*\dagger}$	$0.0901^{*\dagger}$	$0.0642^{*\dagger}$	$0.0744^{*\dagger}$	0.0603^{*}
Single-P+multiple sampling	$0.5981^{*\dagger}$	$0.5463^{*\dagger}$	$0.0714^{*\dagger}$	$0.0466^{*\dagger}$	$0.0633^{*\dagger}$	$0.0442^{*\dagger}$
Multi-P+DNS	**0.6478**	**0.6271**	**0.1128**	**0.0708**	**0.0800**	$\underline{0.0613}$
Multi-P+multiple sampling	$0.6002^{*\dagger}$	$0.5669^{*\dagger}$	$0.0901^{*\dagger}$	$0.0613^{*\dagger}$	$0.0645^{*\dagger}$	$0.0472^{*\dagger}$

7.4.2 Detailed Performances of the Integrated Models

In Table 7.2, we report the effectiveness of the four integrated models and their constituent approaches across different groups of users on the Foursquare and Epinions datasets. We do not report the results for the MovieLens-1M dataset because as we have mentioned in Section 4.6.5, the MovieLens-1M dataset has only users with more than 20 interactions, while typically users with more than 20 interactions can hardly be called as *cold-start* users. From Table 7.2, we find that **Multi-P**+DNS almost significantly outperforms all other models across different groups of users. At the same time, **Single-P**+DNS can also outperform LightGCN$_{+\textbf{Single-P}}$. Therefore, we can conclude that by incorporating the DNS approach, the recommendation effectiveness of a pre-trained recommender can be further improved for both the *regular* and *cold-start* users. However, we notice that the improvement of **Multi-P**+DNS over LightGCN$_{\textbf{Multi-P}}$ on the *cold-start* users is relatively marginal (Foursquare: 0.5829→0.5907). On the Epinions dataset, **Multi-P**+DNS does not outperform LightGCN$_{\textbf{Multi-P}}$ regarding the recommendation effectiveness for

Table 7.2: NDCG@10 performances of four integrated approaches and our proposed models across different groups of users on the two used datasets. We use $*$ to denote a significant difference between the best performing model i.e., **Multi-P**+DNS, and the rest, according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$.

Model	Foursquare			Epinions		
	Overall	Cold-start	Regular	Overall	Cold-start	Regular
LightGCN	0.5362*	0.5117*	0.5538*	0.0677*	0.0532*	0.0683*
LightGCN$_{+\textbf{Single-P}}$	0.6162*	0.5458*	0.6397*	0.0717*	0.0653*	0.0731*
LightGCN$_{+\textbf{Multi-P}}$	<u>0.6364</u>*	<u>0.5839</u>*	<u>0.6458</u>*	<u>0.0792</u>*	**0.0701**	<u>0.0802</u>*
LightGCN$_{\text{DNS}}$	0.6182*	0.5829*	0.6397*	0.0728*	0.0658*	0.0732*
MLP-CGRec	0.5228*	0.5409*	0.5458*	0.0593*	0.0561*	0.0602*
Single-P+DNS	0.6259*	0.5748*	0.6338*	0.0744*	0.0677*	0.0798*
Single-P+multiple sampling	0.5981*	0.5534*	0.6139*	0.0633*	0.0603*	0.0700*
Multi-P+DNS	**0.6478**	**0.5907**	**0.6597**	**0.0800**	<u>0.0699</u>	**0.0811**
Multi-P+multiple sampling	0.6002*	0.5539*	0.6208*	0.0645*	0.0625*	0.0728*

Table 7.3: Standard deviations (denoted std.) and means of the NDCG@10 performances over 50 random seeds. A lower standard deviation indicates a better stability. The best and second best values are marked in boldface or underlined, respectively

Model	Foursquare		Epinions		MovieLens-1M	
	std.	mean	std.	mean	std.	mean
LightGCN	0.0209	0.5365	0.0092	0.0657	0.0232	0.0692
LightGCN$_{+\textbf{Single-P}}$	<u>0.0050</u>	0.6262	<u>0.0039</u>	0.0711	<u>0.0089</u>	0.0952
LightGCN$_{+\textbf{Multi-P}}$	**0.0039**	0.6264	**0.0031**	0.0780	**0.0041**	0.1068
LightGCN$_{\text{DNS}}$	0.0189	0.6163	0.0089	0.0700	0.0200	0.0902
MLP-CGRec	0.0227	0.5289	0.0156	0.0586	0.0212	0.0618
Single-P+DNS	0.0089	0.6237	0.0069	0.0734	0.0123	0.0901
Single-P+multiple sampling	0.0153	0.5883	0.0089	0.0630	0.0195	0.0714
Multi-P+DNS	0.0077	0.6378	0.0054	0.0792	0.0101	0.1128
Multi-P+multiple sampling	0.0129	0.6082	0.0097	0.0637	0.0153	0.0901

the *cold-start* users. This observation reveals that these sampling approaches cannot help the *cold-start* recommendation. This is because our proposed sampling approaches relies on the learned embeddings; hence poor representations of the *cold-start* users and items lead to less informative samples. As a result, the corresponding integrated models cannot largely improve the recommendation effectiveness for *cold-start* users, compared with the original pre-trained models i.e., **Single-P** and **Multi-P**.

In Table 7.3, we analyse the standard deviations and average NDCG@10 measure of the four integrated models and other baselines over 50 random seeds. In this table, a lower standard deviation indicates a higher robustness against different random seeds i.e., different initialisations. We observe that our integrated models cannot provide more robust recommendations compared with LightGCN$_{+\textbf{Single-P}}$ and LightGCN$_{+\textbf{Multi-P}}$. We explain this observation from the standard deviations of LightGCN$_{\text{DNS}}$ and MLP-CGRec as follows. On the used datasets, two sampling-based models i.e., LightGCN$_{\text{DNS}}$ and MLP-CGRec, exhibit a worse robustness compared with those two pre-trained models. In fact, LightGCN$_{\text{DNS}}$ and MLP-CGRec are equally or even less robust than the base model LightGCN. Given different initialisations, sampling-based models might select different positive neighbours or negative items, which may lead to a higher variance of the final recommendation effectiveness. As a result, these sampling methods will degrade the robustness of our integrated models. Although integrated models cannot achieve the best robustness, we find that **Single-P**+DNS and **Multi-P**+DNS are more robust than LightGCN$_{\text{DNS}}$; **Single-P**+multiple sampling and **Multi-P**+multiple sampling are also more robust than MLP-CGRec. This observation suggests that we can improve the robustness of sampling-based models by incorporating the pre-trained embeddings. In addition, our proposed integrated models are all indeed more robust than the LightGCN model. Therefore, we can conclude that by combing the graph pre-training and graph-based sampling methods, the integrated models can achieve a higher robustness compared with the model based on a classic graph neural network i.e., LightGCN, but they are less robust than our proposed pre-training models.

Finally, we monitor and report the maximum memory consumption, average epoch time and total training time of the four integrated models and the used baselines in Table 7.4. From this table, we find that MLP-CGRec is the most efficient model among our proposed models across different metrics, while all the integrated models exhibit a larger memory consumption and longer training periods. This is also expected because all the integrated models include two training phases i.e., the pre-training and fine-tuning phases. Therefore, all the integrated models have doubled the total training time. In addition, the maximum memory consumption of the integrated models usually occurs during the pre-training phase leading to a higher consumption than MLP-CGRec. Therefore, we can conclude that the integrated models cannot provide a higher efficiency in terms of both the memory consumption and training time.

