# HETEROGENEOUS GRAPH-BASED NEURAL NETWORK FOR SOCIAL RECOMMENDATIONS WITH BALANCED RANDOM WALK INITIALIZATION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Amirreza Salamat

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

December 2020

Purdue University

Indianapolis, Indiana

# THE PURDUE UNIVERSITY GRADUATE SCHOOL
# STATEMENT OF THESIS APPROVAL

Dr. Ali Jafari, Co-Chair

Department of Computer and Information Technology

Dr. Brian King, Co-Chair

Department of Electrical and Computer Engineering

Dr. Xiao Luo

Department of Computer and Information Technology

**Approved by:**

Dr. Brian King

Head of the Graduate Program

Dedicated to my parents who are my main source of motivation.

# ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my thesis commitee, Dr. Jafari, Dr. Luo and Dr. King for helping me immensely. I would also like to thank Sherrie Tucker for keeping me on track since the beginning of the program. Last but not least, I thank all the members of the CyberLab for creating such a wonderful atmosphere for me to research and learn in.

TABLE OF CONTENTS

Page

# LIST OF TABLES

# LIST OF FIGURES

Figure Page

# SYMBOLS

| | |
|---|---|
| $A$ | graph adjacency matrix |
| $C$ | size of the context window |
| $d$ | embedding dimensions |
| $E$ | graph edges |
| $f$ | node occurence frequencies |
| $G$ | graph |
| $h$ | node latent features |
| $i$ | item |
| $l$ | walk length |
| $V$ | graph vertices |
| $u$ | user |

# ABBREVIATIONS

| | |
|---|---|
| AI | Artificial Intelligence |
| CN | CourseNetworking |
| CNN | Convolutional Neural Network |
| DevOps | Development and IT Operations |
| GAT | Graph Attention Network |
| GCN | Graph Convolutional Network |
| GNN | Graph Neural Network |
| ML | Machine Learning |
| NN | Neural Network |
| NCE | Noise Contrastive Estimate |
| NLP | Natural Language Processing |
| RNN | Recurring Neural Network |
| SoRec | Social Recommender |

# ABSTRACT

Salamat, Amirreza. MSECE, Purdue University, December 2020. Heterogeneous Graph-Based Neural Network for Social Recommendations with Balanced Random Walk Initialization. Major Professor: Ali Jafari and Brian King.

Research on social networks and understanding the interactions of the users can be modeled as a task of graph mining, such as predicting nodes and edges in networks. Dealing with such unstructured data in large social networks has been a challenge for researchers in several years. Neural Networks have recently proven very successful in performing predictions on number of speech, image, and text data and have become the de facto method when dealing with such data in a large volume. Graph Neural Networks, however, have only recently become mature enough to be used in real large-scale graph prediction tasks, and require proper structure and data modeling to be viable and successful. In this research, we provide a new modeling of the social network which captures the attributes of the nodes from various dimensions. We also introduce the Neural Network architecture that is required for optimally utilizing the new data structure. Finally, in order to provide a hot-start for our model, we initialize the weights of the neural network using a pre-trained graph embedding method. We have also developed a new graph embedding algorithm. We will first explain how previous graph embedding methods are not optimal for all types of graphs, and then provide a solution on how to combat those limitations and come up with a new graph embedding method.

# 1. INTRODUCTION

## 1.1 Graph Recommender

Social recommendations have been the topic of study for several years and are crucial in filtering vast amounts of information for every person into small digestible pieces. Research on social networks and understanding the interactions of the users can be modeled as a task of graph mining, such as predicting nodes and edges in networks. Learning the user's preferences from all the possible sources of information is one of the main challenges when building social recommenders. GNNs (Graph Neural Networks) have been gaining momentum in recent years and have been successful when dealing with large-scale graphs, and they can be applied to social networks with some modifications. In this research, we propose the HeteroGraphRec, which provides social recommendations by modeling the social network as a heterogeneous graph and utilizing GNNs and attention mechanisms to intelligently aggregate information from all sources when building the connections between user to user, item to item, and user to item. HeteroGraphRec is capable of gathering information about the user's connections (friendships, trust network), item interaction history, and item similarities to attain rich information about the preferences. To evaluate the HeteroGraphRec, we use three real-world benchmark datasets and demonstrate that the proposed HeteroGraphRec achieves superior performance compared to ten other state-of-the-art social recommender systems. We extensively analyze the HeteroGraphRec model to illustrate the effectiveness by changing the embedding dimensions of the users and items and show the interpretability of our model by examining each component of the model's contribution. The model analysis shows that HeteroGraphRec is robust and can consistently perform better than the other systems.

Social recommenders need to consider user interactions and item preferences to provide accurate recommendations. Most of the typical social recommender systems consider the interactions between users and items through various previous activities; these interaction histories are the first clues to understanding user's preferences. The interaction between a user and other users creates a social network that gives insight into each user's preference because a user's friends or connections generally influence their choices. Moreover, the preference of other users who like the same items is another valuable source of information and shows the collective preference of the users.

The similarity between items are often considered in many collaborative filtering based algorithms. However, they are generally defined within the vector space, and are inefficient when working with large graph data. In other words, the connections are represented as vectors, which are as natural and dynamic as the graph representation is.

On the other hand, for an accurate and robust recommender system, aggregating information from various sources to gain a full picture of the user preferences is crucial. Hence, there are several challenges when dealing with social data with graph representations.

The main challenge is that graphs are, by nature, unstructured. So that most machine learning algorithms cannot be applied to graphs, or they are highly inefficient. The scalability of the algorithms working with graph representation is the other key concern when dealing with social networks. It is not out of the ordinary to have hundreds of thousands of nodes and millions of edges in a graph. When providing social recommendations, there are two or more different entities (users, items and etc.) with various features and different types of interactions.

Graph Neural Networks (GNNs) [1] [2] [3] [4] have proven effective when dealing with graph data and have shown success when applied to tasks like node classification and social predictions. Recent studies [4] [5] have introduced the attention mechanisms to the neural network, which makes it possible to attend to relevant

neighborhood information instead of all the neighbors simultaneously, which is invaluable for social recommenders. Yet, there is still room to improve the framework of social modeling and utilize the GNNs structure to use the strength of these neural networks in social models.

In this paper, we propose a versatile social recommender system based on GNNs, which captures various aspects of a social network and makes recommendations and predictions based on this knowledge. This method takes advantage of the recent improvements in GNN architecture. It modifies the neural network architecture and social modeling to provide accurate recommendations and overcome the aforementioned social recommenders. Different from the existing research in the literature, we develop a new model of a social network where items are connected in a way similar to how users are connected. Furthermore, we provided a framework that takes advantage of the new model to create a more accurate, more versatile social recommender.

We believe the items in the social network are structured signals and have similarities between each other. These similarities can be explicitly derived (e.g., through item categories or the likeness of the item content) or inferred (e.g., from user interactions). Sequentially, these items could be interconnected based on structural information. These connections represent the structure of the items. By feeding the structure of items alongside other inputs to the neural network, the model can have a better perception of the network as a whole, ergo, receiving a boost to recommendation accuracy.

The other advantage of the proposed HeteroGraphRec model is that the items are treated as nodes rather than the features for users. Thus, having general node information, the aggregation method can find latent features for both users and items in a similar manner. Prediction tasks such as rating prediction can use node aggregators for the target user and item to make predictions, while friend recommendations can use the same node aggregator architecture for target users. Item bundling and similarity prediction can again use the same node aggregator but for two target items

instead. Therefore modeling items as nodes and having a general node aggregator can make the social recommender more flexible and versatile.

We evaluate our results on three real-world data sets against ten other state-of-the-art recommender systems. Our model consistently outperforms the compared ones based on the returned Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) values. The results also show that our model can extract deep connections in the network to make relevant recommendations. We implement extensive model analysis to demonstrate the effectiveness of our model from different prospects, such as embedding dimensions of users and items, the components of the framework, and so on. Finally, we compared the performance of the model in making top-K predictions and demonstrated how initialization could help with the convergence speed of the model. The results show that our model, HeteroGraphRec, is robust and can consistently outperform the state-of-the-art recommender systems.

Our contribution in this research can be summarized in two steps:

- We provide a model for social network data that models the network as a heterogeneous graph with the differentiation that items in this network are treated as structured data, and are interconnected based on their structure and similarities.

- We design a recommender system that utilizes the social model mentioned above to obtain information from more sources for each node, and attempts to augment its data by collecting information from similar items. This results in a more accurate, more robust recommendations.

- We demonstrate that the item connections provide consistency in modeling social networks, as the items are treated as a node rather than a feature for users, which, as mentioned before, can make the recommender system more versatile.

- We thoroughly evaluated our model against the most recent recommendation models using three different real-world data sets. The results show that our approach consistently outperforms all previous baselines on all three datasets.

## 1.2 Balanced Random Walks

Additional to the new Neural Network Architecture, we also introduce another initialization method which improves the training accuracy and the performance of the model by initializing the network with pre-trained network representations to provide a hot-start to the network. The challenges of building the graph representation include engineering features used by learning algorithms. Recent research in representation learning can automate the prediction by learning the features themselves. Many types of research have performed graph sampling using random walks or its derivatives. However, the random walk sometimes cannot represent the features of the graph accurately enough. In this research, we propose BalNode2Vec – a new sampling algorithm for learning feature representations for nodes in networks by using balanced random walks. We define a notion of a node's network neighborhood and design a balanced random walk procedure, which adapts to the graph topology. We show that through exploring the graph through a balanced random walk can generate richer representations. Efficacy of BalNode2vec over existing state-of-the-art techniques on link prediction is demonstrated by using several real-world networks from different domains.

