Master thesis on Sound and Music Computing
Universitat Pompeu Fabra

Heterogeneous Graph Neural Network Music Recommendation

Dean Cochran

Supervisor: Lorenzo Porcaro

Co-Supervisor: Emilia Gómez

August 2022

Universitat Pompeu Fabra Barcelona

Master thesis on Sound and Music Computing
Universitat Pompeu Fabra

Heterogeneous Graph Neural Network Music Recommendation

Dean Cochran

Supervisor: Lorenzo Porcaro

Co-Supervisor: Emilia Gómez

August 2022

Universitat Pompeu Fabra
Barcelona

Contents

Dedication

I would like to dedicate this work to my family for their endless support for me while I participated in this program abroad.

Acknowledgement

During my experience as a member of the Music Technology Group at the Universidad de Pompeu Fabra, I would like to emphasize my sincerest gratitude to my advisor Lorenzo Porcaro, and co-advisor Emilia Gómez for steering my thesis into a cohesive manuscript, spending countless hours providing intelligent feedback, and allowing me to develop as a student. I plan to use all the advice provided to me throughout the next phase of my life and wish you both all the best.

Additionally, I acknowledge the rest of the Music Technology Group for their generously provided resources and education. As well as Filip Korzeniowski and Fabien Gouyon from Pandora for their time spent discussing Graph Based learning with me. Francesco Fabbri for his time discussing technical aspects of Graph-Based learning applied to my project. Markus Schedl and the rest of the team of researchers at JKU for their conducted research on the LFM-1b data set. Finally, the communities of Pytorch Geometric, Deep Graph Library, and OpenHGNN for the numerous answered forum questions that were so helpful for the implementation of this project.

Abstract

While personalized music recommendation has changed the way many users listen to music. Graph Neural Networks have also become a state-of-the-art machine learning practice for predicting recommendations. The LFM-1b is a data set contains a high density of information to address the sparsity issues of similar data sets like the The Million Song Data set. Additionally, as the provided information of the data set can be represented as a heterogeneous graph, their is a lot of available opportunities to evaluate the important connections that users have with their favorite tracks, albums, artists, and even genres. However, as the music recommendation system research community has witnessed the promising capabilities of graph neural networks, and as the limitations of a not having a publicly available, large scale, high in density data set as been alleviated, the LFM-1b data set is underutilized amidst the Music Information Retrieval community for graph based machine learning research.

This thesis will dive deep into the specifics required to utilize the LFM-1b data set for heterogeneous graph neural network research. With a primary focus on providing an machine learning recommendation system implementation, an analysis on the models' capabilities to provide recommendations to users whilst understanding user listening preferences is to be evaluated. The contributions of this thesis will include a LFM-1b data set loading class for the Deep Graph Library framework in Pytorch, as well as an implementation of several link prediction graph neural network models, to validate the LFM-1b data set's applicability for music recommendation system machine learning research.

Keywords: Graph Neural Networks; Link Prediction; Music Recommendation; Last.fm;

Chapter 1

Introduction

As technology continues to advance, there has been a dramatic explosion of digital content that is now accessible online. Due to the sheer volume of accessible information, it is often overwhelming for users to find the specific information they search for. To alleviate the tedious act of filtering through online digital multimedia content, algorithms have been created to match, score and rank the relevance of the content concerning the searching user, creating Recommendation Systems (RS) [1].

Though task-specific, the relevance score for a single piece of content, measured by a RS, is based on the scenario the RS is used for. Recommendation systems are powerful algorithms that must be monitored and structured properly. For cases like the YouTube[2], Netflix[3], and Pinterest [4], the recommendation algorithm, is actually the service the company provides.

These companies also utilize RS to advertise or tailor suggestions for their products, to increase user retention and engagement, hence increasing company revenue [5]. Therefore, a result of the billions of user interactions, recommend by a RS model, RS are seen as potentially dangerous as they can influence users poorly. This idea that RS can mislead users has led the academic community to research different ways to mitigate bias, promote fairness, and improve the privacy in RS [6]. The goal is to produce not only fair and unbiased music recommendations, but to provide users with relevant and interesting content. The main objective of RS services in fact, is to retain the loyalty of their users [7] by offering users a service that is more engaging than the RS services provided by the other companies.

The research field of Music RS is largely supported by the academic community of Music information retrieval (MIR), as well as the larger Recommendation System community. These communities have common interests, which have often produced useful research that modern-day music streaming services utilize to form better music recommendations. Notably, Music RS services utilize a barrage of different methods to collect explicit and implicit data from user interactions. Explicit feedback includes information on when users like, rate, or save a piece of content. Since explicit feedback often is tedious to collect, implicit feedback, or data collected on user listening events, clicks, and skips are collected. As the user interacts with the platform more frequently, the preferences elicited from the listening behaviors of the user can be inferred.

Often this inferred information is stored in a database where a user will be then labeled as having a connection to a piece of content. This information does not indicate an explicit opinion from the user, rather as the user interacts with the platform more frequently, implicit preferences from a user can be used to predict their behavioral habits, interactions, and even emotions over time.

1.1 Contributions

The way music streaming services utilize explicit and implicit information also comes in a variety of forms, most commonly found are the content (CB) and or collaborative (CF) based filtering methods. These have been utilized for decades and have been proven to be effective for finding content that is relevant for a given user.

As RS have improved significantly in the last decade, users are often given personalized recommendations based on more than just their implicit feedback. Modern day RS often use newer methodologies, complex mathematics, and machine learning to alleviate the pitfalls of traditional content and collaborative filtering methods such as user interaction modeling, the cold-start problem, robustness, and the ability to explain why recommendations are made [8].

Most notably for this thesis, the state of the art recommendation system approach involves using graph-based deep learning. This unique approach utilizes methods such as nonlinear transformations, representation learning, and sequence modeling which are widely accepted by the scientific community. Though these method are highly effective, there are some limitations when implementing these methods due to the "black-box" phenomenon and the lack of mathematical transparency in complex computations made by deep learning algorithms [8]. Uniquely however, graph-based deep learning holds the natural capability to represent implicit user interactions, in a way that has recently been able to show major performance improvements for RS. Continuous studies on have allowed graph-based machine learning models to be widely used for their promised capabilities [9]. In fact, graph-based learning has been capable of influencing many other fields of research; from traffic prediction [10], to prediction a molecular structures [11]. Additionally, there have been numerous graph-based recommendation methods that have demonstrated their applicability in the music industry [12][13][14].

A multimedia specialist and professor at Johannes Kepler University, Markus Shedl, offers a modern perspective on Music RS by mentioning, "even though the number of the respective publications are increasing, neural network architectures are still surprisingly sparsely adopted for Music RS" [15]. Furthermore, discussions of the lack of established multi-modal data sets like the Million Song data set (MSD), and the lack of diversity within available data sets, makes the ability to reproduce published experiments, or even compare models quite challenging [16]. This high-level overview and structural breakdown of some modern Music RS limitations, identifies the fundamental need to understand how new graph neural network algorithms can improve the performance of Music RS. For this reason, the objectives of the thesis are outlined in the following sections.

1.1.1 A LFM-1b data set loader

The MIR community has several available research data sets, however many of the collections are not suitable for music recommendation research [17]. This leaves MIR researchers with a small collection of data sets [18] [19] [20]. Due to the lack of variety and diversity from within the limited amount publicly available data sets, many researchers are not willing to utilize these collections for scientific research. Rather than using an 'off the shelf' collection, much of the published Music RS research generates their own data set collections. This is particularly troublesome and as mentioned before, limiting the reproducible research available in the field. For this reason, there is need to draw attention to the LFM-1b data set, a particular interesting collection of user listening histories gathered from the last.fm API [21].

The LFM-1b is a collection of more than one billion listening events, intended to be used for various music retrieval and recommendation tasks. Specifically, LFM-1b provides more types of user listening behavior information that many other data sets do not provide. Music RS research utilizing a large collection like LFM-1b would encourage other researchers understand the importance of such a data set for studying graph-based Music RS, as well as increase the amount of reproducible research in the field. Notably, there is not much research using the LFM-1b data set in pairing with Graph Neural Networks. Therefore, this thesis implements customizable data loading class to ease the process of reproducing graph neural network experiments on the LFM-1b data set.

1.1.2 Graph Based Deep Learning Link Prediction

As the development of deep learning in recommendation has seen a massive increase in novel graph-based algorithms, there has not been ample support by the MIR community to challenge, discuss or implement these deep learning algorithms for Music RS [15] [9]. As mentioned, there is not much support for Music RS deep learning research, so utilizing deep learning to contribute to the slowing research should be seen as necessary to maintain the topic's relevancy in the coming years.

Among the novel recommendation algorithms, specifically within the field of deep learning, graph-based machine learning methods have been shown to perform competitively against more traditional recommendation system methods [22]. Therefore, with the knowledge of deep learning in Music RS slowing [15], and the success of graph-based machine learning, this thesis will aim to deploy state-of-the-art Graph Neural Network models to recommend undiscovered music to users in the LFM-1b data set.