In answer to **RQ7.2**, our integrated models, especially **Multi-P**+DNS can achieve an effective *cold-start* recommendation while ensuring the recommendation effectiveness for *regular* users. Although different combinations cannot achieve the best robustness, we can use the pre-trained embeddings to enhance the robustness of the sampling-based models. It is of note that the integrated models are more robust than a standard baseline i.e LightGCN. Last but not least, we cannot further improve the efficiency of our proposed models by combining a pre-training scheme with a sampling-based approach.

In summary, we can use the integrated models to further boost the recommendation effectiveness of the graph-based recommender systems as well as alleviate the *cold-start* problem. However, the integrated models are less robust than our proposed models i.e., LightGCN$_{\textbf{Multi-P}}$ and LightGCN$_{\textbf{Single-P}}$. Furthermore, the training efficiency of the integrated models is also lower than our proposed MLP-CGRec model.

Table 7.4: Efficiencies comparison of four integrated models with baselines, where Memory, Epoch, Total denote the maximum memory consumption (gigabyte), average epoch time (second) and total training time (minute). The best and second best efficiencies are marked in boldface or underlined, respectively.

	Foursquare		
	Memory	Epoch	Total
LightGCN	<u>2.91</u>	<u>48.2</u>	<u>78.5</u>
LightGCN$_{+\textbf{Single-P}}$	3.88	58.7	100.5
LightGCN$_{+\textbf{Multi-P}}$	4.13	65.7	161.0
LightGCN$_{\text{DNS}}$	3.96	75.8	118.5
MLP-CGRec	**2.51**	**45.8**	**61.2**
Single-P+DNS	4.05	63.9	194.6
Single-P+multiple sampling	3.26	50.2	208.0
Multi-P+DNS	4.39	70.9	287.3
Multi-P+multiple sampling	3.91	60.2	210.3
	Epinions		
	Memory	Epoch	Total
LightGCN	<u>5.42</u>	<u>54.5</u>	<u>96.7</u>
LightGCN$_{+\textbf{Single-P}}$	6.07	58.9	123.3
LightGCN$_{+\textbf{Multi-P}}$	7.64	68.4	168.6
LightGCN$_{\text{DNS}}$	6.43	75.3	136.6
MLP-CGRec	**5.03**	**48.9**	**90.8**
Single-P+DNS	6.18	76.9	266.4
Single-P+multiple sampling	5.87	52.3	212.5
Multi-P+DNS	7.78	82.8	313.6
Multi-P+multiple sampling	7.02	60.5	289.3
	MovieLens-1M		
	Memory	Epoch	Total
LightGCN	<u>8.92</u>	<u>58.8</u>	<u>152.7</u>
LightGCN$_{+\textbf{Single-P}}$	10.28	60.12	189.3
LightGCN$_{+\textbf{Multi-P}}$	15.96	61.78	203.9
LightGCN$_{\text{DNS}}$	9.98	78.52	166.6
MLP-CGRec	**6.39**	**49.21**	**110.8**
Single-P+DNS	10.28	78.92	336.5
Single-P+multiple sampling	10.28	55.23	296.4
Multi-P+DNS	15.96	79.77	343.9
Multi-P+multiple sampling	15.96	58.97	309.3

7.5 Conclusions

In this chapter, we studied how to combine our proposed models and schemes from the previous chapters. Specifically, we proposed four integrated models consisting of the combinations between two pre-training schemes (Chapter 4) and two sampling-based methods (Chapter 5 and Chapter 6). After presenting each integrated model in Section 7.2, we evaluated the recommendation effectiveness of the integrated models compared with our own models from the previous chapters as well as 13 competitive baselines. In particular, we used a unified evaluation method to calculate the effectiveness of all examined models, where we used the full set of negative items in the testing set.

To answer **RQ7.1**, we compared the recommendation effectiveness of our integrated models and all other baselines. Extensive results showed that **Multi-P**+DNS constantly and significantly outperform all baselines and other integrated models on the three used datasets (Table 7.1). In particular, **Multi-P**+DNS also improves the NDCG measure of its constituent components i.e., LightGCN$_{+\textbf{Multi-P}}$ and LightGCN$_{\text{DNS}}$ on the Foursquare dataset by 3.4% and 4.9%, respectively, demonstrating that our proposed models can be combined to achieve a better recommendation. To answer **RQ7.2**, we first compared the recommendation effectiveness of the integrated models and baselines across different group of users. Table 7.2 showed that **Multi-P**+DNS can maintain the effective *cold-start* recommendation of the pre-trained models and ensure an enhanced recommendation effectiveness for the *regular* users. Afterwards, we compared the robustness (Table 7.3) and efficiency (Table 7.4) of all evaluated models. The detailed results showed that our integrated models cannot achieve a higher robustness nor a higher efficiency compared with our models proposed in the previous chapters. In this chapter, we further consolidated our proposed thesis statement (Section 1.2) by integrating different techniques of graph representation learning instead of leveraging only one single technique. Although we cannot further enhance the robustness and efficiency of graph-based recommenders by combining our proposed methods, **Multi-P**+DNS has been shown promising for the graph-based recommendation. In the next chapter, we will close this thesis by summarising the results and conclusions of each chapter and discussing possible future directions uncovered by this work.

Chapter 8

Conclusions and Future Work

8.1 Contributions and Conclusions

This thesis addressed the challenges of incorporating graph representation learning to make effective ranking-based recommendations and alleviate the *cold-start* problem as well as the issues of low robustness and low training efficiency of graph-based recommender systems.

Specifically, we postulated that by using the heterogeneous graph representation learning, graph pre-training and graph contrastive learning techniques, we can enhance the ranking-based recommendations provided by graph-based recommender systems. In Section 1.1, we argued that existing graph-based recommender systems have the following challenges:

- **C1:** Graph-based recommender systems cannot learn the heterogeneous relations between users and items, especially when side information e.g., social relations between users, are available.

- **C2:** When multiple types of side information are available, graph-based recommender systems struggle to incorporate them simultaneously.

- **C3:** Graph-based recommender systems still use the randomly sampled negative items for the pairwise learning, which are not informative.

- **C4:** Graph-based recommender systems consume extensive GPU computational power and training time, leading to an unsatisfactory training efficiency.

To address the four challenges above, we have proposed various models/schemes all through this thesis. Below, we will describe our main contributions and conclusions in addressing these challenges:

- **Conclusion 1: Effective Recommendations Using the Heterogeneous Graph Representation Learning:** To address **C1,** we explored how to leverage graph representation learning to capture the heterogeneous relations between users and items, including the user-item interactions and user-user social relations (see Chapter 3). In particular, we leveraged the heterogeneous graph technique to generate pre-trained embeddings using the social relations among users (Section 3.3.2). Next, we used the Gaussian Mixture Model (GMM) to decode the obtained initialisations from the pre-trained embeddings (Section 3.3.3.1). Finally, a graph-based model was used as the fine-tuning model to rank items for each user (Section 3.3.3.2). Extensive experimental results on three public datasets (Table 3.3) showed that our proposed Social-aware Gaussian Pre-trained Model significantly outperformed state-of-the-art baselines. Furthermore, the detailed user analyses in Tables 3.4 and 3.5 demonstrated that SGP is effective to address the *cold-start* problem.