The most recent research in graph networks investigated approaches to generate graph embedding, which learns a mapping from a network to a vector space while preserving relevant network properties. DeepWalk [6], LINE [7], and node2vec [8] are the graph embedding algorithms used in various domains, such as assessing protein interactions or predictinggenome interactions. Graph embedding not only improves the efficiency and accuracy of user profiling but also enables vector analysis techniques to be applied. So that data can be transformed with less computational costs. The

applications built upon the graph network, such as the recommender system can be more scalable and also be used in real-time applications.

The Node2Vec is built based on the idea of word embedding generation techniques, such as the Skip Gram model [9]. The basic theory is that performing short random walks on a scale-free graph can generate node frequencies that closely follows Zipf's law [6], which is similar to the word distribution of a corpus. Although the random walk provides a means for sampling the graph, it may not represent the features of the graph accurately, especially with a directed graph. The first issue is that the initial starting node is chosen at random, meaning that the nodes with lower degrees are treated equally with the higher degree nodes. As the length of the random walk increases, the walks bias towards nodes with more in-degrees and moves away from the nodes with fewer in-degrees, which causes those nodes to be underrepresented while the nodes with more in-degrees are over-represented. This behavior is illustrated by Figure 1.1, which shows that node $A$ might not be walked on very often, and node $E$ appears on several walks. While the latter issue emerges in directed graphs, the former appears in both directed and undirected graphs.

On the other hand, the notion of using random walks for sampling a graph was that in a scale-free network, the short random walks follow the power-law similar to the degree distribution [6]. Therefore, word embedding techniques like Skip Gram could be applied to the graph to create embeddings. However, with this type of approach, there are two issues:

- The sampling error increases as the size of the graph increases.
- The node distribution diverges from the power-law as the length of the walk increases. Hence, more in-depth, more abstract features cannot be extracted from the network.

In this research, our objective is to develop a novel algorithm to prevent the sampling issues in calculating the probabilities of walking each node when using a random walk and comparing them with the actual degree distribution (which follows

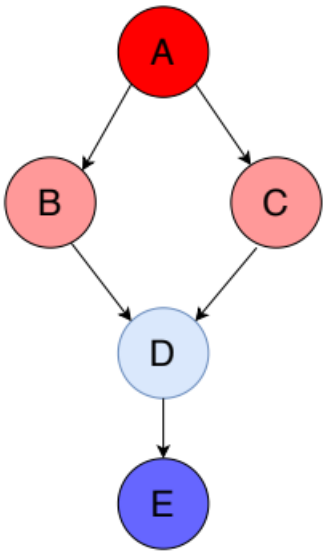


Fig. 1.1. A sample directed graph with nodes with various in-degrees to demonstrate that random walks on this graph tends to select bias towards the nodes with more direct or propagated in-degrees.

the power-law). Therefore, the sequences are more suitable for being used in an algorithm like the Skip Gram algorithm. Specifically, we propose a balanced random walk sampling algorithm – BalNode2Vec to generate the graph embedding. The algorithm is designed to walk on the nodes that are underrepresented using the random walk and put less emphasis on the nodes that are walked often. This new approach balances the node distribution and adjusts suit the Skip Gram optimizer. This proposed BalNode2Vec is able to produce relevant representations for large graphs without biasing on the nodes with more in-degrees and efficient for running and updating without advanced dedicated hardware, scalable to vast networks of hundreds of thousands of edges, adaptable to the variance of the data over time. We evaluate the BalNode2Vec on five different data sets. The comparison against the state-of-the-art algorithms demonstrates that BalNode2Vec gains the highest AUC scores on all data sets.

# 2. RELATED WORK

Social recommendations has been the subject of several studies in recent years in recent years [10] [11] [12]. These methods utilize the fact that the user's preference is affected by the neighboring or connected users, which can be confirmed from social correlation theories [13] [14] [15]. Based on these social correlation theories, SoRec [16] proposed a co-factorization method, which shares a common latent user-feature matrix factorized by ratings and by social relations. TrustMF [17] modeled mutual influence between users and mapped users into two low-dimensional spaces: truster space and trustee space, by factorizing social trust networks. SoDimRec [10] first adopted a community detection algorithm to partition users into several clusters and then exploited the heterogeneity of social relations and weak dependency connections for a recommendation.

Neural Networks achieved relative success in modeling vibrant and abstract data in large quantities. Deep Neural Networks were already being used in fields like speech processing, Computer Vision, and Natural Language Processing. In contrast, they were not initially effective at modeling graphs [18]. They went through several iterations before achieving relative success [19]. Early attempts in using neural networks in arbitrarily structured data used recursive neural networks to process data represented in graph domains as directed acyclic graphs [20] [21]. GNNs were generalized in [2] [1] and they could directly deal with a more general class of graphs, e.g., cyclic, directed and undirected graphs. This network updates node states by exchanging neighborhood information recurrently until a stable equilibrium is reached. Then, a neural network would produce an output for these predictions.

SMRMNRL [22] developed a social-aware movie recommendation in social media from the viewpoint of learning a multimodal heterogeneous network representation for ranking. They exploited the recurrent neural network and convolutional neural

network to learn the representation of movies' textual description and poster image and adopted a random-walk based learning method into multimodal neural networks. In all these works [23] [22], they addressed the task of cross-domain social recommendations for ranking metric, which is different from traditional social recommender systems.

The most relevant work with neural networks includes DLMF [24] and DeepSoR [25]. DLMF [24] used auto-encoder on ratings to learn representation for initializing an existing matrix factorization. A two-phase trust-aware recommendation process was proposed to utilize deep neural networks in matrix factorization's initialization. The system synthesizes the user's interests and their trust friends' interests together with the impact of community effect based on matrix factorization for recommendations. DeepSoR [25] integrated neural networks for user's social relations into probabilistic matrix factorization. They first represented users using pre-trained node embedding technique, and further exploited k-nearest neighbors to bridge user embedding features and neural network.

Graph embedding algorithms can embed the structural information of nodes into vectors, which can then be added to the existing user features to generate recommendations [26] [27] [28] [29]. For instance, Deepwalk [6] and Node2Vec [8] embed nodes into vectors based on the probability of node co-occurrences in a random walk.

DeepWalk [6] introduced deep learning (unsupervised feature learning) techniques that learn representations of graph vertices and by modeling a stream of random walks on the graph and feeding the generated strings into Skip Gram. DeepWalk achieved scalability and adaptability in large networks but was not configurable to the pattern that we wanted to capture from the graph.

LINE [7] is another embedding generation algorithm that embeds the graph into $d$ dimensional vectors and learns $d/2$ embeddings using Breadth-First Search (BFS) and another $d/2$ using Depth First Search(DFS), this method loses several benefits of random walk which is explained in [8] and is not entirely configurable to different patterns of data.

In the paper of Node2Vec [8], it describes that the prediction task in graphs is comprised of two different aspects of the network, homophily and structural equivalence. Homophily hypothesis formulates the nodes which belong to the same community and have frequent interactions must be embedded together [30], while structural equivalence hypothesis [31] states that nodes with similar structural role should be embedded together. Node2vec [8] followed the same procedure as Deepwalk except that it used second-order random walks and introduced two new parameters, return parameter and in-out parameter, and claimed that these two parameters simulate the effects of BFS and DFS which represent structural equivalence and homophily respectively. This algorithm allowed the random walks to be configured based on the requirements, but it distorts the node frequencies that Negative Sampling needs to consider. This issue is elevated when dealing with graphs that have lower average edge per node. Also, in the same study, several link prediction algorithms were proposed to evaluate the performance of the model, these link prediction algorithms were compared in [32] [33] [34] in terms of performance, the major shortcoming in all of these techniques has high computational efficiency.

The metapath2vec and metapath2vec++ were introduced in [35], which incorporated metapaths to model heterogeneous networks better, but the drawback of these methods is that domain knowledge is required to define these metapaths which may not be a viable option in many scenarios. Walklet [36] was another approach that captured multiscale node representation on graphs by sampling edges from higher powers of the adjacency matrix and as a consequence, skipping some nodes. The authors claimed that sampling from each order of the adjacency matrix captures a specific dimension of social interactions, but some deeper connections can only be extracted from a blend of different orders of the adjacency matrix. Additionally, the training was carried out by a MultiLayer Perceptron, which has performance issues on larger output classes.

PinSage [37] used this concept of random walk for recommendations in webscale graphs. sRMGCNN [38] adopted GNNs to extract graph embeddings for users and

items, and used the RNN for the diffusion process. While these methods are capable of adding the structural information into node features, one issue is that they consider all the neighboring nodes, but ignore the fact that not all neighboring nodes have equal influence on the user. Detecting the important connections out of all the neighbors is crucial, especially in some prediction tasks. Resolving this issue requires deeper, more complex algorithms to comprehend all the neighborhood information fully.

Graph Convolutional Network (GCN) addresses the cyclic mutual dependencies architecturally using a fixed number of layers with different weights in each layer. The convolution operation is more convenient and efficient when dealing with neighborhood information. This method aggregates all the neighboring information of a node using an aggregate function to derive the latent features of a particular node. The most common aggregator function for GCN's is the mean function which means all the neighboring nodes are considered with equal weights. One advantage of GCN's is that they are transductive, meaning that even if a new node arrives in the network, the network can obtain information about it by comparing it to another node with the same connection structure.