Hence this thesis concludes by implementing a recommendation algorithm utilizing graph-based link prediction machine learning models and evaluates the performance of the novel algorithms. These conclusions will offer further insight into not just the LFM-1b data set but will help researchers understand how implicit user preferences can be learned to create interesting recommendations. Additionally, there will be a discussion on evaluations of the proposed recommendation system. This paired with some of the results provide additional insights into measurements like diversity, novelty, and the coverage of the recommendation model.

1.2 Structure of Report

Throughout this body of work, the objectives of the thesis will be addressed in a sequential structure offering insight and understanding into the world of graph-based learning for Music RS.

1.2.1 State of the Art

The state of the art will provide readers with the relevant and necessary information to understand the research presented throughout the thesis. Emphasizing recent RS research findings, a discussion outlining Music RS data sets and limitations of the Music RS field will be shared. As the state of the art continues, the introduction to graph-based deep learning will allow this thesis to introduce the fundamental theory to achieve the objectives. Particularly outlined in this section, readers will be able to interpret not only how graph algorithms work, but how they are mathematically computed. This syntax is necessary for the reader to understand deep learning algorithms in the context of Music RS, which will be utilized throughout the thesis.

1.2.2 Methodology

With the understanding of the previous sections, the methodology section will begin by framing the objectives into individual studies. The data set utilized for the thesis will be discussed much more in detail, including the advantages, and limitations. The methodology then will outline the techniques utilized to implement state-of-the-art graph-based machine learning models. This will be elaborated on as the implementation begins to discuss the technical specifics of each topic, and by the end of the methodology, the reader will understand the formal actions taken to provide tested results.

1.2.3 Results

As the requirements to understand the results section of the thesis should be made clear. The results section will able the discussed topics and models presented through the thesis as they pertain to the analysis of link prediction within the LFM-1b data set. The best-performing models will be selected to compute recommendations for all users within the LFM-1b data set. Once the recommendations have been acquired, this analysis should be able to provide a more comprehensive review of the performance capabilities of using graph-based deep learning algorithms to recommend music content to users in the data set.

1.2.4 Conclusion

To provide insights into what the results of the evaluations offer, and to interpret them for meaningful information, the conclusion will analyze the findings of the thesis. Upon analyzing, a discussion on the findings will be presented in hopes to clarify the results which may not be so intuitive to comprehend. Finally, within the discussion, the limitations as well as the successes of the thesis will be addressed to support the proceedings and motivation of the thesis.

Chapter 2

State of the Art

Implementing RS to simplify choice amidst billions of possible decisions that a user's face is no new problem. These technologies to filter overwhelming amounts of information or forecast a user interactions with a particular piece content have existed long before digital recommendation. Traditionally, RS have utilized Content-Based (CB) and Collaborative-Based (CF) filtering methods to recommend items to users by utilizing past user interactions. However, these traditional RS methods have been discovered to be constrained by the scientific community. These common recommendation constraints include the cold start problem (when a system wants to recommend items to a user with no profile information) and data sparsity (when a system has many users and many items, but little information to compute relevant recommendations). While these constraints are fundamental to understanding RS limitations, the scientific community have developed machine learning techniques that can alleviate these constraints while providing accurate and relevant recommendations. In the following section an observational analysis of the modern approaches for Music RS and Graph Neural Networks will be presented. Afterwards a in deep dive into the architecture of Graph Neural Networks is necessary to comprehend the methodology of this thesis.

2.1 Music Recommendation Systems (Music RS)

Particularly, due to increased user consumption on musical streaming platforms, giving users an immersive personalized experience is a necessity [23]. When considering the various industries that utilize RS, it is important to observe how the scope of Music RS is different from other multimedia RS. In comparison to other industry domains like movie streaming or e-commerce, music has a short consumption duration, with billions of ways for users to listen to music. The time it takes for a user to interact with a track, album, or artist, paired with the abundance of available content, implies that recommending a piece content that is not perfectly relevant the user's preference, will not affect the user experience detrimentally. Even repeated recommendations can be appreciated by the user [24].

These unique characteristics of Music RS stand out from other industry-specific RS and have encouraged research in the fields like sequence-based recommendation, which leverages the time a user listens to content as a primary indicator of what the user would like to be recommended [25]. Another encouraged topic called session-based recommendation, leverages the style of each listening period as a primary indicator of what the user would like to be recommended [26].

2.1.1 Machine Learning Music Recommendation

To address the modern approaches using Machine Learning in the context of Music RS, this section will highlight past machine learning research relevant to the thesis and the relevant data sets that are available for research. What is particularly useful to understand in context of the available data sets, is that due to many user privacy restrictions, a lot of the industrial RS data sets are not able to be made public. As a result of this, there is a lot of advocacy for publicly available Music RS data sets. One of the older and more popular public Music RS data sets, The Million Song Data set, offers researchers a large collection of user interaction information [16].

However, much of the user interaction information is limited as the MSD has sparsity issues. From within the MIR community in 2017, the LFM-1b data set was published for public usage [21]. What was notably different about this data set was that the LFM-1b collection offers over one billion timestamped user listening events, well over twenty times the amount that MSD offers. Specifically, the data set was properly tailored for deep learning models which requires massive volumes of data for training. In short, the LFM-1b enables large scale experimentation in Music RS research.

Whilst deep learning is surely not a new topic in Music RS, a significant increase in graph-based machine learning papers has shown their relevancy within their selective topic of choice. Topics like modeling expressive piano performance, measuring similarity between artists, and many others have made great leaps into a new collection of machine learning studies as General Neural Networks have become more popular [27] [28].

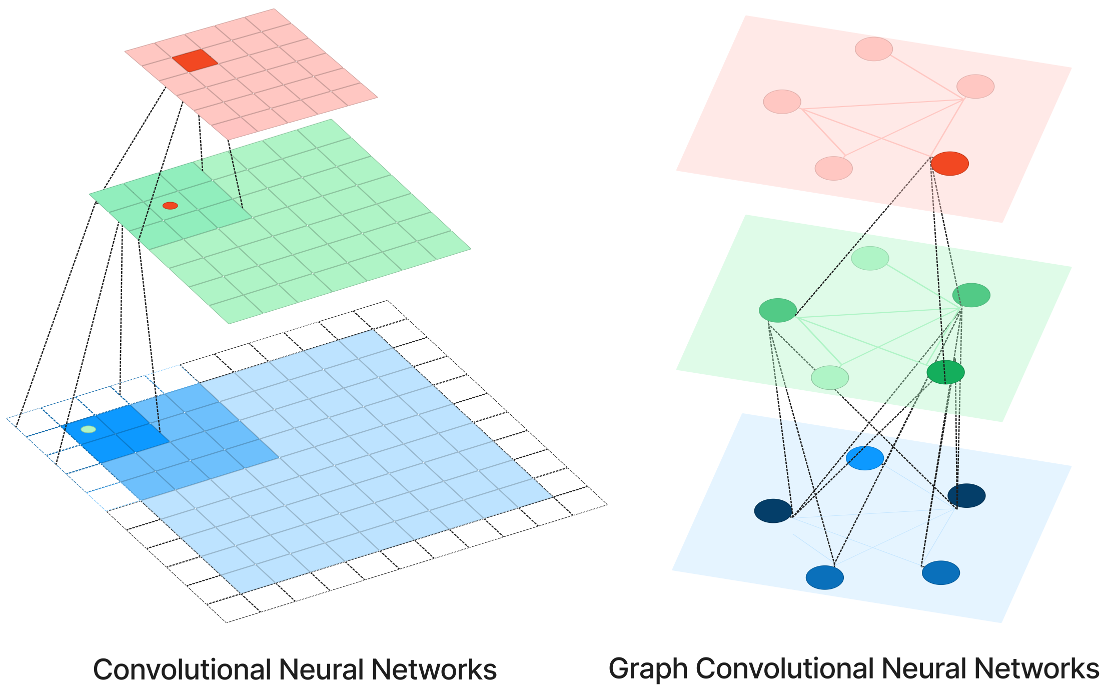

Convolutional Neural Networks
Graph Convolutional Neural Networks

Figure 1: A visual comparison of Convolutional Neural Networks (Left) and Graph Convolutional Neural Networks (Right)

With the rise of the popular Graph Convolutional Networks (GCN) in 2017 (Depicted in Figure 1), there are a lot of new research findings on the topic of music recommendation with graph-based machine learning [29] [30]. These newer methods focus on representation learning, a method to compute vector representations of nodes that can be measured in an embedding space [22].

Utilizing node representations for the input data of Music RS has been established as a common practice [31]. Practices for Graph Neural Networks are constantly evolving, Galvan's "Contributions to Representation Learning with Graph Auto encoders and Applications to Music Recommendation" have shed more light on this specific matter [32]. As discussed in his work too, there have been major improvements in learning the node representations to assist downstream tasks like link prediction and recommendation [33] [34].