- **Conclusion 2: Enhanced Recommendations by Graph Pre-training:** To address **C2**, we introduced two pre-training schemes, namely **Single-P** and **Multi-P** to enhance the effectiveness of representation-based recommender systems. In particular, we built single-relational and multi-relational graphs for **Single-P** and **Multi-P**, respectively, to capture the multiple types of relations between users and items (see Section 4.5.2). Compared with the aforementioned SGP model, **Single-P** and **Multi-P** can leverage multiple types of side information instead of relying on the social relations only. The extensive evaluation of our pre-training schemes showed that effectively pre-training the embeddings of both the users and items can significantly improve the recommendation effectiveness of existing recommender systems, namely NCF, MF, NGCF and LightGCN (see Figure 4.5). In addition, after pre-training by our proposed schemes, four existing models exhibited a higher robustness against different initialisations i.e., variable random seeds (see Table 4.3). Moreover, an in-depth analysis across different groups of users showed that by leveraging the side information through pre-training, our **Single-P** and **Multi-P** schemes can successfully alleviate the *cold-start* problem while ensuring effective recommendations for *regular* users.

- **Conclusion 3: Improved Ranking-based Recommendation by Graph Contrastive Negative Sampling:** We proposed a dynamic negative sampling (DNS) scheme to provide more informative negative samples for ranking-based recommender systems (see Chapter 5). Specifically, to address **C3**, DNS dynamically samples contrastive negative items after each training epoch solely based on the similarity scores between the users and items' latent embeddings without using any side information. In addition, we proposed a novel objective function (Equation (5.5)) to leverage those additionally sampled negative samples. Extensive experimental results showed that DNS can generally improve the performance of four ranking-based recommenders and can significantly outperform two strong baselines using different advanced negative sampling approaches (see Table 5.3). Furthermore, our entropy-based analysis indicates that DNS improves general ranking-based recommenders by providing contrastive negative items with s higher information gain compared with the randomly sampled negative items (see Figure 5.3.)

- **Conclusion 4: Efficient Graph-based Recommender Systems by Contrastive Multiple Sampling:** We proposed MLP-CGRec, an MLP-based recommender that uses a neighbourhood construction method and a contrastive objective function to replace the neighbourhood aggregation and the message passing of graph-based recommenders (see Chapter 6). In order to address **C4**, our neighbourhood construction method efficiently selects pseudo-positive samples i.e., the multi-hop positive neighbours of both the users and items to avoid the computationally expensive graph operations. In addition, we used our novel objective function to incorporate these pseudo-positive samples and contrastive negative items. The difference between the sampling methods of DNS (Chapter 5) and MLP-CGRec is that DNS only samples negative items while MLP-CGRec samples both positive and negative samples thereby achieving higher efficiency. In addition, an efficiency gain is obtained by avoiding the incorporation of all multi-hop neighbours and the multi-layer message passing. The experimental results in Table 6.1 showed that MLP-CGRec can effectively mitigate the low-efficiency issue of graph-based recommenders by reducing the GPU memory consumption and training time.

- **Conclusion 5: Combined Effectiveness by Integrating Multiple Graph-based Components:** We investigated whether we can combine the different models/schemes proposed in Chapters 3-6 to achieve integrated models that retain the benefits of each incorporated component. We first split the overall recommendation framework into three phases and listed four possible combinations in Section 7.2 (see Figure 7.2). Specifically, we can combine **Single-P** with DNS, **Single-P** with MLP-CGRec, **Multi-P** with DNS and **Multi-P** with MLP-CGRec. Next, we evaluated these four integrated models in comparison to 13 baselines in Section 7.3. Table 7.1 showed that **Multi-P**+DNS can significantly outperform competitive baselines as well as our own proposed models across three public datasets. In addition, we observed that **Multi-P**+DNS benefited *cold-start* users and ensured improved recommendation effectiveness for the *regular* users as reported in Table 7.2. However, none of the integrated models provided an enhanced robustness compared with our pre-trained models including LightGCN$_{+\textbf{Multi-P}}$. Furthermore, the integrated models are less efficient than MLP-CGRec because all the integrated models include a pre-training scheme, which costs more GPU memory consumption and training time.

Next, based on the results from Chapters 3 to 7, we now validate our thesis statement proposed in Section 1.2. Our thesis stated that we can enhance the effectiveness of the ranking-based recommender system by leveraging graph representation learning. Specifically, graph representation learning includes heterogeneous graph learning, graph pre-training and graph contrastive learning. Furthermore, we argued that we could use these techniques to alleviate the *cold-start* problem as well as the low-robustness and low-efficiency issues. In the following, we discuss the corresponding experimental results and observations that validate our proposed thesis statement.

- **Claim 1:** *By leveraging heterogeneous graph representation learning, a graph-based recommender system can achieve an enhanced performance and benefit cold-start users.* Our experiments in Chapter 3 validated this claim by showing that our proposed Social-aware Gaussian Pre-trained (SGP) model can significantly outperform 13 baselines (see Table 3.3). In particular, SGP leverages heterogeneous graph representation learning to encode the heterogeneous relations including the user-item interaction and user-user social relations to provide effective initialisations for both the users and items. Table 3.4 and Table 3.5 showed that SGP can also effectively alleviate the *cold-start* problem.

- **Claim 2:** *The graph pre-training technique can be used to provide an enhanced recommendation effectiveness with a higher robustness.* We argue that we have validated this claim in Chapter 4, where we proposed two pre-training schemes, namely **Single-P** and **Multi-P**. In particular, our proposed schemes can incorporate multiple types of side information to generate the users and items' pre-trained embeddings (see Sections 4.4.2 and 4.4.3). Afterwards, these pre-trained embeddings can be fine-tuned by existing ranking-based recommender systems. Experimental results in Figure 4.5 demonstrated that both pre-training schemes can improve the effectiveness of four existing recommenders including MF, NGCF, LightGCN and NCF. In addition, Table 4.2 and Table 4.3 showed that our pre-trained models significantly outperformed 10 state-of-the-art baselines and exhibited a higher robustness against different initialisations.

- **Claim 3:** *By leveraging the graph contrastive learning technique, graph-based recommenders can obtain an improved recommendation effectiveness and a higher training efficiency.* This claim has been validated in Chapter 5 and Chapter 6, where we proposed two contrastive sampling methods. First, dynamic negative sampling (DNS) is proposed in Chapter 5 to provide more informative negative items for ranking-based recommender systems. With these negative items sampled by DNS, the effectiveness of four existing models, namely BPR, LightGCN, SGL and SimGCL, are improved, as shown in Table 5.3. Second, in Chapter 6 we proposed an MLP-based graph recommender i.e., MLP-CGRec that incorporated the multiple sampling method consisting of DNS and a pseudo-positive sampler. Compared with DNS, MLP-CGRec has been shown to be more efficient since it can efficiently select pseudo-positive samples to avoid the redundant multi-layer message passing of graph neural networks (see Section 6.3.1). Indeed, the experimental results in Table 6.1 showed that MLP-CGRec remained the most efficient model among the tested graph-based recommenders and achieved competitive recommendation effectiveness.

In summary, we have validated each of the claims of our thesis statement in Section 1.2. We have showed that we can improve the effectiveness of ranking-based recommender systems using heterogeneous graph learning, graph pre-training and graph contrastive learning techniques. In particular, these techniques mentioned above can enhance the effectiveness of graph-based recommender systems and alleviate the *cold-start* problem as well as the low-robustness and low-efficiency issues. Furthermore, by combining the graph pre-training (**Multi-P**) and graph contrastive learning (DNS) techniques, we can further boost the effectiveness of the graph-based recommender systems. Next, we describe some future research directions for graph-based recommender systems in Section 8.2.