GCMC [39] proposed a graph auto-encoder framework, which produced latent features of users and items using these convolutional layers. The concept of borrowing features from adjacent nodes is also known as message passing. GCN considers equal weights for each neighboring node when passing features. As the number of neighbors of a node can vary from one to thousands, it is inefficient to take the full size of a node's neighborhood. GraphSage [40] adopts sampling to obtain a fixed number of neighbors for each node. Graph Attention Network (GAT) adopts the attention mechanism to assign a weight to each neighbor based on how relevant they are, so that more important nodes receive larger weights. ScAN [41] employed the co-attention mechanism to find the important friends of each user, which influence their decision the most, and used this information to improve the recommendations. The adoption of attention mechanisms over the convolution allowed the model to learn which interactions to attend. Thus, it can provide more fine-grained control of information when

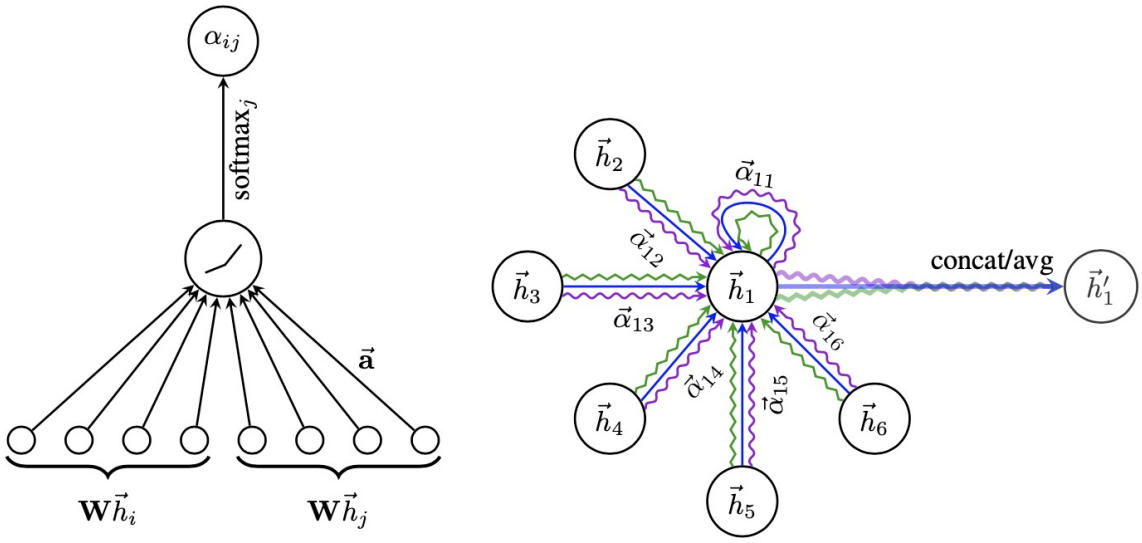


Fig. 2.1. Left: The attention mechanism $a\left(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j\right)$ employed by Graph Attention Network, parametrized by a weight vector $\vec{\mathbf{a}} \in R^{2F'}$, applying a LeakyReLU activation. Right: An illustration of multihead attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

training. As a result, it provided more accurate recommendations. GraphRec [42] also uses the attention mechanism similar to ScAN. It uses the attention network to gather information about the users who use the same item in addition to what ScAN does. One limitation of this method is that it does not directly consider the structure of the items, and the similarities between items to supplement its information of the item.

Unlike the previous research, our model considers the items as structured data and models the network as a heterogeneous graph. It then utilizes the recent neural network architectures and attention mechanism to find important information from all the sources intelligently. The concept of using the data structure as one of the inputs has already been used in Neural Structured learning [43] [44] [45] [46] in fields

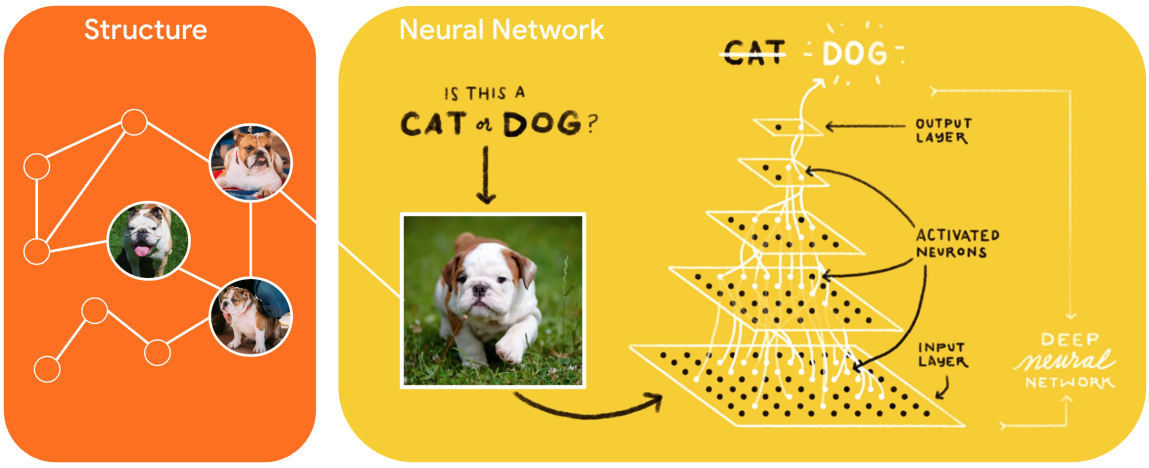


Fig. 2.2. Neural Structured Learning (NSL) is a new learning paradigm to train neural networks by leveraging structured signals in addition to feature inputs. Structure can be explicit as represented by a graph or implicit as induced by adversarial perturbation.

of image processing [47] [48] as show in 2.2 and shown improvements in prediction and classification tasks. In the following sections, we elaborate on how items can be represented as node structure, and the additional item to item connections can improve the model's accuracy.

# 3. COURSENETWORKING

CourseNetworking(CN) [49] is an educational website which is constituted by three main aspects: LMS, Eportfolio and Social Network.

The LMS aspect of CN consists of courses which can be issued by the university, or personally made for custom groups. Courses allow for posts, polls, events and contents that are made available per instructor's request.

The ePortfolio aspect of CN allows for a user to store and showcase various types of their learning evidence, such as their skills 3.1 and expertise 3.2. From the data modeling standpoint, there are two types of information that a user can store, personal and collaborative. Personal information (like education, skills, expertise) are related only to the specific user, while recommendations, endorsements and connections 3.3 are about the interactions between two users.

The social network aspect of the CN also allows features posts and polls and events between large number of people that share the same interest. User's can join different communities based on their preferences and attempt to collaborate and share information between themselves.

The main challenge in modeling the data in such graphs is the unstructured nature of them. Most of the users do not fill all the information in all fields, and the connections between users is not identical as well. For instance, one user may have more than 100 connections while another user may have no connections. This unique, unstructured data is best modeled as a graph in which the nodes are the users and the edges are their connections. These nodes and edges can have various features based on their profile.

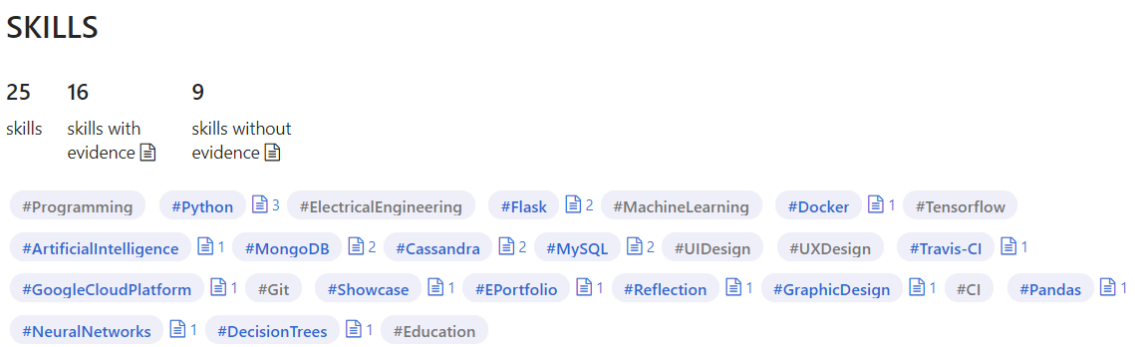


Fig. 3.1. The skills section of a user's ePortfolio in CourseNetworking

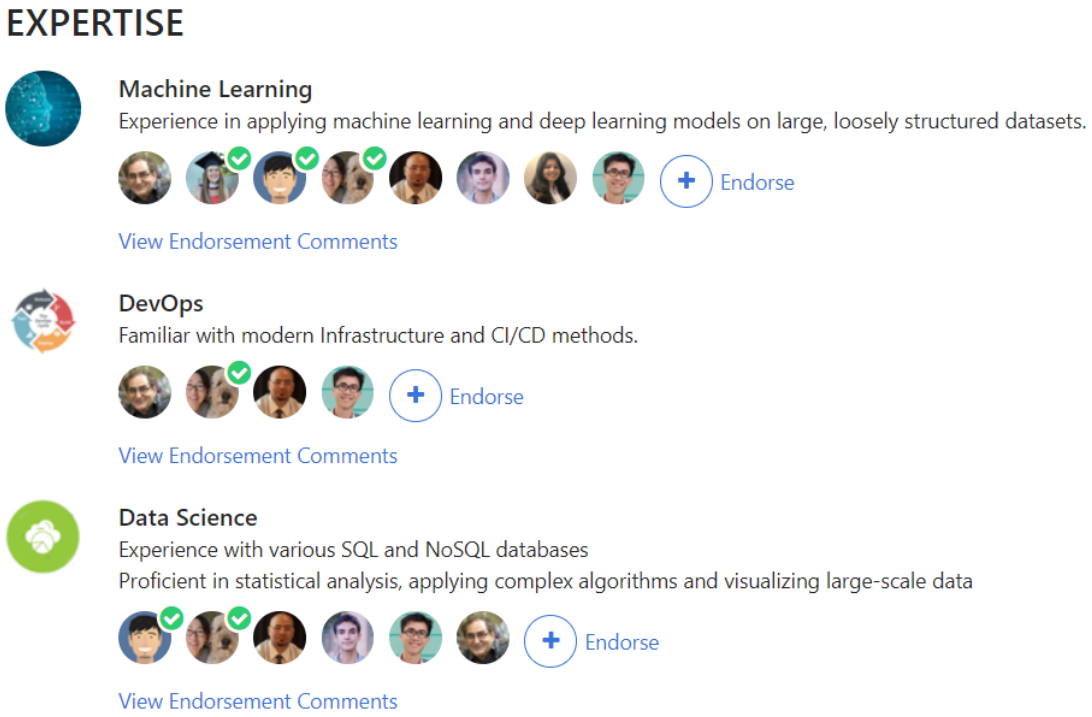


Fig. 3.2. The skills section of a user's ePortfolio in CourseNetworking with other user's endorsements