2.2 Graph Neural Networks (GNN)

Due to the relevancy of GNNs within the field of recommendation, this thesis will utilize graph neural networks to compute predictions using complex mathematics. Therefore, is worth devoting some time discussing the fundamentals of GNNs in the context of music recommendation.

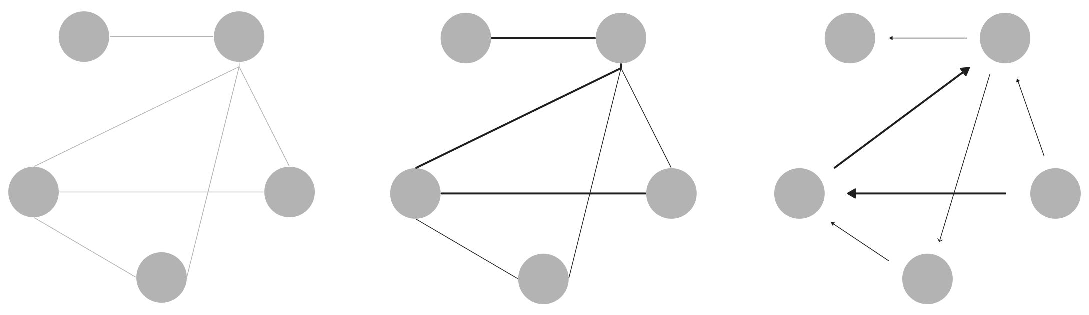


Figure 2: Different ways to represent a graph. A undirected, unweighted graph (Left). A undirected and weighted graph (Center). A directed and weighted graph (Right)

Using Figure 2 as a reference, a graph can be represented as a set of edges as nodes. Graphs can also be undirected or directed. In an undirected graph, nodes that are connected share a relationship, represented as an edge. In a directed graph, connected nodes do not share the same edge, rather if two nodes are connected in a directed graph, there is one source node connected to one destination node. Another way of identifying graphs is by whether they are unweighted or weighted.

In an unweighted graph, all edges have the same label. In a weighted graph, each edge is associated with a label value representing its weight. This fundamental graph structure can model a variety of real-world scenarios quite easily. However, in the homogeneous graph description we have discussed so far, all nodes have the same type, as do all the edges. In a heterogeneous graph, nodes can be of a different type, and there can be different types of edges existing between them.

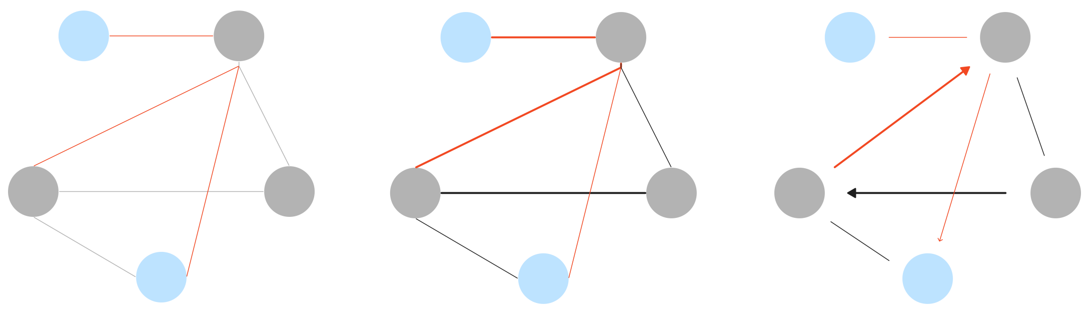


Figure 3: Different ways to represent a heterogeneous graph. A undirected, unweighted heterogeneous graph (Left). A undirected and weighted heterogeneous graph (Center). A directed and weighted heterogeneous graph (Right)

Many real-world problems can be represented as a heterogeneous graph, in fact this thesis and its methodological approach use a heterogeneous graph. In heterogeneous graphs, shown in Figure 3, edges are represented as triples with a source node, relation type, and a destination node. To relate this to the thesis, users have a edge connection with their favorite tracks, albums, and artists. Utilizing heterogeneous graphs instead of homogeneous graphs has led to a subdivision of graph-based deep learning research. Where this subdivision studies how the additional information provided in heterogeneous graphs can be used to improve the performance of GNNs [35]. The research field of statistical relational learning (SLR) applies directly to this subdivision [36].

2.2.1 Graph Based Machine Learning Tasks

With the understanding of the requirements to create a graph, we can utilize the existence of node, edge, full graph, and the graph's structural information to form inputs for a graph-based neural network. A common way to solve these problems is to embed the graph nodes into a high-dimensional embedding space. Once in this "embedding" or latent space, the nodes representation must capture task-specific information such that the neural network may predict some outcome.

Once a graph and its components have been formed with their embedding representations, the information in the nodes, edges, as well as structural information of the full graph can be utilized for a variety of different prediction tasks.

"Node-level" prediction tasks have the objective to predict the classification of a particular unseen node in the graph (ex. predicting a user's height/weight). To accomplish this, a classifier or regression model can be used to determine each of the labels of the nodes, given the labels of other nodes and their interconnected edges.

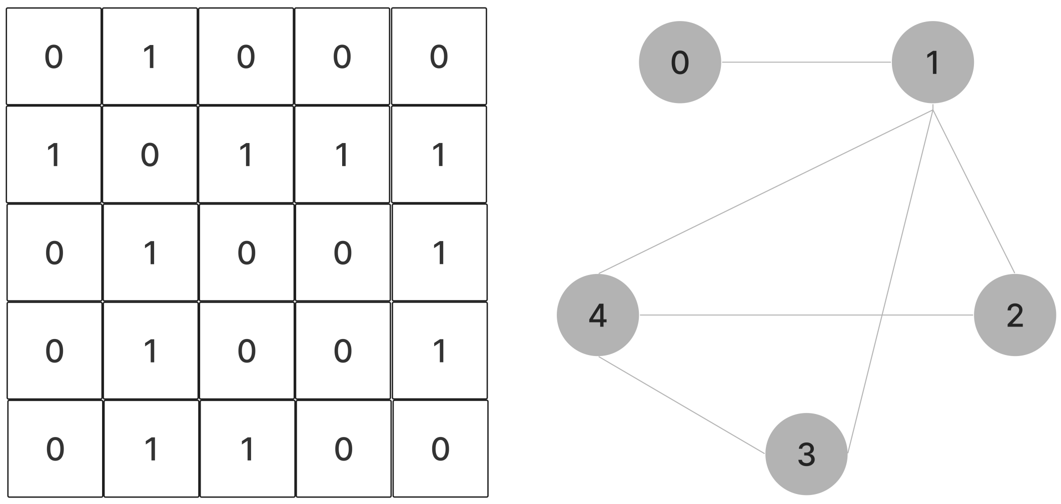
"Edge-level" prediction tasks have the goal of predicting interactions between two given nodes in the graph (like recommending undiscovered tracks to users). To accomplish this, you can utilize the two nodes and the edge connecting them as inputs into a prediction function to provide the likelihood of the existence of the edge occurring between the two nodes (aka link prediction).

For "graph-level" tasks, the goal is to predict the property of an entire unseen graph (ex. predicting a molecules toxicity based on its structure). To accomplish this, the embedding from each of the resulting nodes, edges, and their structural representations are used in an aggregation to determine a label of a full graph representation [37].

2.2.2 Understanding GNNs

To dive into the specifics of a GNN model, we will look at the basic form of GNNs. Assuming there exists an unweighted and undirected graph such that the adjacency matrix is binary and symmetric with cells of ones and zeros according to the connectivity of the nodes.

With the information formed by the adjacency matrix (example shown in Figure 4),



0	1	0	0	0
1	0	1	1	1
0	1	0	0	1
0	1	0	0	1
0	1	1	0	0

Figure 4: An example adjacency matrix representing the edges between nodes in a graph (Left). The resulting graph representation of the provided Adjacency Matrix (Right)

the next step is to use the matrix to update the node representations by applying an operation, for each layer in the neural network.

$$H^{'} = \sigma(AHW) \tag{2.1}$$

As seen in above equation 2.1, $H^{'}$ is the updated node representations, H is the previous node representations, A is the adjacency matrix, A is a learn able weight matrix, and $\sigma(\circ)$ is a nonlinear activation function. Notably, in this high-level overview, the information of each node is not retained in the transformation that occurs with this operation. Therefore, to adjust for this we can utilize a simple matrix summation to add what is commonly referred to as self-loops.

$$H^{'} = \sigma(A^{'}HW) \tag{2.2}$$

As seen in equation 2.2, $A^{'}$ is the adjacency matrix with added self-loops such that $A^{'} = A + I$ (example shown in Figure 5), where I is the identity matrix. There do exist some known limitations for this operation as well. When multiplying the