8.2 Directions for Future Work

In this section, we discuss possible future directions that could further benefit the applications of graph representation learning in the recommendation scenario.

Alleviate the sparsity issue of side information: In Chapter 3 and Chapter 4, we have shown how to incorporate heterogeneous relations between users and items using heterogeneous graph representation learning. Results have shown that the *cold-start* users especially benefited from those additional social relations because they can be used to infer the interests of users or the characteristics of items. However, due to privacy issues (Jeckmans et al., 2013), many users are not willing to share their personal details. For example, some users may set their social relations to be unavailable to the third party recommendation services. Therefore, inferring the attributes of users or the entities of items is an interesting topic to investigate and can enhance recommendation based on side information. Rossi et al. (2021) have proposed a feature propagation method for graph data with missing node features. Hence, it could be promising to apply this feature propagation method to alleviate the sparsity issue of side information in the recommendation scenario.

Incorporate other types of graph neural networks to model the user-item interactions: Across Chapters 3 to 7, we have only leveraged classic node-node type of graph neural networks, which can be regarded as variants of Graph Neural Networks (GNNs) (Kipf and Welling, 2017). However, these graph neural networks may suffer from the over-smoothing (Chen et al., 2020a) and over-squashing issues (Topping et al., 2021). Specifically, the over-smoothing issue refers to the situation when the embeddings of the connected nodes become indistinguishable, while the over-squashing issue refers to when information cannot be effectively propagated beyond 3-4 layers. This over-smoothing issue will affect those users and items whose amount of interactions is more than the average. Moreover, this over-squashing issue will stop graph-based recommenders for capturing the long-range dependency. There are many recent graph-based models that can avoid those issues. For example, it is interesting to investigate different types of graph neural networks for recommendation, where candidates include a hyper-graph network (Bai et al., 2021), a hyperbolic graph neural network (Sun et al., 2021) and a Lorentzian graph neural network (Zhang et al., 2021).

Use graph representation learning methods to enhance sequential recommendation: In the scope of this thesis, we only studied ranking-based recommender systems and did not apply the graph-based method for sequential recommendation. However, sequential recommendation also plays an essential role in many online platforms, particularly video and music recommendation platforms because users of such platforms commonly interact with videos/music in sequences (Quadrana et al., 2018). Although graph-based methods are known to be effective for predicting links between nodes, they usually cannot capture the long-range dependency and are computationally expensive for sequential inputs. Therefore, graph-based methods are less frequently applied for the sequential recommendation task. Recently, the combination of graph neural networks and transformers is becoming popular (Hu et al., 2020b; Rampášek et al., 2022; Yun et al., 2019). These graph transformer models can efficiently leverage sequential inputs and capture the long-range dependency. Therefore, we will consider investigating sequential recommendation using graph transformer networks in future work.

8.3 Concluding Remarks

This thesis has addressed a challenging task: the ranking-based recommendation task. In particular, this thesis contributed to developing the ranking-based recommendation task by incorporating graph representation learning. Specifically, we have shown that heterogeneous graph representation learning, graph pre-training and graph contrastive learning can effectively enhance the effectiveness of ranking-based recommender systems and alleviate the *cold-start* problem as well as the low-robustness and low-efficiency issues. However, there are still exciting topics and complex challenges in this research field, some of which have been highlighted in Section 8.2. This thesis have provided a concrete motivation and the cornerstone for further exploring these research directions in the future. We believe that using graph representation learning to represent users and items will continue to benefit the future development of the recommendation field.

Bibliography

Abdollahpouri, H., Burke, R., and Mobasher, B. (2019). Managing popularity bias in recommender systems with personalized re-ranking. In *Proceedings of AAAI*.

Aggarwal, C. C. et al. (2016). *Recommender systems*, volume 1. Springer.

Atz, K., Grisoni, F., and Schneider, G. (2021). Geometric deep learning on molecular representations. *Nature Machine Intelligence*, 3(12):1023–1032.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. In *Proc. of Deep Learning Symposium*.

Badsha, S., Yi, X., and Khalil, I. (2016). A practical privacy-preserving recommender system. *Data Science and Engineering*, 1(3):161–177.

Badsha, S., Yi, X., Khalil, I., and Bertino, E. (2017). Privacy preserving user-based recommender system. In *2017 IEEE 37th international conference on Distributed Computing Systems (ICDCS)*, pages 1074–1083. IEEE.

Bai, S., Zhang, F., and Torr, P. H. (2021). Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637.

Basu, C., Hirsh, H., Cohen, W., et al. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of AAAI*.

Beel, J., Genzmehr, M., Langer, S., Nürnberger, A., and Gipp, B. (2013). A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. In *Proceedings of the international workshop on reproducibility and replication in recommender systems evaluation*, pages 7–14.

Beutel, A., Chen, J., Doshi, T., Qian, H., Wei, L., Wu, Y., Heldt, L., Zhao, Z., Hong, L., Chi, E. H., et al. (2019). Fairness in recommendation ranking through pairwise comparisons. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2212–2220.

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, 46:109–132.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Conference on Neural Information Processing Systems*, pages 1–9.

Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., Dickie, D. A., Hernández, M. V., Wardlaw, J., and Rueckert, D. (2018). Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Proceedings of NeurIPS*.

Burke, R., O'Mahony, M. P., and Hurley, N. J. (2015). Robust collaborative recommendation. In *Recommender systems handbook*, pages 961–995. Springer.

Carter, T. (2007). An introduction to information theory and entropy. *Complex systems summer school, Santa Fe*.

Castells, P., Hurley, N., and Vargas, S. (2022). Novelty and diversity in recommender systems. In *Recommender systems handbook*, pages 603–646. Springer.

Chakraborty, S., Bisong, E., Bhatt, S., Wagner, T., Elliott, R., and Mosconi, F. (2020). Biomedbert: A pre-trained biomedical language model for qa and ir. In *Proceedings of COLING*.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. (2020a). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3438–3445.

Chen, J., Dong, H., Wang, X., Feng, F., Wang, M., and He, X. (2020b). Bias and debias in recommender system: A survey and future directions. *arXiv preprint arXiv:2010.03240*.

Chen, J., Ma, T., and Xiao, C. (2018). Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*.

Chen, M. and Liu, P. (2017). Performance evaluation of recommender systems. *International Journal of Performability Engineering*, 13(8):1246.

Chen, S., Niu, G., Gong, C., Li, J., Yang, J., and Sugiyama, M. (2021). Large-margin contrastive learning with distance polarization regularizer. In *Proc. of ICML*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020c). A simple framework for contrastive learning of visual representations. In *Proceedings of ICML*.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. (2020d). Big self-supervised models are strong semi-supervised learners. In *Proceedings of NeurIPS*.

Chen, Y. and de Rijke, M. (2018). A collective variational autoencoder for top-n recommendation with side information. In *Deep Learning for Recommender Systems*, pages 3–9.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. (2016). Wide & deep learning for recommender systems. In *Workshop on Deep Learning for Recommender Systems*, pages 7–10.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. (2019). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266.

Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Proc. of CVPR*.