## 3.1 Rumi

Learning Management Systems (LMS's) are essential to online instructors. With the existence of these systems, online teachers can share their courses and interact

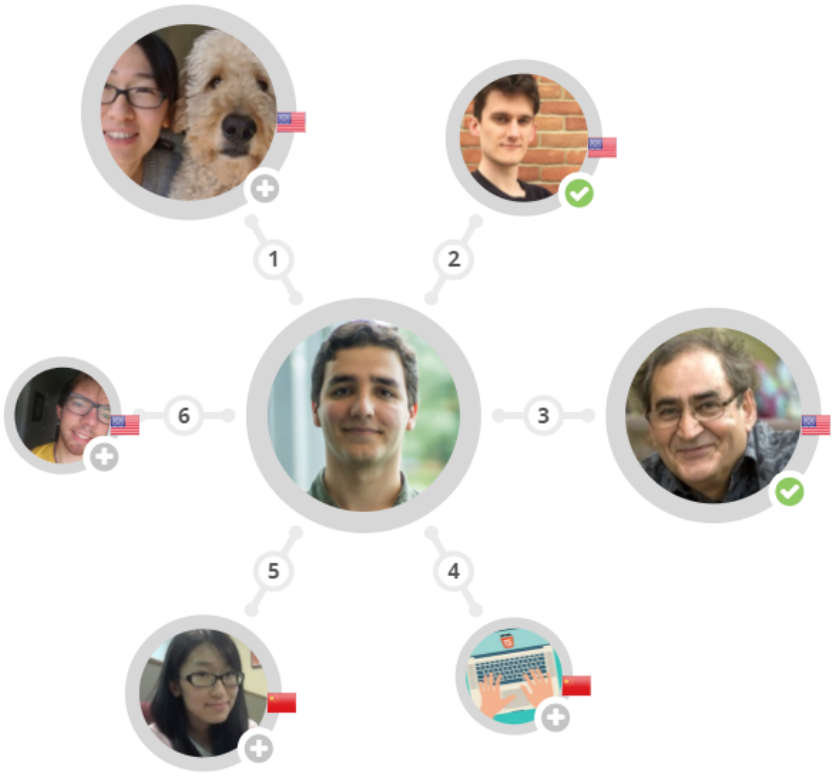


Fig. 3.3. User's interactions with other users in CourseNetworking.

with students around the world. There are plenty of LMS's on the market such as Canvas, Blackboard, etc. which are static and not tailored to the individual user's personal needs. The use of an Intelligent Agent can make these learning environments more dynamic and personalized by guiding students to increase their productivity [51].

Rumi [50] is the codename of the intelligent agent developed at the IUPUI Cyber-Lab in collaboration with CourseNetworking LLC (theCN.com). In this presentation, graduate students at IUPUI will discuss and demonstrate the research and development of Rumi using Artificial Intelligence (AI) as a resource provider, networking agent, and caring mentor to incentivize students to use the LMS .

As an intelligent agent, Rumi is designed to be an expert on users and their preferences. It makes recommendations that are useful to the users and are tailored

to their personal needs using AI techniques such as Machine Learning (ML), pattern recognition, and data mining.

Rumi is expected to play the role of a digital mentor, personal teaching assistant, career advisor, caring classmate, and an entertaining buddy. The agent will dynamically and intelligently recommend ePortfolio customization opportunities, networking activities, career plans, new learning credentials, and even job recommendations to users.

Rumi is designed to function within both an ePortfolio and LMS environment. Current ePortfolio and LMS systems are built to be static lacking the interactivity and personalization users require. Adoption of an intelligent agent can substantially improve the stickiness and effectiveness of these legacy learning environments.

### 3.2 Rumi Challenges

Like all other recommender systems, Rumi needs data from interactions to improve his recommendation accuracy, this problem is known as "Cold Start" in recommender systems. One of the ways we can prevent cold start in such systems in gamification. By creating competition and rewarding winners, the users are motivated to log in more often and through logging in, they interact more with Rumi and then Rumi can gather more data about the users. This boost in data and interaction also helps Rumi perform the other functionalities better too since he has more knowledge about the user and the user becomes more trusting to Rumi and his capabilities, thus, a circle of trust is formed between the user and the agent.

### 3.3 Quotes Competition

Rumi Quote Competition is one of the services that Rumi offers which promotes gamification. In this competition, every user's tagline 3.4 is gathered, and then Rumi shows these quotes to the users in a pair of two, and asks them which of the two they like more. Rumi then collects result of the user's choices and ranks the quotes based

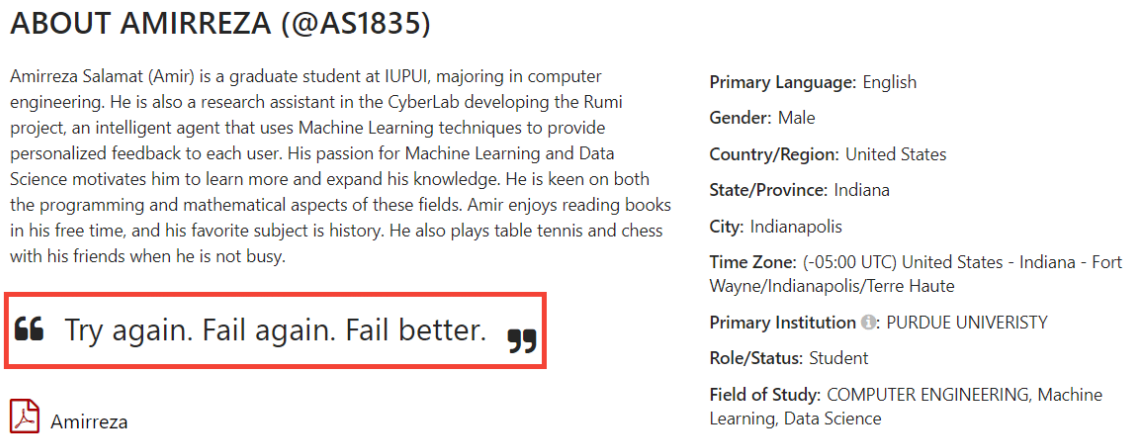


Fig. 3.4. CN bio section which contains the user's tagline (highlighted as red)

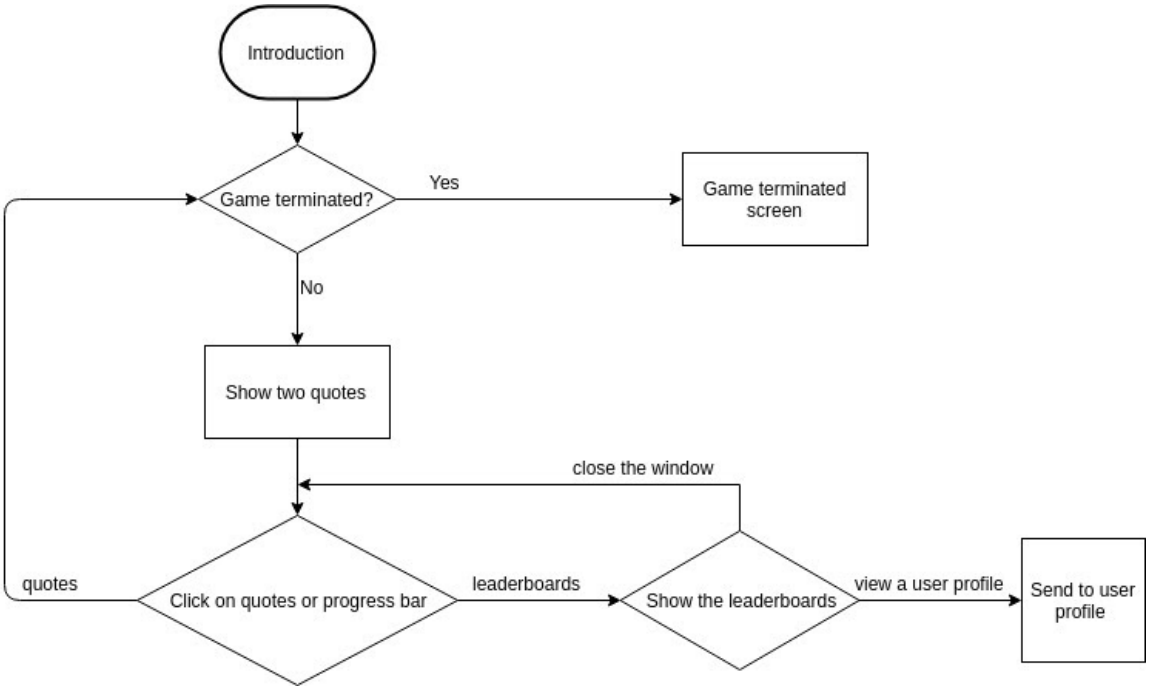


Fig. 3.5. The flowchart of the rumi quote competition.