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

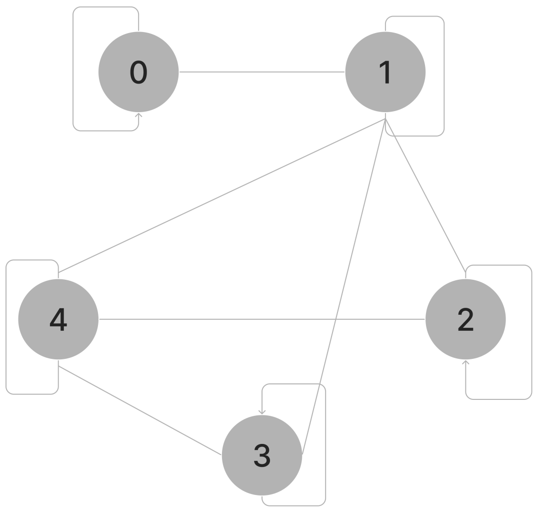


Figure 5: An adjacency matrix and graphical representation with self-loops added.

adjacency matrix for each hidden layer in the neural network, the scale of the features increases rapidly. Therefore, normalization of the features can be applied through the multiplication of the degree matrix [37].

$$H^{'} = \sigma(D^{'-1}A^{'}HW) \tag{2.3}$$

As seen in equation 2.3, $D^{'}$ is the degree matrix of $A^{'}$. There are other ways to normalize the adjacency matrix in more expressive ways. One example is the approach provided by Kipf and Welling in their research on the Graph Convolutional Networks; known as symmetric normalization [29].

$$\mathbf{H}^{'} = \sigma(D^{'-1/2}A^{'}D^{'-1/2}HW) \tag{2.4}$$

Upon each layer in the neural network, the update operation (Equation 2.4) will be called on all nodes in the graph. This is similar in concept to a traditional neural network, as the output from the first layer is the input to the second layer. The difference is that there is an additional processing steps, where each target node updates its feature representations, by aggregating the representations of its neighbors.

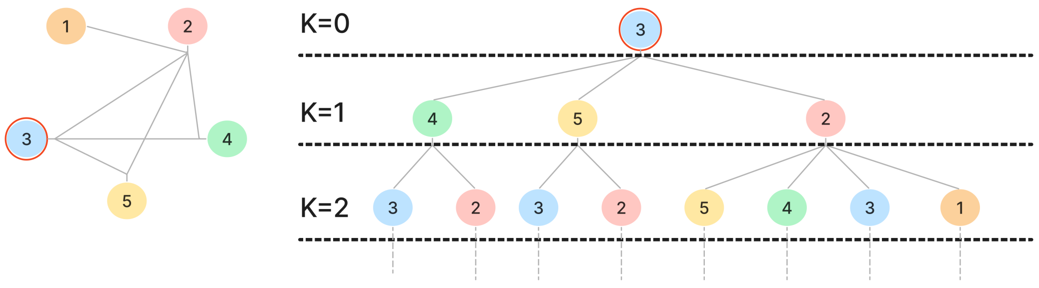


Figure 6: A target node's expanding neighborhood size of a graph neural network with K layers

This leads to some very expressive node embedding representations as the number of GCN layers corresponds to how far the information of a node can travel between neighbors. A GNN with multiple layers as seen in Figure 6, has a target node, who's features can only be incorporated into the features of all nodes that are k hops away from the target node, where k is the number of layers in the GNN model. Some limitations of this model include the assumption of the adjacency matrix being binary and symmetric, limiting the GCN model as it only indirectly supports edge features. This means that GCNs are not suitable for propagating information between nodes of different types, or pairs of nodes whose edge connections are of different types. Heterogeneous GNNs (HGNNs) are the solution to this limitation and will be more formally introduced in the later sections.

Chapter 3

Methods

From the understanding of modern graph-based deep learning Music RS, as well as the known limitations that are associated with these approaches as they relate to Music RS, this chapter will address the technical details to form the methodology of the thesis. Firstly, this methodology section outlines the data set that is utilized for the thesis. Recalling the state-of-the-art chapter, this methodology discusses the HGNN models selected to be used. Additionally, since the training process for link prediction is unique for graph-based machine learning, the chapter continues discussing how the selected models are to be trained. In doing so, the chapter addresses how recommendation is achieved with the implementation of different link prediction models, and how it can be utilized in the context of track, album, or artist recommendation for users in the LFM-1b data set. Finally, since these approaches need to be justified, the chapter provides insights on the evaluation metrics used interpret the performance of the thesis objectives.

3.1 DGL LFM-1b Data set loader

Uniquely, at such a large volume the LFM-1b data set provides more information that can be utilized in a deep learning scenario than most Music RS data sets. The LFM-1b data set incorporates user listening behaviors for tracks, albums, and artists. The significance of incorporating this information in large deep learning graphs to discern the impact of incorporating listening behaviors for tracks, albums, and artists for Music RS in graph neural networks is not well studied [38] [21].

Implementing a custom data loader, that can be utilized by graph-based deep learning communities, increase the necessary exposure the Music RS community needs to stay relevant in the field of graph-based deep learning, and encourage the music recommendation research community to understand the impact of utilizing the information that is collected in the LFM-1b data set.

3.2 Frameworks for GNNs

The familiar machine learning frameworks of Pytorch and TensorFlow only optimize for workflows with fixed-size graphs. These assumptions do not hold for GNN training. Additionally, for neighbor sampling, the neighbors used for each epoch of batch training processes are going to be different (more on this later). Therefore, there are specialized frameworks for deep learning with GNNs.

Specifically open-source frameworks like PyTorch Geometric, Deep Graph Library, Networkx, and many others have been in development in the last decade [39] [40] [41]. The frameworks facilitate the steep learning requirements for implementing deep learning GNN functionalities like message passing, aggregation, update operation, sampling, and data management. These libraries are constantly being updated as well as utilized by the world's most renowned GNN researchers. For the given objective, a custom in-memory Deep Graph Library (DGL) data set loader would properly expose the MIR community to one of the most popular graph-based deep learning frameworks. Of the available data sets in the GNN frameworks, none utilize the LFM-1b data set.

3.2.1 Data Loading

Before outlining the implementation concerning the DGL framework, it is important to understand the demographics and structure of the LFM-1b data set which can be downloaded at the publishing website's location [21]. Inside the downloadable zip folder there are text files corresponding to the following information: users, artists, albums, tracks, and the one billion listening events. Additionally, the data set can be downloaded with the accompaniment of the LFM-1b UGP data set which provides genres of the artists, and users. The authors of the data set have published a distributional analysis on the website which is useful for those interested in the specific features and structure of the physical data set.

As discussed in the state of the art, the Deep Graph Library (DGL) provides a framework for computing graph-based machine learning tasks on large heterogeneous graphs. As the documentation for DGL describes, there are customizable classes made for custom in-memory data sets. These classes include functions like $load()$, $save()$, and a $process()$ to load the data into memory. The $save()$ and $load()$ functions simply work by reading and writing the final representations of the compiled heterogeneous graph to and from memory.

When the $process()$ function is called a large number of operations are required to read through the LFM-1b data set and compile one singular $HeteroGraph$ data object (depicted in Figure 7). Specifically, these processes can be divided into three parts: compiling graph, node, and edge data.

3.2.2 Compiling DGL Graph Data

To load a heterogeneous graph using the DGL framework, it is required to collect a dictionary of every edge index that exists in the graph. This information is not structured like a typical adjacency matrix due to the large storage space that it requires to compile an adjacency matrix of billions of edges. An adjacency list rather is utilized because it is not so harsh on memory (see Figure 8).As a technical note on the structural representation of the graph, there is not an edge connection provided for connecting tracks to albums. The justification for this resides in the structure of the LFM-1b files. This is particularly not computed since the information of track and album occurrences can only be evaluated through filtering of the listen events file.