Chuang, C.-Y., Robinson, J., Yen-Chen, L., Torralba, A., and Jegelka, S. (2020). Debiased contrastive learning. In *Proceedings of NeurIPS*.

Costa, E., Lorena, A., Carvalho, A., and Freitas, A. (2007). A review of performance evaluation measures for hierarchical classifiers. In *Evaluation methods for machine learning II: Papers from the AAAI-2007 workshop*, pages 1–6.

Cotter, P. and Smyth, B. (2000). Ptv: Intelligent personalised tv guides. In *Proceedings of AAAI*.

Dacrema, M. F., Cremonesi, P., and Jannach, D. (2019). Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM conference on recommender systems*, pages 101–109.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

Del Olmo, F. H. and Gaudioso, E. (2008). Evaluation of recommender systems: A new approach. *Expert Systems with Applications*, 35(3):790–804.

Deng, Z.-H., Huang, L., Wang, C.-D., Lai, J.-H., and Philip, S. Y. (2019). Deepcf: A unified framework of representation learning and matching function learning in recommender system. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 61–68.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ding, J., Quan, Y., He, X., Li, Y., and Jin, D. (2019). Reinforced negative sampling for recommendation with exposure data. In *Proceedings of IJCAI*.

Ebesu, T., Shen, B., and Fang, Y. (2018). Collaborative memory network for recommendation systems. In *SIGIR Conference on Research and Development in Information Retrieval*, pages 515–524.

Edwards, A., Camacho-Collados, J., De Ribaupierre, H., and Preece, A. (2020). Go simple and pre-train on domain-specific corpora: On the role of training data for text classification. In *Proceedings of COLING*.

Eksombatchai, C., Jindal, P., Liu, J. Z., Liu, Y., Sharma, R., Sugnet, C., Ulrich, M., and Leskovec, J. (2018). Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 world wide web conference*, pages 1775–1784.

Erhan, D., Courville, A., Bengio, Y., and Vincent, P. (2010). Why does unsupervised pre-training help deep learning? In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 201–208.

Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., and Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Proceedings of AISTATS*.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. (2019). Graph neural networks for social recommendation. In *The ACM Web Conference 2019*, pages 417–426.

Farseev, A., Samborskii, I., Filchenkov, A., and Chua, T.-S. (2017). Cross-domain recommendation via clustering on multi-layer graphs. In *Proceedings of SIGIR*.

Fouss, F., Pirotte, A., Renders, J.-M., and Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering*, 19(3):355–369.

Fu, D. and He, J. (2021). Sdg: A simplified and dynamic graph neural network. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2273–2277.

Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., and Schmidt-Thieme, L. (2010). Learning attribute-to-feature mappings for cold-start recommendations. In *Proceedings of ICDM*.

Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., et al. (2021). Graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Information Systems*, 1.

Ge, M., Delgado-Battenfeld, C., and Jannach, D. (2010). Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 257–260.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al. (2018). Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377.

Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017). Deepfm: A factorization-machine based neural network for ctr prediction. In *International Joint Conference on Artificial Intelligence*, pages 1725–1731.

Hamilton, W., Ying, Z., and Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Conference on Neural Information Processing Systems*, pages 1024–1034.

Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

Hansen, C., Hansen, C., Maystre, L., Mehrotra, R., Brost, B., Tomasi, F., and Lalmas, M. (2020). Contextual and sequential user embeddings for large-scale music recommendation. In *Fourteenth ACM conference on recommender systems*, pages 53–62.

Hao, B., Zhang, J., Yin, H., Li, C., and Chen, H. (2021a). Pre-training graph neural networks for cold-start users and items representation. In *Proceedings of WSDM*.

Hao, B., Zhang, J., Yin, H., Li, C., and Chen, H. (2021b). Pre-training graph neural networks for cold-start users and items representation. In *International Conference on Web Search and Data Mining*, pages 265–273.

Hariadi, A. I. and Nurjanah, D. (2017). Hybrid attribute and personality based recommender system for book recommendation. In *Proceedings of ICoDSE*.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. (2020). Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR Conference on Research and Development in Information Retrieval*, pages 639–648.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Web Conference*, pages 173–182.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hershey, J. R. and Olsen, P. A. (2007). Approximating the Kullback Leibler divergence between Gaussian mixture models. In *Proceedings of ICASSP*.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. (2019). Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Proc. of IEEE International Conference on Data Mining*, pages 263–272.

Hu, Y., You, H., Wang, Z., Wang, Z., Zhou, E., and Gao, Y. (2021). Graph-mlp: node classification without message passing in graph. *arXiv preprint arXiv:2106.04051*.

Hu, Z., Dong, Y., Wang, K., Chang, K.-W., and Sun, Y. (2020a). Gpt-gnn: Generative pre-training of graph neural networks. In *SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Hu, Z., Dong, Y., Wang, K., and Sun, Y. (2020b). Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710.

Huang, C., Xu, H., Xu, Y., Dai, P., Xiao, L., Lu, M., Bo, L., Xing, H., Lai, X., and Ye, Y. (2021a). Knowledge-aware coupled graph neural network for social recommendation. In *AAAI Conference on Artificial Intelligence*, pages 4115–4122.

Huang, T., Dong, Y., Ding, M., Yang, Z., Feng, W., Wang, X., and Tang, J. (2021b). Mixgcf: An improved training method for graph neural network-based recommender systems. In *Proceedings of SIGKDD*.

Huang, T., Dong, Y., Ding, M., Yang, Z., Feng, W., Wang, X., and Tang, J. (2021c). Mixgcf: An improved training method for graph neural network-based recommender systems. In *SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 665–674.

Hui, B., Yan, D., and Ku, W.-S. (2021). Node-polysemy aware recommendation by matrix completion with side information. In *IEEE International Conference on Big Data*, pages 636–642.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender systems: an introduction*. Cambridge University Press.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Jeckmans, A. J., Beye, M., Erkin, Z., Hartel, P., Lagendijk, R. L., and Tang, Q. (2013). Privacy in recommender systems. In *Social media retrieval*, pages 263–281. Springer.

Jha, K., Saha, S., and Singh, H. (2022). Prediction of protein–protein interaction using graph neural networks. *Scientific Reports*, 12(1):1–12.

Jiang, G., Wang, H., Chen, J., Wang, H., Lian, D., and Chen, E. (2021). xlightfm: Extremely memory-efficient factorization machine. In *SIGIR Conference on Research and Development in Information Retrieval*, pages 337–346.

Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-T. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*.

Ketkar, N. (2017). Stochastic gradient descent. In *Deep learning with Python*, pages 113–132. Springer.

Khenissi, S., Mariem, B., and Nasraoui, O. (2020). Theoretical modeling of the iterative properties of user discovery in a collaborative filtering recommender system. In *Proceedings of RecSys*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Koren, Y., Rendle, S., and Bell, R. (2022). Advances in collaborative filtering. *Recommender systems handbook*, pages 91–142.

Kraskov, A., Stögbauer, H., and Grassberger, P. (2004). Estimating mutual information. *Physical review E*, 69(6).

Kratsios, A. (2021). The universal approximation property. *Annals of Mathematics and Artificial Intelligence*, 89(5):435–469.

Krichene, W. and Rendle, S. (2022). On sampled metrics for item recommendation. *Communications of the ACM*, 65(7):75–83.

Kunaver, M. and Požrl, T. (2017). Diversity in recommender systems–a survey. *Knowledge-based systems*, 123:154–162.