on them to find the quotes that the user's like the most. The ranking is based on an ELO rating system. the flowchart of the process is shown in Figure 3.5. The records

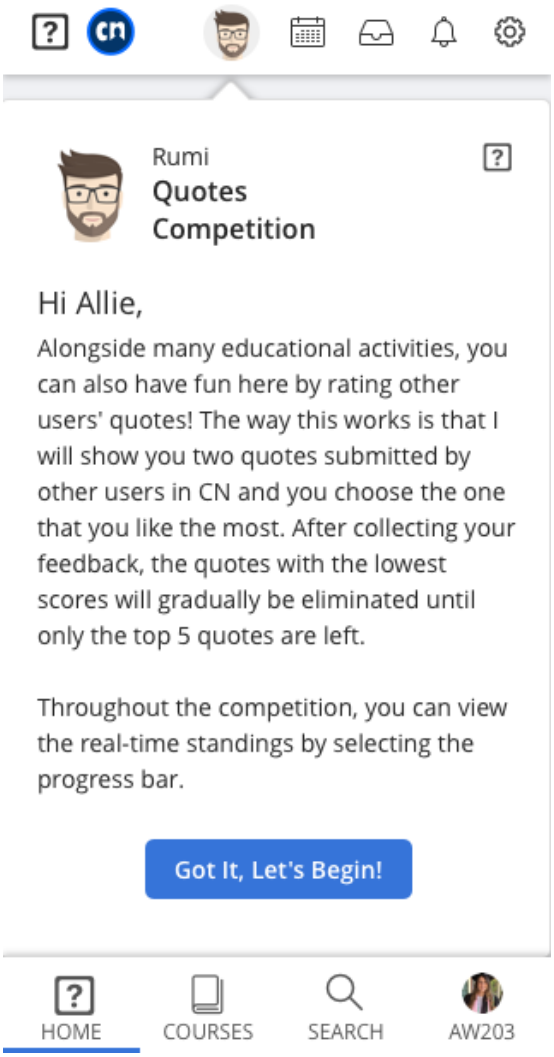


Fig. 3.6. Rumi Quote introduction page for the first-time users of the service.

of the interactions that are stored could be used as the label for the neural network to be trained on in order to better understand what types of quotes does each user prefer.

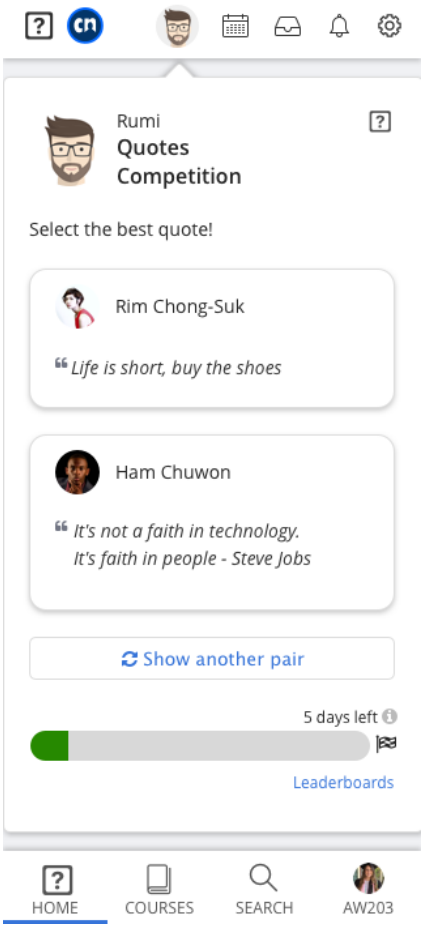


Fig. 3.7. Rumi Quotes main quote selection page.

### 3.3.1 Introduction

The first time the user quotes are shown, Rumi greetings page appears as in 3.6 and gives them an introduction. This introduction will persist until the user confirms it.

### 3.3.2 Main Quote Page

After accepting the introductory text, two quotes are shown to the user which he/she can either choose one of the quotes that they like the most, or request another pair of quotes. This page can be seen in figure 3.7

The quotes are chosen from the pool of available quotes. The pool of available quotes consists of the quotes that all the users submitted minus the following quotes:

- User's own quote
- Duplicate quotes (quotes that were shown to that user before)
- Eliminated quotes (explained later)

From the pool of available quotes, the quote which has the minimum views is chosen, this would determine the first quote. For the second quote, the quote that has the closest winrate(score/views) to the first quote is found and set as the second quote.

### 3.3.3 User Feedback

If the user selected one of the quotes, that quote will have it's score incremented by 1, while the other quote will have its score decremented by 1. Also, the view counts of both quotes are incremented by 1. If the user opted for two different quotes, the scores remain unchanged but the view count is incremented by one for both quotes.

After modifying the scores and views, winrate (= score/views) is calculated for the quotes. The quotes that have more losses than a certain threshold will be eliminated from the competition and will not be shown anymore. Currently this threshold is set as 3 meaning quotes that have 3 or more losses will be removed.

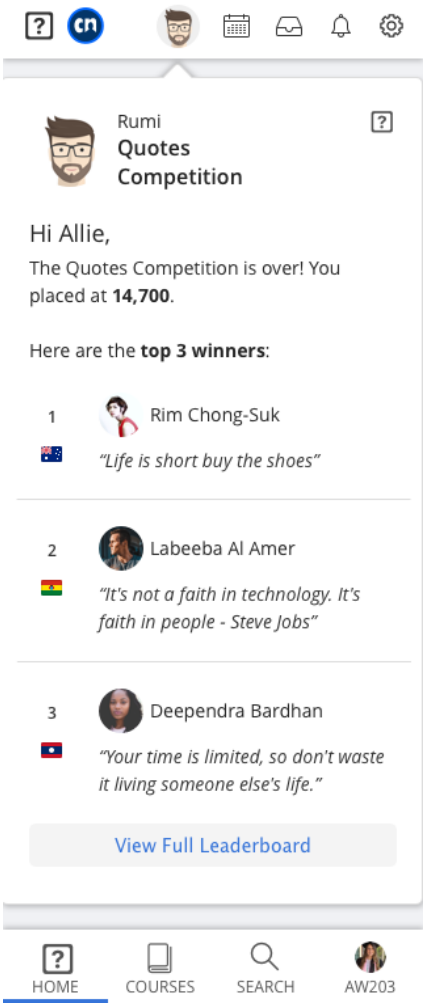


Fig. 3.8. The page that is shown to the user when the competition is terminated.

### 3.3.4 Competition Termination

As the game progresses, more quotes are eliminated and fewer quotes remain, this process continues until only 50 quotes remain, then the competition will officially end and and the competition terminated page will be shown as in Figure 3.8.

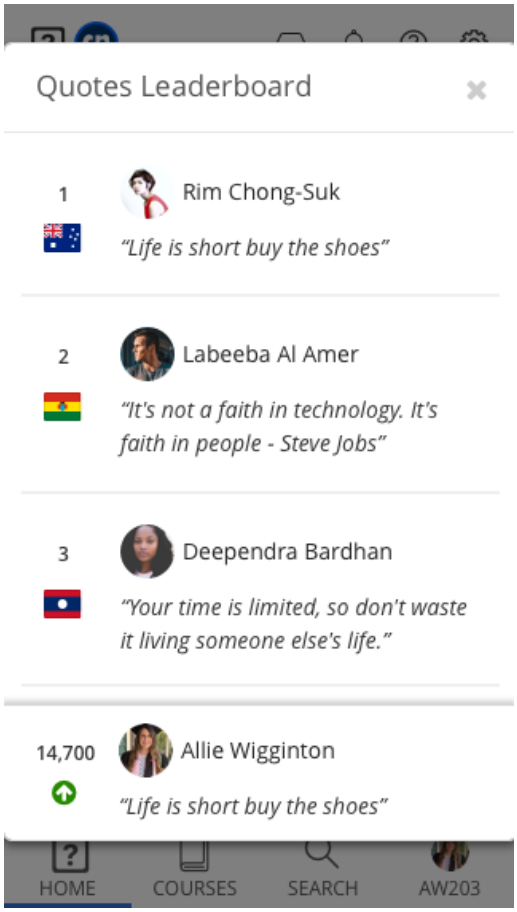


Fig. 3.9. The Rumi Quotes leaderboard page.

### 3.3.5 Leaderboard

In the leaderboard, the quotes are ranked based on score and elimination time. First, the quotes that are eliminated are ranked based on order (the sooner they eliminate, the lower the ranking is), then, the remaining quotes (the ones that are not eliminated) will be ranked based on score among themselves.