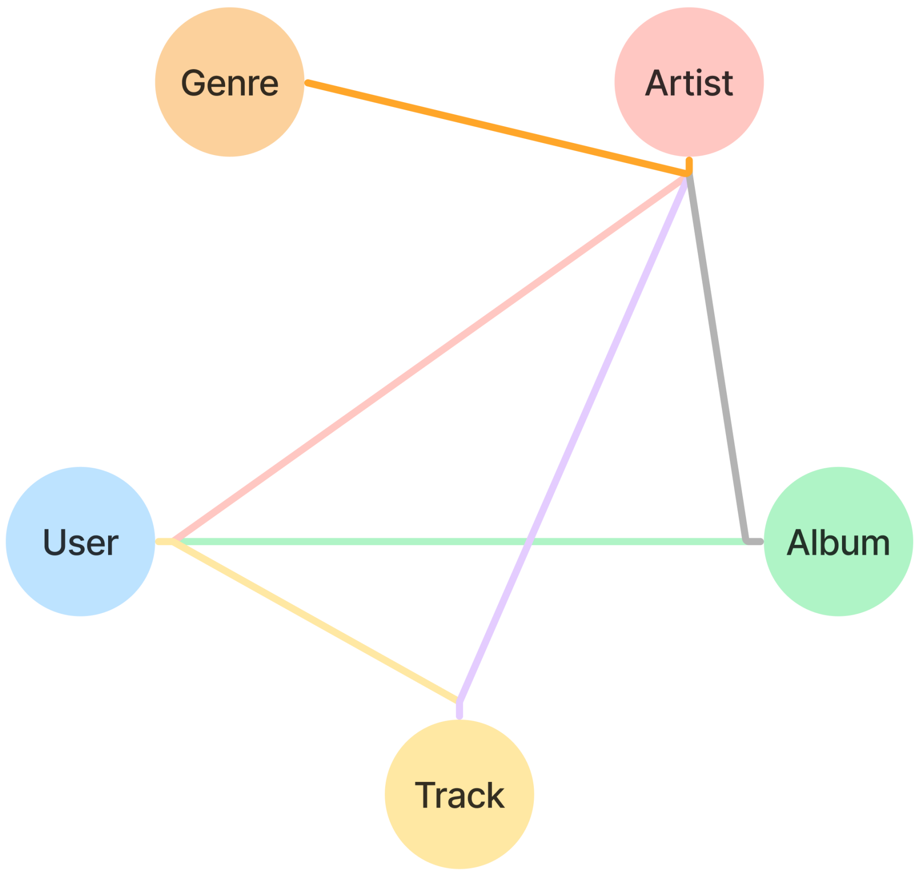


Figure 7: The schema of the resulting heterogeneous graph

Once this step is finished, the DGL native *HeteroGraph* object should hold all the necessary information to form a heterogeneous LFM-1b graph with the user, genre, artist, album, and track nodes. However, due to the nature of this thesis, the capability to select a subset of node types is included in the final DGL data loading class.

3.2.3 Compiling DGL Node Data

Once the graph data has been loaded into memory, the next task before the processing function finishes is to add features to the nodes and edges of the graph. This step can be done in a variety of different ways. Notably, the LFM-1b only pro-

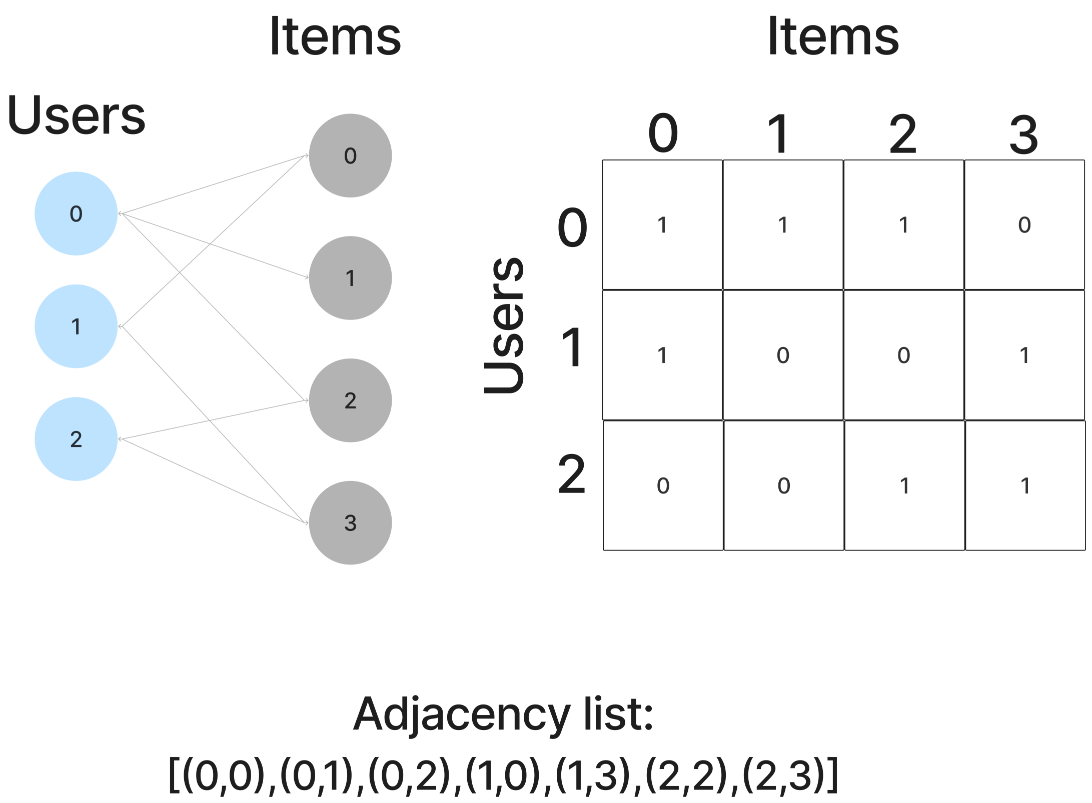


Figure 8: An example adjacency list representation, of a adjacency matrix

vides the names of artists, tracks, and albums, but not audio or input embedding representations. As the information that is provided in the LFM-1b is not particularly expressive for the artists, albums, or tracks, utilizing the limited features as input for our models is optimal. Instead, utilizing pre-established methodologies irrespective of node features, but respective of the graph structure would provide more expressive representations of the nodes.

metapath2vec

The metapath2vec method was published in 2017 to propose a solution to finding a resourceful way create node embedding representations in heterogeneous networks by utilizing the structure of the graph, instead of the explicit node features [42]. The published work offered a solution to the issue of heterogeneous graphs having irregular, or sparse collections of explicit features. The preserved node context in the final embedding representations was observed to improve downstream graph

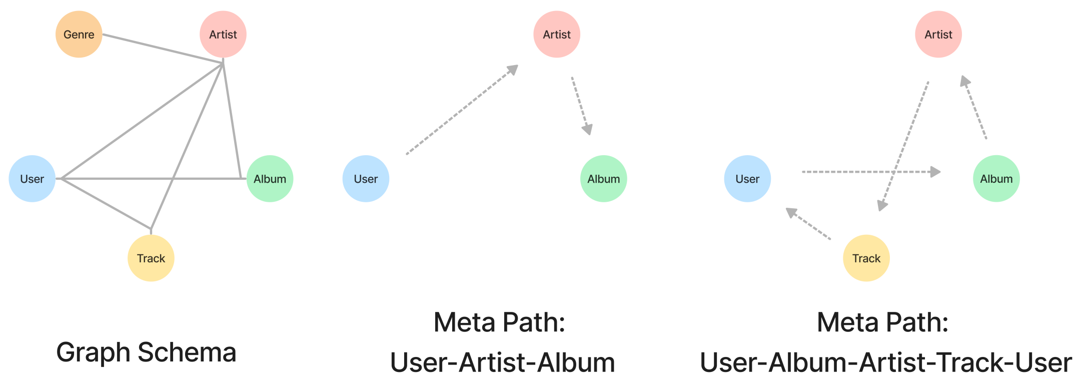


Figure 9: Examples of metpath2vec random walk meta-paths for the LFM-1b heterogeneous graph

learning tasks like node classification, link prediction, clustering, and graph classification. The metapath2vec method can capture semantic and structural relationships between different types of nodes by performing random walk operations over a specified heterogeneous "meta-paths". As a result, many different heterogeneous graph models utilize the final embedding representations of the nodes generated by the metapath2vec model as inputs into a deep learning model for different prediction tasks [42].

As a note on metapath2vec, user and item interactions should be emphasized more than other types of interactions in the graph. For our recommendation task, the objective is to provide a recommendation of undiscovered artists, albums, and tracks to users. Therefore, only user item meta-paths are selected for the random walk sequence, which has been shown to improve downstream node embedding representations [38] (see Figure 9). The metapath2vec algorithm is applied to the LFM-1b graph object to compute high-quality node embedding representations as inputs into the selected HGNN models discussed later. As the data loader should be flexible for future use, the hyper-parameters of the metapath2vec method are customizable inside the DGL LFM-1b data loading class provided in this thesis.