Lam, X. N., Vu, T., Le, T. D., and Duong, A. D. (2008). Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 208–211.

Leino, J. and Räihä, K.-J. (2007). Case amazon: ratings and reviews as part of recommendations. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 137–140.

Li, H., Wang, Y., Lyu, Z., and Shi, J. (2020a). Multi-task learning for recommendation over heterogeneous information network. *IEEE Transactions on Knowledge and Data Engineering*, 34:789 – 802.

Li, J., Wang, Y., and McAuley, J. (2020b). Time interval aware self-attention for sequential recommendation. In *Proceedings of WSDM*.

Li, P. and Tuzhilin, A. (2020). Ddtcdr: Deep dual transfer cross domain recommendation. In *Proceedings of WSDM*.

Li, S., Kawale, J., and Fu, Y. (2015). Deep collaborative filtering via marginalized denoising auto-encoder. In *Conference on Information & Knowledge Management*, pages 811–820.

Li, X., Liu, Z., Guo, S., Liu, Z., Peng, H., Philip, S. Y., and Achan, K. (2021). Pre-training recommender systems via reinforced attentive multi-relational graph neural network. In *Proceedings of IEEE BigData*.

Li, X. and She, J. (2017). Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 305–314.

Lian, D., Liu, Q., and Chen, E. (2020a). Personalized ranking with importance sampling. In *Proceedings of WWW*.

Lian, D., Wang, H., Liu, Z., Lian, J., Chen, E., and Xie, X. (2020b). Lightrec: A memory and search-efficient recommender system. In *Web Conference*, pages 695–705.

Liang, D., Krishnan, R. G., Hoffman, M. D., and Jebara, T. (2018). Variational autoencoders for collaborative filtering. In *Proceedings of WWW*.

Lin, X., Zhen, H.-L., Li, Z., Zhang, Q.-F., and Kwong, S. (2019). Pareto multi-task learning. *Advances in neural information processing systems*, 32.

Liu, B., Craswell, N., Lu, X., Kurland, O., and Culpepper, J. S. (2019a). A comparative analysis of human and automatic query variants. In *Proc. of SIGIR*.

Liu, H., Dai, Z., So, D. R., and Le, Q. V. (2021a). Pay attention to mlps. In *Proc. of NeurIPS*.

Liu, S., Ounis, I., and Macdonald, C. (2022). An mlp-based algorithm for efficient contrastive graph recommendations. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2431–2436.

Liu, S., Ounis, I., Macdonald, C., and Meng, Z. (2020). A heterogeneous graph neural model for cold-start recommendation. In *SIGIR Conference on Research and Development in Information Retrieval*, pages 2029–2032.

Liu, T., Wang, Z., Tang, J., Yang, S., Huang, G. Y., and Liu, Z. (2019b). Recommender systems with heterogeneous side information. In *Web Conference*, pages 3027–3033.

Liu, Z., Ma, Y., Ouyang, Y., and Xiong, Z. (2021b). Contrastive learning for recommender system. *arXiv preprint arXiv:2101.01317*.

Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

Lu, J., Wu, D., Mao, M., Wang, W., and Zhang, G. (2015). Recommender system application developments: a survey. *Decision Support Systems*, 74:12–32.

Lu, S., Chen, H., Zhou, X., Wang, B., Wang, H., and Hong, Q. (2018). Graph-based collaborative filtering with mlp. *Mathematical Problems in Engineering*, 2018.

Ma, H., Zhou, D., Liu, C., Lyu, M. R., and King, I. (2011). Recommender systems with social regularization. In *Proceedings of WSDM*.

Ma, X., Guo, J., Zhang, R., Fan, Y., Ji, X., and Cheng, X. (2021). Prop: Pre-training with representative words prediction for ad-hoc retrieval. In *Proceedings of WSDM*.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of ICML*.

MacAvaney, S., Nardini, F. M., Perego, R., Tonellotto, N., Goharian, N., and Frieder, O. (2020). Efficient document re-ranking for transformers by precomputing term representations. In *Proc. of SIGIR*.

Mackenzie, J., Culpepper, J. S., Blanco, R., Crane, M., Clarke, C. L., and Lin, J. (2018). Query driven algorithm selection in early stage retrieval. In *Proc. of WSDM*.

Man, T., Shen, H., Jin, X., and Cheng, X. (2017). Cross-domain recommendation: An embedding and mapping approach. In *Proceedings of IJCAI*.

Manotumruksa, J., Macdonald, C., and Ounis, I. (2017). A personalised ranking framework with multiple sampling criteria for venue recommendation. In *Proceedings of CIKM*.

Manotumruksa, J., Macdonald, C., and Ounis, I. (2018). A contextual attention recurrent architecture for context-aware venue recommendation. In *Proceedings of SIGIR*.

Manotumruksa, J., Rafailidis, D., Macdonald, C., and Ounis, I. (2019). On cross-domain transfer in venue recommendation. In *Proceedings of ECIR*.

Mao, K., Zhu, J., Xiao, X., Lu, B., Wang, Z., and He, X. (2021). Ultragcn: Ultra simplification of graph convolutional networks for recommendation. In *Conference on Information & Knowledge Management*, pages 1253–1262.

Martinez, M. and Stiefelhagen, R. (2018). Taming the cross entropy loss. In *Proceedings of DAGM*.

Mauro, N., Ardissono, L., and Hu, Z. F. (2019). Multi-faceted trust-based collaborative filtering. In *Proceedings of UMAP*.

Medsker, L. and Jain, L. C. (1999). *Recurrent neural networks: design and applications*. CRC press.

Melville, P., Mooney, R. J., and Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *Proceedings of AAAI*.

Meng, Z., Liu, F., Shareghi, E., Su, Y., Collins, C., and Collier, N. (2022). Rewire-then-probe: A contrastive recipe for probing biomedical knowledge of pre-trained language models. In *Proceedings of ACL*.

Meng, Z., Liu, S., Macdonald, C., and Ounis, I. (2021). Graph neural pre-training for enhancing recommendations using side information. *arXiv preprint arXiv:2107.03936*.

Meng, Z., McCreadie, R., Macdonald, C., Ounis, I., Liu, S., Wu, Y., Wang, X., Liang, S., Liang, Y., Zeng, G., et al. (2020). Beta-rec: Build, evaluate and tune automated recommender systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 588–590.

Merris, R. (1994). Laplacian matrices of graphs: a survey. *Linear algebra and its applications*, 197:143–176.

Mohammad, H. R., Xu, K., Callan, J., and Culpepper, J. S. (2018). Dynamic shard cutoff prediction for selective search. In *Proc. of SIGIR*.

Muruganandam, S. and Srininvasan, N. (2017). Personalised e-learning system using learner profile ontology and sequential pattern mining-based recommendation. *Int. J. Bus. Intell. Data Min.*, 12(1):78–93.

Ning, X. and Karypis, G. (2011). Slim: Sparse linear methods for top-n recommender systems. In *Proceedings of ICDM*.

Ning, X. and Karypis, G. (2012). Sparse linear methods with side information for top-n recommendations. In *Recommender Systems Conference*, pages 155–162.

Noia, T. D., Ostuni, V. C., Tomeo, P., and Sciascio, E. D. (2016). Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Transactions on Intelligent Systems and Technology*, 8.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Palumbo, E., Rizzo, G., Troncy, R., Baralis, E., Osella, M., and Ferro, E. (2018). Translational models for item recommendation. In *Proceedings of ESWC*.