In the leaderboard page, the user will always be shown no matter the rank, and the quotes that are eliminated will be greyed out. An arrow is shown next to the user that shows how they have changed over the last day. The color is relative to

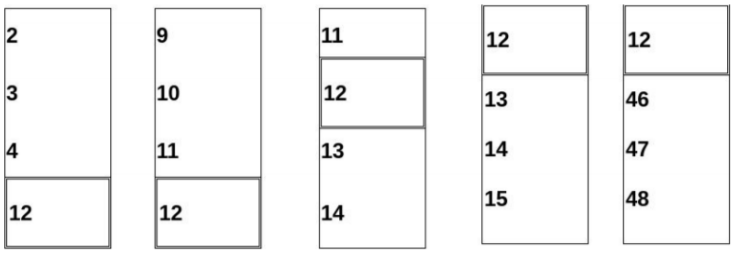


Fig. 3.10. The behavior of the Rumi Quotes leaderboard menu upon scrolling.

the user's change in ranking, so green shows improvement from last time, red means lower rank and yellow means no change.

The page view changes based on the user's placement in the rankings. The panel stays on the top when the other quotes are lower and stays on the bottom when the other quotes are higher, this behaviour is shown in Figure 3.10

In each recommendation, a visual report of the overall progress of the competition (including the estimated time left until the end of the competition) is shown and by clicking it the user will see the leaderboard. For first time viewers, the progress bar will be blinking. The first time user views the progress bar, it should be flashing to prompt the users to click on it. The leaderboard page can be seen in Figure 3.9.

### 3.3.6 Embedding Quotes

In order to embed text into feature vectors for future use with Neural Networks, a sentence embedding technique has to be used. Word2Vec has been very effective in converting words into feature vectors, however, one main issue with it is that it does not consider the context when generating word embeddings [52], hence, words with multiple meanings based on context are not going to be embedded properly. BERT [52] is a new pre-trained word embedding model which is based on attention mechanism and has achieved state-of-the-art accuracy in many text classification

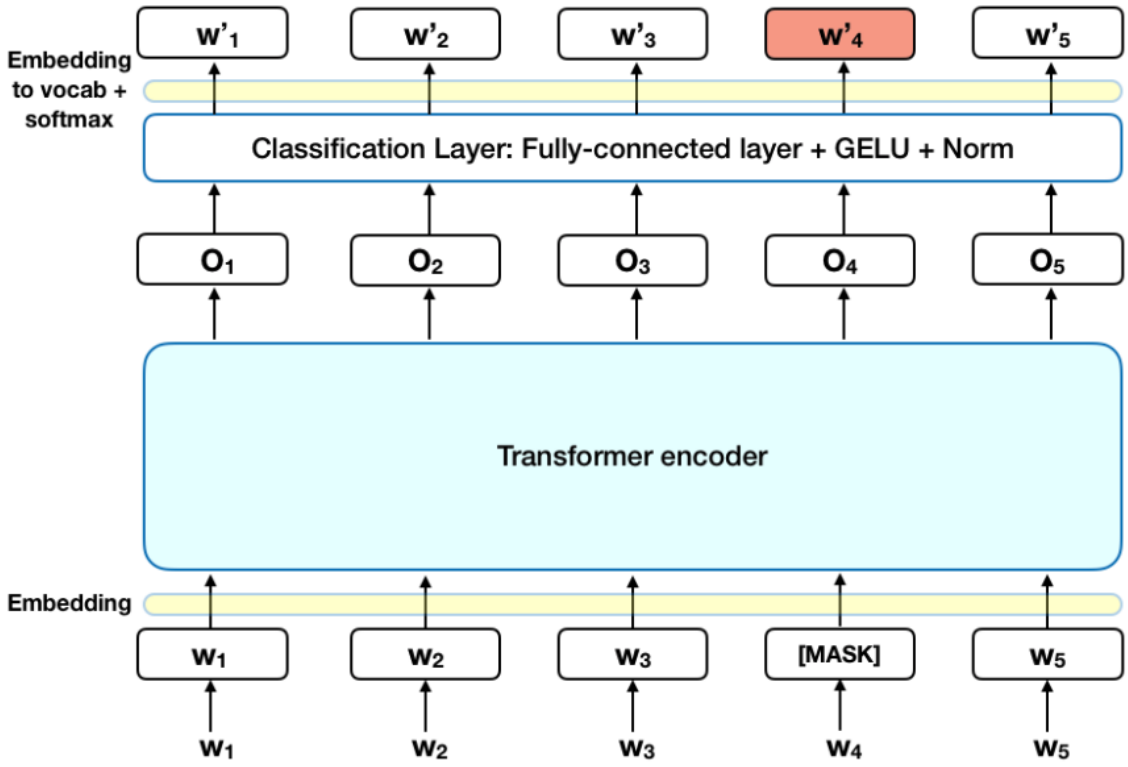


Fig. 3.11. BERT input representation. The input embeddings are derived from the sum of the token embeddings, the segmentation embeddings and the position embeddings.

tasks. BERT also produces different embeddings based on the context of the word which is very useful for short, ambiguous texts. BERT achieves this performance by converting words into tokens and then embedding the words based on their constituent tokens and their position as shown in Figure 3.11

The quotes of the users are embedded into feature vectors using the BERT pre-trained network. The resulting vector has a variable length and in order to equalize the number of features in all quotes, we take the average of the embeddings to create the final feature vector for every quote.

## 3.4 Deployment Setup

The Rumi server is written on Python language and is served using the Flask library. The whole application is containerized using Docker containers in order to have a unified environment for both development and deployment. The database that we use for storing and retrieving data is MongoDB which is a NoSQL database. The periodic tasks and time consuming jobs that don't need immediate results are done by Celery which will schedule these jobs and run them using a worker. Celery uses a message queue in order to synchronize jobs between workers, the message queue in our case is RabbitMQ.

### 3.4.1 Database

MongoDB was chosen as the database to use alongside Rumi as it had many of the desired features. MongoDB is a document-oriented database, a document is equivalent to a row or entry in the database. Each of these entries is a collection of key-value pairs. The values of these fields could be a string, integer, array, or even another nested key-value pair, these values are stored as BSON (Binary JavaScript Object Notation) format in the database. Every document in the database has a unique identifier that is used for finding that document, MongoDB refers to this identifier as "ObjectId". This identifier consists of a 4-byte timestamp, 5-byte random, and 3-byte incrementing counter which makes a total of 12 bytes. The main advantage of documents over rows in relational databases is that the document does not follow a rigid structure, and can have a fluid schema independent of the other documents [53].

One of the advantages of MongoDB is the support for vertical scaling as well as horizontal scaling. Vertical scaling is done by upgrading the hardware to increase the number of resources accessible to the software running it, while horizontal scaling scales up the application by adding identical replicas of the software and then attempting to distribute the load between them using a load balancing mechanism. Vertical scaling is easier to achieve as it does not change the structure of the code

that is being run, however, upgrading the hardware would be very expensive or even impossible beyond some point. That's why modern software must be able to scale horizontally to meet the demands.

In recent years, MongoDB was able to achieve horizontal scalability by using an architecture called "Replica Sets". Replica sets provide high availability and data redundancy features and consists of three main components. The first component is the replica set which consists of nodes that contain the data. The second component is the query router, the router receives queries from the user and then runs those query scripts on the replica set. The last component is the config server, the config server contains the centralized metadata for all the nodes and the replica set [54].

MongoDB previously followed the master/slave scheme rather strictly and therefore, was vulnerable to having a single point of failure which was the master. This problem has been resolved in the replica set architecture by creating some fallback mechanisms in case the master went offline. Nodes in replica set send heartbeats to each other to ensure the liveness and readiness of the other nodes. If a node is considered as dead based on the result of the heartbeat, the remaining nodes will hold an election to select the next master, thus, maintaining the availability of the system by using secondary nodes.

MongoDB's replica set requires a minimum of three nodes, and scaling up beyond three nodes also requires nodes to be in groups of three. A replica consists of three data-bearing members, with one of the three nodes being the primary and the other two being the secondary. However, if maintaining three data-bearing nodes is not feasible either due to costs or other reasons, one of the secondary nodes could be replaced by an arbiter. Rumi also follows this schema, the diagram of this schema is shown in Figure 3.12. In the first scenario, one of the secondary nodes will become the next primary in times of election, while in the second scenario, the arbiter would hold an election and make the decision to switch to the other secondary [54].

Both of the aforementioned scenarios can recover from the master being down, but the first scenario is generally favored because of having a higher expected availability

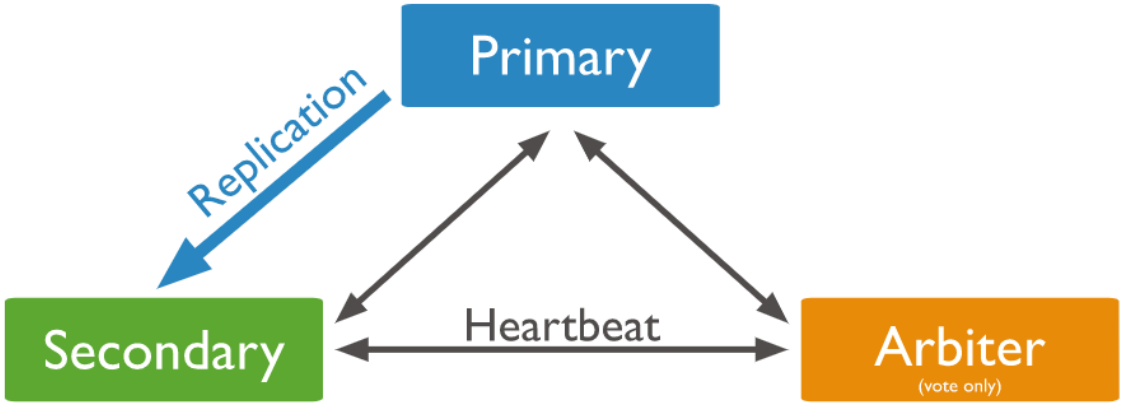


Fig. 3.12. MongoDB replicaset structure with one Primary, one Secondary and one Arbiter shards.