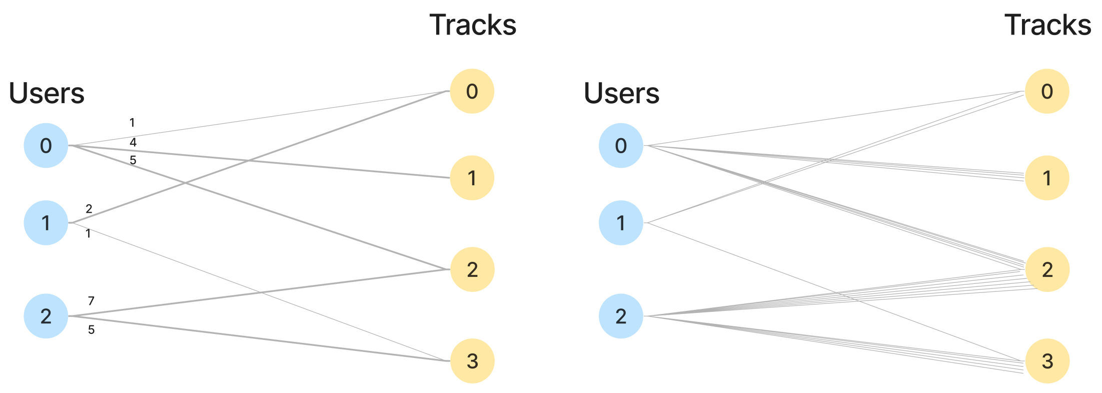


Figure 10: A weighted edge data representation (Left), a default edge data representation, where each edge is denoted with a timestamp (Right)

3.2.4 Compiling DGL Edge Data

As one of the main objectives of the thesis is to provide a reusable resource to the MIR community and enable researchers to use all the contextual information that is available in the LFM-1b data set. Incorporating edge data into the final representation of the DGL LFM-1b graph is needed. Specifically, there two different approaches that can be optionally enabled to add edge data to our graph representation (see Figure 10). By default, edge data in a graph can be represented as interactions between nodes in a graph (a user listening to a track, album, or artist is represented a single edge between the corresponding pair of nodes, each denoted with a timestamp). Alternatively, since users often listen to the same track, album, or artist more than once, the edges from a particular user can be denoted as one edge with a weighted value (a "listen count" value), corresponding to the number of interactions made with a particular track, album, or artist. Additionally, this value can be normalized as it is possible that some users will listen to specific artists, albums, or tracks in an unbalanced manner, more than other users. For both the different representations of the edge data, the affected edges in the LFM-1b data set are only interactions that exist between users and tracks, albums, or artists. Therefore, this implementation of the DGL LFM-1b data loader offers flexibility when producing a *HeteroGraph* object.

3.3 Models

The LFM-1b data set contains information on listening behaviors for its users and their interacted tracks, albums, or artists. Representing this information in a homogeneous graph cannot be done without sacrificing contextual and semantic information loss. Additionally, this limitation prevents the objectives of the thesis to be computed using a traditional Graph Convolutional Neural Network (GCN). As mentioned before, the original GCN model can only operate on homogeneous graphs. Therefore, to apply a neural network to a heterogeneous graph representation of the LFM-1b data set. A Heterogeneous Graph Neural Network (HGNN) will need to be utilized. HGNN RS leverage statistical relational learning inside the deep learning algorithms. This is particularly why GCNs were introduced in the state of the art. Specifically in this thesis, two algorithms will be utilized to perform link prediction on the LFM-1b data set.

3.3.1 Relational Graph Convolutional Network (RGCN)

Relational Graph Convolutional Networks is an extension of the GCN model that can traverse over a graph with multiple edge types. Notably, in the GCN paper, it can be observed that the weight matrix while being trainable, is shared amongst all the nodes in the graph. Specifically, for RGCN, there are unique trainable weight matrices for each edge type. This is an important factor during the propagation phase of the RGCN layer specified by the relation type [30]. RGCNs will have more parameters than GCNs due to the increase of relation weight matrices. This inevitably provides us with the need to regularize the rapidly increasing parameters with each new edge type that exists in the given graph. The RGCN paper provides two solutions to regularize the weights, Block Decomposition and Basis Decomposition. Both produce adequate regularization of the weights, with Basis Decomposition being the more commonly utilized method. More information can be found on the authors paper [30].

Notably, within this example, it was not expressed that the algorithm could traverse a graph of multiple node types as well as multiple edge types. In this thesis, the LFM-1b data set is represented as a heterogeneous graph, RGCN can still be applied by using a heterogeneous operation wrapper (a structure which enables graph-based machine learning models to be implemented on heterogeneous graphs). This is simply an operation that allows models to be able to handle multiple node types by applying multiple instances of the model. The OpenHGNN framework provides an implementation of the RGCN algorithm with such a heterogeneous wrapper. This implementation can be utilized later to validate other performance findings presented in this thesis.

3.3.2 Heterogeneous Graph Representation Learning with Relation Awareness (RHGNN)

Of the constantly improving HGNN models, there are very few which aim to corporate the factor of edge representations into the downstream node embedding representations. This concept was challenged by the researchers studying Heterogeneous Graph Representation Learning with Relation Awareness in 2021 [43].

The authors determined that it is substantially important to not just learn the embedding representations of edge relations, but also node representations concerning different relational interconnections. This builds upon the fundamental HGNN model by incorporating multiple RGCN models, utilizing a cross-relation operation to improve node representations, and proposing a fusing operation to aggregate relation-aware node representations into a single low dimensional embedding [43]. From the author's findings and their results of the RHGNN performance amongst other models, they observed a noteworthy performance increase over the standard baseline models including RGCN.

3.4 Training Graph Based Models

There exist some noteworthy differences in training graph-based models on a large heterogeneous graph provided by the DGL LFM-1b data loader. These general differences in training will be briefly outlined in the following sections, with additional task-specific information to be outlined after. Common limitations of these graph-based models reside in the number of computations required for training. This can severely limit the ability to train a graph-based model. For instance, as the LFM-1b graph for instance has millions of nodes and billions of edges a standard approach would load the full graph in memory, compute new embedding representations for every node, and for every layer in the model. Since this is so highly inefficient, mini-batch training is often utilized to efficiently increase the computation speed, and minimize the load placed on graph-based while training.

In training, a series of operations for every node in the given graph needs to be met. Therefore, for each target node, the neighbors of the node are used in the update operation as discussed. The objective is to fit the necessary computations for all the target nodes into a particular batch of training data. When utilizing mini-batch training however, mini batches still may suffer from large computation graphs (a target node, and the neighbors of the node) due individual nodes being highly interconnected. For this reason, graph-based models often employ a neighbor sampling function, such that a subset of the target node's computation graph neighbors is used to approximate a node's full computation graph. This allows the approximated computation graph to be bounded in size such that it can be utilized for large network training.

3.4.1 The Link Prediction Training Process

As mentioned earlier, it is also important to outline the unique operations required to train a model for the task of link prediction. This is an important factor in training, as link prediction is unlike most other graph-based tasks. Notice that randomly splitting edges of a full heterogeneous graph into training, validation, and tests sets may create unbalanced node or edge data. Therefore, ensuring a splitting mechanism to balance the node and edge types in a particular split of the graph is necessary. Additionally, for graph-based link prediction, there must be four splits made for training a model for link prediction: training, validation, test, and supervision edge sets (see Figure 11) [44].

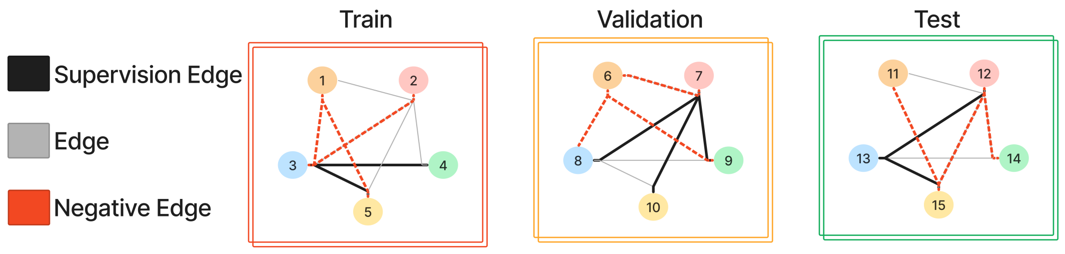


Figure 11: An example of splitting data into training, validation, test, and supervision edges for link prediction

For every training supervision edge in the provided heterogeneous graph, we need to use the training edges to predict the likelihood of the given supervision edge. To compute the predictions with the training edges, we also must identify negative edges in amidst the training edges, to perturb the supervision edges (the negative edges should not belong to the training or supervision edges) [44]. With the training, supervision, and negative edges, a predicted score can be assigned to the training supervision edge and the negative edges. Specifically, each score represents the likelihood that the given negative or training supervision edge existing for a particular source node.

Upon completion, we can formulate a loss function that maximizes the training supervision edge score (a true edge) and minimize the negative edge score (a false edge). With the necessary information to compute the training stage of the model, at validation time, the use of all the training, and training supervision edges are utilized to predict the validation supervision edges. Since the goal is to evaluate how each model can predict the existence of an edge of a specified type, we must compute the validation supervision edge and negative edge scores [44]. All the validation negative edges do not exist in the set of training or training supervision edges.

Upon completion, the testing phase begins where the training, training supervision, and validation supervision edges are used to predict test supervision edges against the negative test edges, not in the set of training or training supervision edges.

3.5 Evaluation

To validate the proposed methods chosen for this thesis, evaluations of the LFM-1b data loader and the LFM-1b link prediction models are presented with the chosen evaluation metrics.

3.5.1 Evaluation of the LFM-1b Data loader

The LFM-1b data loader implemented in the DGL framework provides many customizable features. The data loader also provides the capability of specifying subsets of the full LFM-1b heterogeneous graph. These complementary features are resourceful for researchers, as compiling the full LFM-1b heterogeneous graph in memory can take quite a long time to compute and require significant storage space. To evaluate the data loader and its flexible class structure, the values of the final node count and edge count, for all types will be measured for all possible subsets of the graph.