Pan, E. and Kang, Z. (2021). Multi-view contrastive graph clustering. In *Conference on Neural Information Processing Systems*.

Pang, B., Yang, M., and Wang, C. (2019). A novel top-n recommendation approach based on conditional variational auto-encoder. In *Pacific-Asia Conference on Knowledge Discovery & Data Mining*, pages 357–368.

Parambath, S. P., Anagnostopoulos, C., Murray-Smith, R., MacAvaney, S., et al. (2021). Max-utility based arm selection strategy for sequential query recommendations. In *Asian Conference on Machine Learning*, pages 564–579. PMLR.

Park, D. H. and Chang, Y. (2019). Adversarial sampling and training for semi-supervised information retrieval. In *Proceedings of WWW*.

Park, S., Kim, Y.-D., and Choi, S. (2013). Hierarchical bayesian matrix factorization with side information. In *International Joint Conference on Artificial Intelligence*, pages 1593–1599.

Park, S.-T. and Chu, W. (2009). Pairwise preference regression for cold-start recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 21–28.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8.

Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. (2018). Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104.

Pinkus, A. (1999). Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195.

Puthiya Parambath, S. A. and Chawla, S. (2020). Simple and effective neural-free soft-cluster embeddings for item cold-start recommendations. *Data Mining and Knowledge Discovery*, 34(5):1560–1588.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. (2020). Gcc: Graph contrastive coding for graph neural network pre-training. In *SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160.

Quadrana, M., Cremonesi, P., and Jannach, D. (2018). Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4):1–36.

Ramchoun, H., Idrissi, M. A. J., Ghanou, Y., and Ettaouil, M. (2016). Multilayer perceptron: Architecture optimization and training. *Journal of Interactive Multimedia and Artificial Intelligence*, 4.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. (2022). Recipe for a general, powerful, scalable graph transformer. *arXiv preprint arXiv:2205.12454*.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of EMNLP*.

Rendle, S. (2010). Factorization machines. In *IEEE International conference on data mining*, pages 995–1000.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). Bpr: Bayesian personalized ranking from implicit feedback. In *Conference on Uncertainty in Artificial Intelligence*, pages 452–461.

Rendle, S., Krichene, W., Zhang, L., and Anderson, J. (2020). Neural collaborative filtering vs. matrix factorization revisited. In *Recommender Systems Conference*, page 240–248.

Reynolds, D. A. (2009). Gaussian mixture models. *Encyclopedia of biometrics*, 741.

Ricci, F., Rokach, L., and Shapira, B. (2015). Recommender systems: introduction and challenges. In *Recommender systems handbook*, pages 1–34. Springer.

Robinson, J. D., Chuang, C.-Y., Sra, S., and Jegelka, S. (2020). Contrastive learning with hard negative samples. In *Proceedings of ICLR*.

Rossi, E., Kenlay, H., Gorinova, M. I., Chamberlain, B. P., Dong, X., and Bronstein, M. (2021). On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. *arXiv preprint arXiv:2111.12128*.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. (2018). Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR.

Sanz-Cruzado, J., Castells, P., Macdonald, C., and Ounis, I. (2020). Effective contact recommendation in social networks by adaptation of information retrieval models. *Information Processing & Management*, 57(5).

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of WWW*.

Shen, Y., Wu, Y., Zhang, Y., Shan, C., Zhang, J., Letaief, B. K., and Li, D. (2021). How powerful is graph convolution for recommendation? In *Conference on Information & Knowledge Management*, pages 1619–1629.

Sifaoui, A., Abdelkrim, A., and Benrejeb, M. (2008). On the use of neural network as a universal approximator. *Int. J. Sci. Tech. Control Comput. Eng*, 2:386–399.

Song, Y., Elkahky, A. M., and He, X. (2016). Multi-rate deep learning for temporal recommendation. In *Proceedings of SIGIR*.

Sun, J., Zhang, Y., Ma, C., Coates, M., Guo, H., Tang, R., and He, X. (2019). Multi-graph convolution collaborative filtering. In *IEEE International Conference on Data Mining*, pages 1306–1311.

Sun, L., Zhang, Z., Zhang, J., Wang, F., Peng, H., Su, S., and Philip, S. Y. (2021). Hyperbolic variational graph neural network for modeling dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4375–4383.

Tang, J., Hu, X., and Liu, H. (2013). Social recommendation: a review. *Social Network Analysis and Mining*, 3(4):1113–1133.

Tang, J., Wu, S., Sun, J., and Su, H. (2012). Cross-domain collaboration recommendation. In *Proceedings of SIGKDD*.

Thorat, P. B., Goudar, R. M., and Barve, S. (2015). Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4):31–36.

Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al. (2021). Mlp-mixer: An all-mlp architecture for vision. In *Proc. of NeurIPS*.

Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., and Bronstein, M. M. (2021). Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*.

Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Izacard, G., Joulin, A., Synnaeve, G., Verbeek, J., et al. (2021). Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*.

Tran, V.-A., Hennequin, R., Royo-Letelier, J., and Moussallam, M. (2019). Improving collaborative metric learning with efficient negative sampling. In *Proceedings of SIGIR*.

Valcarce, D., Landin, A., Parapar, J., and Barreiro, Á. (2019). Collaborative filtering embeddings for memory-based recommender systems. *Engineering Applications of Artificial Intelligence*, 85.

van den Berg, R., Kipf, T. N., and Welling, M. (2017). Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9.

Van Engelen, J. E. and Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, 109(2).

Vashishth, S., Sanyal, S., Nitin, V., and Talukdar, P. (2020). Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

Vasile, F., Smirnova, E., and Conneau, A. (2016). Meta-prod2vec: Product embeddings using side-information for recommendation. In *Recommender Systems Conference*, pages 225–232.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2019). Deep graph infomax. *ICLR (Poster)*, 2(3):4.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. *Advances in neural information processing systems*, 29.

Wan, M., Wang, D., Liu, J., Bennett, P., and McAuley, J. (2018). Representing and recommending shopping baskets with complementarity, compatibility and loyalty. In *Conference on Information & Knowledge Management*, pages 1133–1142.

Wang, C., Samari, B., and Siddiqi, K. (2018a). Local spectral graph convolution for point set feature learning. In *Proceedings of ECCV*.

Wang, H., Shi, X., and Yeung, D.-Y. (2015). Relational stacked denoising autoencoder for tag recommendation. In *AAAI Conference on Artificial Intelligence*, page 3052–3058.

Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., and Guo, M. (2019a). Multi-task feature learning for knowledge graph enhanced recommendation. In *Web Conference*, pages 2000–2010.

Wang, J., Huang, P., Zhao, H., Zhang, Z., Zhao, B., and Lee, D. L. (2018b). Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 839–848.

Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., and Zhang, D. (2017). Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of SIGIR*.

Wang, Q., Yin, H., Chen, T., Yu, J., Zhou, A., and Zhang, X. (2021a). Fast-adapting and privacy-preserving federated recommender system. *The VLDB Journal*, pages 1–20.

Wang, S., Hu, L., Wang, Y., He, X., Sheng, Q. Z., Orgun, M. A., Cao, L., Ricci, F., and Yu, P. S. (2021b). Graph learning based recommender systems: A review. *arXiv preprint arXiv:2105.06339*.

Wang, X., He, X., Cao, Y., Liu, M., and Chua, T.-S. (2019b). Kgat: Knowledge graph attention network for recommendation. In *SIGKDD international conference on knowledge discovery & data mining*, pages 950–958.