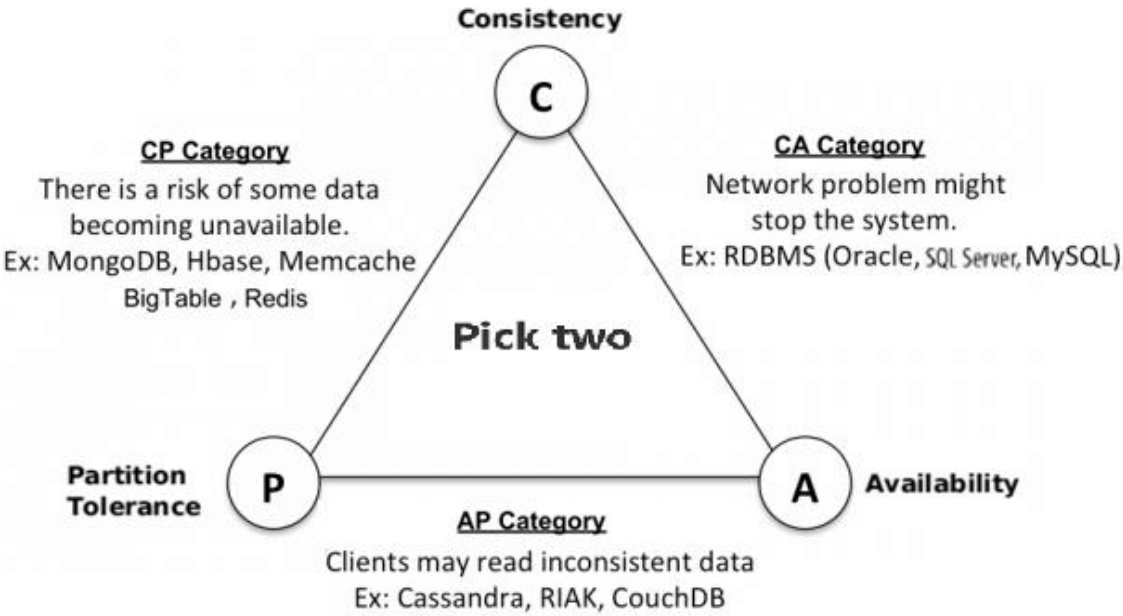


Fig. 3.13. CAP Theorem affirms that a modern database cannot satisfy all three constraints of Consistency, Availability and Partition Tolerance.

uptime due to having two data-bearing nodes. After the master is back online, it will rejoin the replica set.

There are multiple parameters to consider when choosing a database. CAP Theorem [55] states that a modern database cannot satisfy all three constraints of Consistency, Availability and Partition Tolerance, as shown in Figure 3.13 which is from **??**. For the purposes of a learning agent, we should first consider which two of the three parameters in CAP Theorem is more important for us. Rumi is going to be storing logs of most of the user's actions and preferences in the future, therefore, holding all the information in one partition for such a large amount of data may not be feasible. On the other hand, since the agent does not rely on having critical information such as passwords or payment information, immediate availability may not be as crucial for the agent. Therefore MongoDB could be a viable choice as the database for Rumi.

### 3.4.2 Cassandra

Another major competitor in the choice of the database was Apache Cassandra. Cassandra had many useful attributes that were important for a learning agent [57] and could streamline the data pipeline processes.

Cassandra is a column-oriented, open-source database written in Java that was initially developed by Facebook for handling the increasing load on their databases. This project became open-source in 2009 and was further developed by the community. Cassandra's overall data structure consists of a Keyspace, column families, rows, and columns. This structure makes it inherently NoSQL, but it also shares a lot of similarities with traditional SQL databases in terms of overall data structure [58].

The Keyspace is an abstract container or environment created to hold a logical grouping of unique identifiers for multiple datasets. There could be one or multiple column-families in each Keyspace, these column-families bundle a multitude of columns within themselves. Every row can also have several columns assigned to them, these assignments are very flexible and do not follow a rigid structure like the SQL databases do, as a result, rows in Cassandra can have a varying number of columns with any required data type. Every row in the database has a unique

identifier to differentiate it from other rows and is also indexed for fast lookups, these identifiers are called UUID (universally unique id) [58].

The way Cassandra handles the scaling is by distributing the data to multiple identical replicas of the database on multiple virtual machines, every virtual machine, in this case, is called a node. These nodes could either be physically separated on different computers or virtually separated like the server instances for cloud providers. One major difference between Cassandra and some databases like MongoDB is that Cassandra has a masterless architecture and this enables the database to be more resilient and less sensitive to the failure of one node. Every row of data is assigned to one or more nodes based on the configured number of replicas for that row. The load balancing of such replicas of data between nodes is done through a hash function that achieves near-randomness when performing data distribution [58].

As mentioned before, all these replicas of the database are identical, therefore, the initial node that they need to connect to does not matter from an end user's perspective. All the nodes have a manifest containing the location of the data regardless of whether that data resides in that particular node or not. The connections between the nodes are similar to a ring which means that each node could only directly communicate with the neighboring nodes but all the nodes are eventually interconnected and can communicate with each other in order either directly, or using neighboring nodes to retrieve the required data.

In order to ensure that the nodes in the cluster are online and ready to accept requests, and also to maintain the correctness and consistency of the data, Cassandra employs the gossip protocol. Gossip is the message system that the nodes use to transfer information regarding the state of the cluster, such as the node readiness and data replication factor. This messaging system is run every second and with each gossip, information is transferred between adjacent nodes, and eventually, through consecutive gossips, the local information is transported globally to the whole cluster and all the nodes receive the latest information about one another. This method of

transmitting information locally is very efficient and still manages to relay information to all the nodes in a short and acceptable time frame.

Cassandra is classified in the AP category in the CAP theorem3.13. This category has availability and partition tolerance but sacrifices consistency to achieve that. It should be noted that consistency, in this context, refers to the immediate consistency, and Cassandra guarantees eventual consistency for the data. In an update transaction, for example, different replicas of the data receive the update information at different times, hence, different users that are reading the data from different nodes at the specific time that the update is propagating, receive conflicting information. This issue is temporary though, and eventually, the information spreads to the whole cluster and all the nodes become up-to-date. The importance of this limitation depends on the specific use case, and that's why Cassandra is used for datasets that do not update their information frequently but they require very fast write and read speed. A good example of this use case is the storage of sensor information or user logs.

Cassandra also provides a complete query language called CQL (Cassandra Query Language) to make querying the data very intuitive for developers. This query language is very similar to SQL (Structured Query Language) which is widely used not only for querying relational databases but also for many NoSQL databases. Many Most of the people who have worked with a database are familiar with SQL scripting and on that account, can also be very productive with Cassandra and its query language even though the internals are quite different.

While Cassandra provides numerous advantages, in our specific use case, there were two main reasons for not choosing Cassandra over MongoDB:

- Our main website (thecn.com) already operates on MongoDB so accessing data in one place reduces the access latency and data migration complexity.

- Cassandra, while having a more flexible schema than SQL databases, is still not as flexible as a document database, so it may cause some complications in schema design later down the road.

# 4. NEURAL NETWORK ARCHITECTURE

In the following Section 4.0.1, we explain how we model the social network data differently from the traditional approaches. Section 4.0.2 describes the neural network model and the way it passes the data to produce latent features for nodes. Finally, Section 4.0.3 demonstrates how this framework can be used in various prediction scenarios.

### 4.0.1 Model Description

A social network can be modeled as a graph with each node representing a user, and each edge representing a connection. Each user in a social network has interactions with some items, a user can interact with multiple items, and multiple users can interact with the same item. Generally, in GNN-based social recommenders, the social network is modeled a graph of users with items as features of the users, shown as Figure 4.1. The limitation of this model is that it has little insight into the structure of the items. For instance, users who interacted with similar items hold valuable information that can be used by the recommender to gather more information about user preference. However, such information cannot be directly and explicitly extracted from this model, and the recommender has to perform extra processing to derive the data.

Typically, the items are connected to each other based on some similarity measure or equivalent metrics, such as items being in the same category, or even the items with the same ratings from the users. Often, we connect two items based on their similarity metric (e.g., cosine similarity). However, due to a large number of items, it is more computationally feasible to reduce these connections into few, powerful

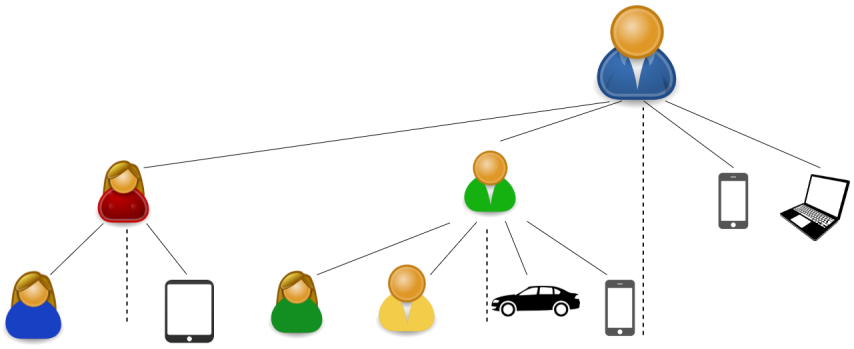

Fig. 4.1. Basic model for social recommenders.