3.5.2 Evaluation of Link Prediction Models

Evaluating the link prediction models trained on the LFM-1b data set is not only important to validate the model, but rather it is also too important validate the significance of the LFM-1b data loaders' applicability to modern GNN research in the DGL framework. Since objective drives the primary focus of evaluating link prediction on the LFM-1b data set, for each user interaction edge type (user-to-album, user-to-track, and user-to-artist), each model's accuracy (ACC), precision (PR), loss, mean absolute error (MAE), and root mean squared error (RMSE) will be measured.

To additionally evaluate each link prediction model's capability to perform recommendations. For each link prediction model, the top-K most likely unconnected edges will be provided as recommendations for all users, such that recommendation evaluation metrics can be utilized to validate each model's performance. For the top-K recommendations, the different models can be evaluated for their recommendation accuracy (HIT@K), diversity (DIV@K), and the coverage (COV@K) of their recommendations.

$$HIT@K = 1/K * \sum(\hat{y}_k == y) \tag{3.1}$$

where $\hat{y}_k$ is the kth recommended item and y is the true items

$$DIV@K = 1/K * \sum(sim_(\hat{y}_k)) \tag{3.2}$$

where $\hat{y}_k$ is the kth recommended item and $sim_(\hat{y}_k)$ this the average similarity of all other recommendations

$$COV@K = I_s/I \tag{3.3}$$

where I_s is total amount of unique items recommended and I this total number of items in the given data

These measurements are specific to RS and provide insights that classical machine learning model evaluations would fall short at measuring. To be more precise, Equation 3.1 HIT@K measures the K (where k is a real integer) most likely songs, albums, or artists recommended to a particular user. The metric is equivalent to the percentage of correct songs, albums, or artists in the top K recommendations. This metric provides in depth insights in the accuracy of a model over a extended usage period. Equation 3.2 DIV@K is another percentage metric that measures the diversity of the top K recommendations by finding the average difference between each song, album, or artist in the top K recommendations. This metric enables researchers to understand how unique the recommendations are from each other. Lastly, Equation 3.3 COV@K is a percentage metric that measures the coverage of the top K recommendations by finding the percentage of possible songs, albums, and artists that can be recommended by the model evaluated. This allows the evaluators of a model to visually interpret how much content a model can or cannot be provided to a user.

Chapter 4

Results

The results will offer further technical insights into the findings from the approaches used to evaluate the objectives. Specifically, this section will provide the metrics calculated by using the DGL data loader class for the LFM-1b data set, performing link prediction, and recommendations.

4.1 LFM-1b Data Loader

Table 1: The number of nodes the full LFM1b data loader can capture

Node Type	**Number of Nodes**
User	120K
Artist	3M
Album	15M
Track	32M
Genre	21

The DGL LFM-1b data loader was created and utilized for the evaluation of the thesis objectives. The data loader, as mentioned, has a significant number of complexities that can be analyzed and customized for Music RS research. To achieve the thesis objectives, the data loader was tasked with compiling a full heterogeneous graph containing all combination variations of the set of node types in Table 1.

Computing the in-memory LFM-1b heterogeneous graph takes a long time. Due to the longevity of the computation time, saving the raw, pre-processed, and processed files for model training is particularly necessary. This, however, does limit the speed of the initial compile time, but prevents the later requirements of creating an entirely new heterogeneous graph, every time a slightly different variation of the graph is compiled.

Table 2: The number of edges the full LFM1b data loader can capture

Edge Type	**Number of Edges**
User -> Artist	61.4M
User -> Album	-
User -> Track	-
Artist -> Genre	400K
Album -> Artist	14.1M
Track -> Artist	27.2M

Due to the significant size of the LFM-1b data set, the full LFM-1b heterogeneous graph in the DGL did not fit into the available computer storage space provided for the analysis of the thesis. As it is apparent in Table 2, the count measurements for the number of edges between users' albums and tracks cannot be analyzed. As this is a particularly limiting result of the data loader such that adding the flexibility to create subsets of the full LFM-1b data set was necessary. While the limitation of storage space is a problematic issue for analyzing the rest of the thesis objectives. The flexibility of the subset features allows for the creation of data sets of a specified user size, that maintains the original demographic and listening behavior distributions of the original LFM-1b data set.

From the resulting complications with loading a full LFM-1b heterogeneous graph, for every combination variation of the node types, a generated subset of 25 users was utilized to predict the most likely edges to be connected for any given user instance. Only by keeping the user nodes below 50, as well as removing non-popular artists, albums, and tracks in the full LFM-1b heterogeneous graph, was the data loader able to successfully compile input data with the available computer storage space provided for the analysis of the thesis. Results from the subset features of the data loader are not displayed as the count measurements of node and edge types are inconsistent between different iterations of subsets.

4.2 LFM-1b Link Prediction

With the understanding of the provided limitations of the LFM-1b data loading class, and the knowledge of the methods utilized to alleviate and provide alternative inputs to the link prediction models, this section will address the discovered results of the prediction models used to identify the likelihood score of a user interacting with an undiscovered artist, album, or track.

Table 3: Evaluation of User to Artist LFM1b Link Prediction

Subset	**Model**	**RMSE**	**MAE**	**ACC**	**PRE**	**HIT@10**	**DIV@10**	**COV@10**
Al, Tr	RHGNN	0.29	0.15	0.97	0.97	0.97	0.38	0.18
Tr	RHGNN	0.59	0.44	0.86	0.81	0.68	0.22	0.13
Al	RHGNN	0.54	0.37	0.88	0.85	0.77	0.59	0.12
all	RHGNN	0.57	0.43	0.85	0.79	0.62	0.39	0.12

Within Table 3, the subset column indicates which node types were not included in the graph variation (all indicates that all node types were included), the model column indicates what model was used, the RMSE is the root mean squared error, the MAE is the mean absolute error, the ACC is the accuracy, the PRE is the precision of the model, HIT@K is the recommendation accuracy with k=10, DIV@K is the recommendation diversity with k=10, and COV@K is the recommendation coverage with k=10. As displayed, the ACC score decreased as more node types were included, and the RMSE score increased as more node types were included. Overall, the best test in terms of metric performance was the graph which included the fewest amount of node types and used RGHNN.

Differently displayed in Table 4, the ACC score however did not decrease as more node types were included, and the RMSE score did in fact decrease as more node types were included. Overall, the best test in terms of metric performance was the

Table 4: Evaluation of User to Album LFM1b Link Prediction

Subset	**Model**	**RMSE**	**MAE**	**ACC**	**PRE**	**HIT@10**	**DIV@10**	**COV@10**
Ar, Tr	RHGNN	0.29	0.44	0.95	0.97	0.96	0.35	0.13
Tr	RHGNN	0.11	0.01	0.98	0.99	1.00	0.05	0.14
all	RHGNN	0.11	0.01	0.99	0.99	1.00	0.03	0.14

graph which included the most amount of node types.

Table 5: Evaluation of User to Track LFM1b Link Prediction

Subset	**Model**	**RMSE**	**MAE**	**ACC**	**PRE**	**HIT@10**	**DIV@10**	**COV@10**
Ar, Al	RHGNN	0.23	0.09	0.95	0.96	1.00	0.02	0.11
Al	RHGNN	0.29	0.09	0.94	0.95	1.00	0.02	0.11
all	RHGNN	0.29	0.089	0.94	0.95	1.00	0.02	0.10

Lastly displayed in Table 5, the ACC score did not vary as more node types were included, and the RMSE score increased as more node types were included. Overall, the best test in terms of metric performance was the graph which included the least amount of node types.

Chapter 5

Conclusions

From the findings, some key observations can be made from each of the results sections. Specifically, the data loader provides optimal flexibility as well as transparency into the architecture of the class structure. The final compiled graph from the data loader is fully functional and can represent the entire LFM-1b data set for large-scale machine learning tasks. Additionally, once compiled the graph can run multiple post-processing operations. As seen, the data loader had a significant number of adjustments to account for. The promised information that can be acquired from the DGL LFM-1b data loading class can in fact be compiled. It in fact is merely not able to be utilized on the device the data loader was built with. Therefore, as mentioned, many of the subsets that were capable of being compiled could be fit into memory to be validated through model testing.

The link prediction models, and recommendation results were directly affected by this limiting result of the data loader. The limitation as mentioned was the reasoning for adding subset functionality to the data loading class. Utilizing subsets to compute link prediction models, was the only way possible to perform the necessary link prediction and single target recommendation. Not displayed in the results, an outsourced RGCN model, implemented through the OpenHGNN framework was used to validate the performance of the RHGNN model [38]. Both models shared overly optimistic, accurate, and precise capabilities in predicting undiscovered edge connections in the LFM-1b graph. The recommendation performance was also considerably overly optimistic as the HIT@K (recommendation accuracy) would even reach a perfect accuracy score.