Wang, X., He, X., Wang, M., Feng, F., and Chua, T.-S. (2019c). Neural graph collaborative filtering. In *SIGIR Conference on Research and Development in Information Retrieval*, page 165–174.

Wang, X., Ounis, I., and Macdonald, C. (2021c). Leveraging review properties for effective recommendation. In *Proceedings of WWW*.

Wang, X., Xu, Y., He, X., Cao, Y., Wang, M., and Chua, T.-S. (2020). Reinforced negative sampling over knowledge graph for recommendation. In *Proceedings of WWW*.

Wang, Y.-X. and Zhang, Y.-J. (2012). Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on knowledge and data engineering*, 25(6):1336–1353.

Wen, Y., Guo, L., Chen, Z., and Ma, J. (2018). Network embedding based recommendation method in social networks. In *Proceedings of WWW*.

Willmott, C. J. and Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82.

Wong, C.-M., Feng, F., Zhang, W., Vong, C.-M., Chen, H., Zhang, Y., He, P., Chen, H., Zhao, K., and Chen, H. (2021). Improving conversational recommender system by pretraining billion-scale knowledge graph. In *Proceedings of ICDE*.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019a). Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871.

Wu, G., Volkovs, M., Soon, C. L., Sanner, S., and Rai, H. (2019b). Noise contrastive estimation for one-class collaborative filtering. In *Proc. of SIGIR*.

Wu, J., Wang, X., Feng, F., He, X., Chen, L., Lian, J., and Xie, X. (2021). Self-supervised graph learning for recommendation. In *SIGIR Conference on Research and Development in Information Retrieval*, pages 726–735.

Wu, L., Li, J., Sun, P., Hong, R., Ge, Y., and Wang, M. (2020a). Diffnet++: A neural influence and interest diffusion network for social recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Wu, Y., Macdonald, C., and Ounis, I. (2020b). A hybrid conditional variational autoencoder model for personalised top-n recommendation. In *SIGIR on International Conference on Theory of Information Retrieval*, pages 89–96.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020c). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1).

Xiao, C., Xie, R., Yao, Y., Liu, Z., Sun, M., Zhang, X., and Lin, L. (2021). Uprec: User-aware pre-training for recommender systems. *CoRR*.

Xie, X., Sun, F., Liu, Z., Gao, J., Ding, B., and Cui, B. (2020). Contrastive pre-training for sequential recommendation. *arXiv*.

Xie, Z., Liu, C., Zhang, Y., Lu, H., Wang, D., and Ding, Y. (2021). Adversarial and contrastive variational autoencoder for sequential recommendation. In *Web Conference*, pages 449–459.

Xin, X., Karatzoglou, A., Arapakis, I., and Jose, J. M. (2020). Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 931–940.

Xinyi, Z. and Chen, L. (2018). Capsule graph neural network. In *Proceedings of ICLR*.

Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P., Ahmed, J., and Overwijk, A. (2020). Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *Proceedings of ICLR*.

Yang, X., Guo, Y., Liu, Y., and Steck, H. (2014). A survey of cf based social recommender systems. *Comput Commun*, 41.

Yang, Z., Ding, M., Zou, X., Tang, J., Xu, B., Zhou, C., and Yang, H. (2022). Region or global a principle for negative sampling in graph-based recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Yi, H.-C., You, Z.-H., Huang, D.-S., and Kwoh, C. K. (2022). Graph representation learning in bioinformatics: trends, methods and applications. *Briefings in Bioinformatics*, 23(1):bbab340.

Yih, W.-t., Toutanova, K., Platt, J. C., and Meek, C. (2011). Learning discriminative projections for text similarity measures. In *Proceedings of CoNLL*.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of SIGKDD*.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. (2020). Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823.

Yu, J., Yin, H., Li, J., Gao, M., Huang, Z., and Cui, L. (2020). Enhance social recommendation with adversarial graph convolutional networks. *IEEE Transactions on Knowledge and Data Engineering*.

Yu, J., Yin, H., Xia, X., Chen, T., Cui, L., and Hung, N. Q. V. (2022a). Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *SIGIR conference on research and development in information retrieval*.

Yu, J., Yin, H., Xia, X., Chen, T., Li, J., and Huang, Z. (2022b). Self-supervised learning for recommender systems: A survey. *arXiv preprint arXiv:2203.15876*.

Yu, J., Yin, H., Xia, X., Cui, L., and Nguyen, Q. V. H. (2022c). Graph augmentation-free contrastive learning for recommendation. In *Proceedings of SIGIR*.

Yu, X., Hu, Q., Li, H., Du, J., Gao, J., and Sun, L. (2022d). Cross-domain recommendation based on latent factor alignment. *Neural Computing and Applications*, 34(5).

Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J. M., and He, X. (2019). A simple convolutional generative network for next item recommendation. In *International Conference on Web Search & Data Mining*, pages 582–590.

Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. (2019). Graph transformer networks. *Advances in neural information processing systems*, 32.

Zhang, C., Song, D., Huang, C., Swami, A., and Chawla, N. V. (2019a). Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 793–803.

Zhang, S., Yao, L., Sun, A., and Tay, Y. (2019b). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, 52:1–38.

Zhang, S., Yao, L., Sun, A., and Tay, Y. (2019c). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Survey*, 52.

Zhang, W., Chen, T., Wang, J., and Yu, Y. (2013). Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of SIGIR*.

Zhang, Y., Ai, Q., Chen, X., and Croft, W. B. (2017). Joint representation learning for top-n recommendation with heterogeneous information sources. In *Conference on Information & Knowledge Management*, pages 1449–1458.

Zhang, Y., Wang, X., Shi, C., Liu, N., and Song, G. (2021). Lorentzian graph convolutional networks. In *Proceedings of the Web Conference 2021*, pages 1249–1261.

Zhang, Z. and Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. In *Proceedings of NeurIPS*.

Zhao, C., Li, C., Xiao, R., Deng, H., and Sun, A. (2020a). Catn: Cross-domain recommendation for cold-start users via aspect transfer network. In *Proceedings of SIGIR*.

Zhao, F. and Guo, Y. (2017). Learning discriminative recommendation systems with side information. In *International Joint Conference on Artificial Intelligence*, pages 3469–3475.

Zhao, H., Wei, L., and Yao, Q. (2020b). Simplifying architecture search for graph neural network. *arXiv preprint arXiv:2008.11652*.

Zhao, S., Sinha, A., He, Y., Perreault, A., Song, J., and Ermon, S. (2021). Comparing distributions by measuring differences that affect decision making. In *Proceedings of ICLR*.

Zhao, T., McAuley, J., and King, I. (2014). Leveraging social connections to improve personalized ranking for collaborative filtering. In *Proceedings of CIKM*.

Zheng, J., Cai, F., Chen, H., and de Rijke, M. (2020a). Pre-train, interact, fine-tune: A novel interaction representation for text classification. *Information Processing & Management*, 57(6).

Zheng, Y., Gao, C., Li, X., He, X., Li, Y., and Jin, D. (2020b). Disentangling user interest and popularity bias for recommendation with causal embedding. *arXiv preprint arXiv:2006.11011*.

Zhou, C., Ma, J., Zhang, J., Zhou, J., and Yang, H. (2021). Contrastive learning for debiased candidate generation in large-scale recommender systems. In *Proceedings of SIGKDD*.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.