It is challenging to conclude upon the direct results of this thesis for two distinct reasons. The first is that the models' performance whilst effective and notably optimistic, some significant over fitting factors need to be validated before supporting such a high-performing result. Secondly, the prediction tasks, whilst working, do not utilize graphs with a higher density of users, effectively reducing the training data of the model to a significantly small value (this can be visualized as the COV@K for these models rarely reached above .15). This could explain the high accuracy and HIT@K scores being so high, but it is challenging to compute results with higher user densities as the computation time to acquire these results rapidly expands with every new user.

5.1 Discussion

To briefly discuss the work that was presented throughout this thesis, it is first appropriate to address the purpose one last time. Music RS is a complex field of study. Due to the lack of explicitly rated interactions that users provide with their songs, albums, and artists, modeling implicit user behaviors in a graph allows the resulting structure to elicit the user preferences. As GNNs have become a state-of-the-art approach for modeling such complex recommendation scenarios. Music Recommendation has benefited significantly from the findings of the new and relevant GNN methodologies.

As described earlier, there is not much research conducted on the LFM-1b data set for music recommendation. With the limited findings of this thesis, it is quite challenging to conclude on the data sets usefulness as there were numerous preventative limitations. However, modeling the LFM-1b data set has brought myself to address that smaller data sets, with similar information may be more optimal for reproducible experiments amidst the scientific communities researching Music RS, utilizing GNNs in a Deep Graph Library or PyTorch Geometric framework.

Though this thesis successfully implemented complex GNN models and utilized them to perform numerous link prediction tasks within the provided graph representation of the LFM-1b data set. There was a significant number of limitations that prevented this thesis from decisively determining the performance changes of utilizing heterogeneous data as input graph for RS models. Being limited solely by allocation and memory requirements, the implementation, when given the necessary space can compute all results and achieve all the predefined objectives of this thesis.

All the necessary code can be found in the two repositories created for this thesis. The first is the repository for the LFM-1b Deep Graph Library customized in-memory data loader [1]. Secondly, the implementation of the data loader and HGNN models for link prediction [2].

[1]https://github.com/deancochran/DGL_LFM1b
[2]https://github.com/deancochran/upf-masters-thesis

List of Figures

List of Tables

Bibliography

[1] Jannach, D. & Zanker, M. Value and impact of recommender systems. *Recommender Systems Handbook* 519–546 (2022). URL https://link.springer.com/chapter/10.1007/978-1-0716-2197-4_14.

[2] Davidson, J. *et al.* The youtube video recommendation system (2010).

[3] Gomez-Uribe, C. A. & Hunt, N. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)* **6**, 1–19 (2015).

[4] Ying, R. *et al.* Graph convolutional neural networks for web-scale recommender systems 974–983 (2018).

[5] Lu, L. *et al.* Recommender systems. *Physics reports* **519**, 1–49 (2012).

[6] Milano, S., Taddeo, M. & Floridi, L. Recommender systems and their ethical challenges. *Ai & Society* **35**, 957–967 (2020).

[7] Jacobson, K., Murali, V., Newett, E., Whitman, B. & Yon, R. Music personalization at spotify 373–373 (2016).

[8] Zhang, S., Tay, Y., Yao, L., Sun, A. & Zhang, C. Deep learning for recommender systems. *Recommender Systems Handbook* 173–210 (2022). URL https://link.springer.com/chapter/10.1007/978-1-0716-2197-4_5.

[9] Gao, C., Wang, X., He, X. & Li, Y. Graph neural networks for recommender system 1623–1625 (2022). URL https://doi.org/10.1145/3488560.3501396.

[10] Derrow-Pinion, A. *et al.* Eta prediction with graph neural networks in google maps 3767–3776 (2021).

[11] Li, B. *et al.* Automated inference of molecular mechanisms of disease from amino acid substitutions. *Bioinformatics* **25**, 2744–2750 (2009).

[12] Li, Z., Xu, Q., Jiang, Y., Cao, X. & Huang, Q. Quaternion-based knowledge graph network for recommendation 880–888 (2020).

[13] Fan, Y., Zhang, L. & Wang, P. Relation-enhanced multi-graph attention network for recommendation 11–17 (2020).

[14] He, G., Li, J., Zhao, W. X., Liu, P. & Wen, J.-R. Mining implicit entity preference from user-item interaction data for knowledge graph completion via adversarial learning 740–751 (2020).

[15] Schedl, M., Knees, P., McFee, B. & Bogdanov, D. Music recommendation systems: Techniques, use cases, and challenges. *Recommender Systems Handbook* 927–971 (2022). URL `https://link.springer.com/chapter/10.1007/978-1-0716-2197-4_24`.

[16] Bertin-Mahieux, T., Ellis, D. P., Whitman, B. & Lamere, P. The million song dataset (2011).

[17] Bittner, R. M. *et al.* mirdata: Software for reproducible usage of datasets (2019).

[18] Lv, Q. *et al.* Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks 1150–1160 (2021). URL `https://doi.org/10.1145/3447548.3467350`.

[19] Fu, X., Zhang, J., Meng, Z. & King, I. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding (2020). URL `https://doi.org/10.1145/3366423.3380297`.

[20] Pang, Y. *et al.* Heterogeneous global graph neural networks for personalized session-based recommendation. *WSDM 2022 - Proceedings of the 15th ACM International Conference on Web Search and Data Mining* 775–783 (2022). URL `https://doi.org/10.1145/3488560.3498505`.

[21] Schedl, M. The lfm-1b dataset for music retrieval and recommendation. *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval* 103–110 (2016). URL `https://doi.org/10.1145/2911996.2912004`.

[22] Gao, C. *et al.* Graph neural networks for recommender systems: Challenges, methods, and directions (2021). URL `http://arxiv.org/abs/2109.12843`.

[23] Aguiar, L. Let the music play? free streaming and its effects on digital music consumption. *Information Economics and Policy* **41**, 1–14 (2017). URL `https://www.sciencedirect.com/science/article/pii/S016762451630110X`.

[24] Riegler, M. A., Wang, J., Su, L. & Schedl, M. Deep learning in music recommendation systems. *Frontiers in Applied Mathematics and Statistics | www.frontiersin.org* **1**, 44 (2019). URL `https://dblp.uni-trier.de`.

[25] Fang, H., Zhang, D., Shu, Y. & Guo, G. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Trans. Inf. Syst.* **39** (2020). URL `https://doi.org/10.1145/3426723`.

[26] Ren, P. *et al.* Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence* **33**, 4806–4813 (2019). URL `https://ojs.aaai.org/index.php/AAAI/article/view/4408`.

[27] Jeong, D., Kwon, T., Kim, Y. & Nam, J. Graph neural network for music score data and modeling expressive piano performance 3060–3070 (2019).

[28] Korzeniowski, F., Oramas, S. & Gouyon, F. Artist similarity with graph neural networks. *arXiv preprint arXiv:2107.14541* (2021).

[29] Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks .

[30] Thanapalasingam, T., Berkel, L. V., Bloem, P. & Groth, P. Relational graph convolutional networks: A closer look URL `https://github.com/thiviyanT/torch-rgcn`.

[31] Raimond, Y., Abdallah, S. A., Sandler, M. B. & Giasson, F. The music ontology. **2007**, 8th (2007).

[32] Salha-Galvan, G. Contributions to representation learning with graph autoencoders and applications to music recommendation (2022). URL `https://arxiv.org/abs/2205.14651`.

[33] Hamilton, W. L. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* **14**, 1–159 (2020).

[34] Kipf, T. N. Deep learning with graph-structured representations (2020).

[35] Wang, X., He, X., Cao, Y., Liu, M. & Chua, T.-S. Kgat: Knowledge graph attention network for recommendation 950–958 (2019).

[36] Koller, D. *et al.* Introduction to statistical relational learning (2007).

[37] Sanchez-Lengeling, B., Reif, E., Pearce, A. & Wiltschko, A. B. A gentle introduction to graph neural networks. *Distill* **6**, e33 (2021).

[38] Chua Shi, P. S. Y., Xiao Wang. Heterogeneous graph representation learning and applications (2022). URL `http://gen.lib.rus.ec/book/index.php?md5=39DC96A5068CB2837EEEC234FA6DEC0B`.

[39] Fey, M. & Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428* (2019).

[40] Wang, M. Y. Deep graph library: Towards efficient and scalable deep learning on graphs (2019).

[41] Hagberg, A., Swart, P. & S Chult, D. Exploring network structure, dynamics, and function using networkx (2008).

[42] Dong, Y., Chawla, N. V. & Swami, A. metapath2vec: Scalable representation learning for heterogeneous networks 135–144 (2017).

[43] Yu, L. *et al.* Heterogeneous graph representation learning with relation awareness (2021). URL `http://arxiv.org/abs/2105.11122http://dx.doi.org/10.1109/TKDE.2022.3160208`.

[44] Leskovec, J. Cs224w: Machine learning with graphs (2021). URL `http://web.stanford.edu/class/cs224w/`.