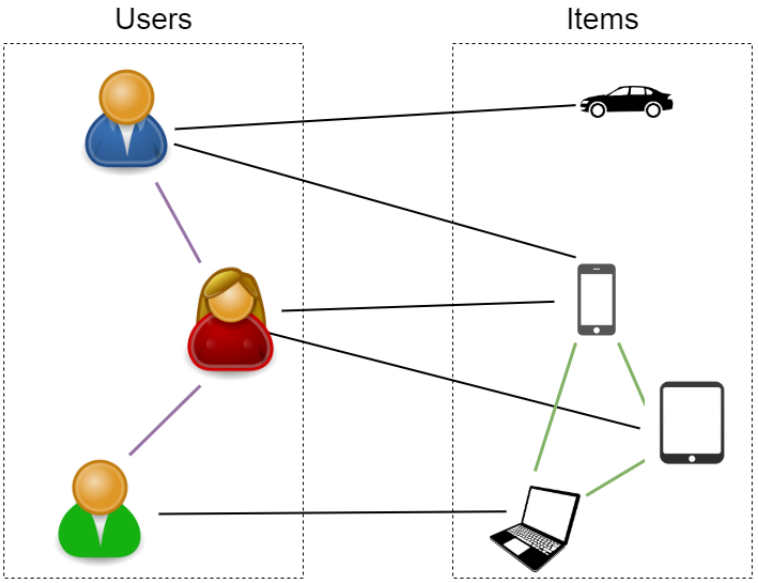


Fig. 4.2. Heterogeneous model of social recommenders.

connections. For instance, we could connect items with a cosine similarity of more than a threshold, and leave the rest of the items intact.

In this research, we propose a heterogeneous graph structure to model the social network. The heterogeneous graph consists of two types of entities – users and items, and three different types of edges which are the user-user, item-item and user-item

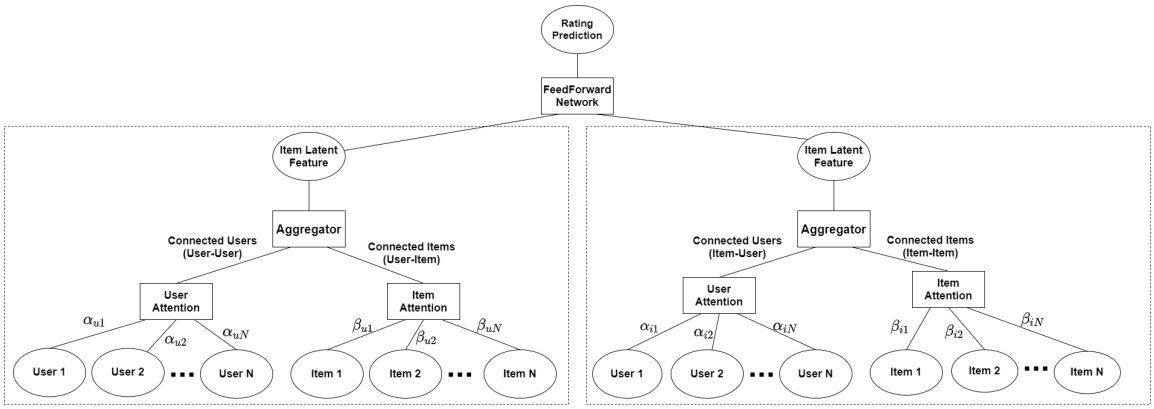


Fig. 4.3. The architecture of HeteroGraphRec, which consists of a rating predictor and two node aggregators for the target user (left section) and item (right section).

connections 4.2. This heterogeneous graph modeling of the social network has two main benefits:

- Connecting the items allows the recommender system to associate better the users who like similar items, and allows deeper connections to be more easily extracted from the network. In other words, the item-item connections in the network provide another dimension to aggregate. This dimension can be described as the structure of the items in the network, which might not add any additional information to the network per se, but it "guides" the neural network into considering the similarities between item features when making predictions.

- Items in this model can be treated almost the same way the users are, making it more convenient and versatile when designing a recommender system. Moreover, many previous graph algorithms can be applied to social data without many additional configurations because the items are not simply featured for users and are actually treated as a separate entity.

To formalize the definition of this heterogeneous social model, let $U = \{u_1, u_2, ....u_n\}$ and $V = \{v_1, v_2, ..., v_n\}$ be the sets of users and items respectively, with $n$ being the

total number of users and $m$ is the total number of items. The whole network consists of three separate graphs: user-user, item-item, and user-item graphs.

The user-user graph's adjacency matrix is $R = R^{nxn}$, in which 1 presents a pair of users in the same trust network and 0 otherwise. The user-item adjacency matrix is $R = R^{nxm}$ with each element of it being $r_ij$, which is the rating that user $u_i$ gave to the item $v_j$, if there has been no interactions between the user and the item, $r_ij = 0$ is given. Finally, the item-item adjacency matrix is $R = R^{mxm}$, and the connections between items are based on similarity. In our case, the matrix element of $s_ij$ is 1 if the two items $i$ and $j$ are very similar. Otherwise, it is set as 0.

We also define feature vectors for the nodes in the graph. The features of the users and items in the network could be represented as a binary sparse feature vector. However, such vectors grow very large and become inefficient when dealing with large numbers of users and items. The goal of the embedding layer is to convert each user and item into a dense vector $\{\vec{h_{u1}}, \vec{h_{u2}}, ..., \vec{h_{uN}}\}$ and $\{\vec{h_{i1}}, \vec{h_{i2}}, ..., \vec{h_{iN}}\}$ respectively. These embeddings are updated throughout the training to extract the latent features.

To demonstrate the use of such heterogeneous graph structure for a rating prediction task, the HeteroGraphRec model is proposed. First and foremost, the social network is modeled as a heterogeneous graph. Then, the users and items embeddings are generated to feed to a neural network for prediction.

The overall architecture of HeteroGraphRec model is shown in Figure 4.3. The framework consists of three main modules: user aggregator, item aggregator, and predictor. The user aggregator consists of two attention networks. The user-user attention network finds the important users from their connections with attention coefficients $\alpha_{u1}, \alpha_{u2}, ...$, whereas the user-item attention network chooses the important items that the user has previously interacted with, with attention weights of $\beta_{u1}, \beta_{u2}, ....$ The results of the two attention networks are merged to find the user latent features. The item aggregator also takes a similar approach to merge the results of the two attention networks to create item latent features. The item-user attention network chooses the important users that have interacted with that specific item with

a weight of $\alpha_{i1}, \alpha_{i2}, ...$, and the second attention network (item-item) chooses similar items that the network should attend to, with weights $\beta_{i1}, \beta_{i2}, ....$ Finally, the predictor uses the two latent features of users and items to make the final prediction about the ratings.

### 4.0.2 Node Aggregations

The node aggregator gathers information from two spaces, the user space, and the item space. For a user, the user space holds the connections to adjacent users, while the item space holds the items that the user has interacted with. For an item, the user space is the set of users that interacted with that specific item, and the item space is the set of similar items to the target item.

The user space aggregator receives a set of node features $H_u = \{\vec{h_{u1}}, \vec{h_{u2}}, ..., \vec{h_{uN}}\}$ and $h_i \in R^F$, where $N$ is the number of nodes(users), and $F_u$ is the number of features in each node. The layer produces a new set of node features (of potentially different cardinality F') $H'_u = \{\vec{h'_{u1}}, \vec{h'_{u2}}, ..., \vec{h'_{uN}}\}$.

To obtain sufficient expressive power to transform the input features into higher-level features, a shared linear transformation, parametrized by a weight matrix, $W \in R^{F'xF}$, is applied to every node. We then perform self-attention on the nodes -— a shared attention mechanism $a : R^{F'}xR^{F'} \rightarrow R$ computes attention coefficients mentioned below:

$$e_{ij} = a(W\vec{h}_{ui}, W\vec{h}_{uj}) \tag{4.1}$$

These attention coefficients indicate how important one node is to the other. These coefficients need to be computed only nodes that are actually connected. We perform masked attention to ensure that only the neighboring node's attention coefficients are calculated, as shown in the equation below:

$$\alpha_{ij} = \text{softmax}_j\left(e_{ij}\right) = \frac{\exp\left(e_{ij}\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(e_{ik}\right)} \tag{4.2}$$

In this experiment, the attention mechanism $a$ is a single layer feed-forward neural network, parametrized by a weight vector $\overrightarrow{\mathbf{a}} \in R^{2F}$ and passed through a Leaky ReLu activation function. The resulting coefficients can be expressed as:

$$\alpha_{ij} = \frac{\exp\Big(\text{ LeakyReLU } \Big(\overrightarrow{\mathbf{a}}^T \Big[\mathbf{W}\vec{h}_{ui} \| \mathbf{W}\vec{h}_{uj}\Big]\Big)\Big)}{\sum_{k\in\mathcal{N}_i} \exp\Big(\text{ LeakyReLU } \Big(\overrightarrow{\mathbf{a}}^T \Big[\mathbf{W}\vec{h}_{ui} \| \mathbf{W}\vec{h}_{uk}\Big]\Big)\Big)} \tag{4.3}$$

Once the attention coefficients are calculated, the output of the attention layer is derived by computing the linear combination of the attention coefficients and the corresponding node features, and applying a nonlinearity:

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}\vec{h}_{uj}\right) \tag{4.4}$$

Employing single attention is unstable and can sometimes result in losing important information. In order to stabilize the learning process and increase representational power in our model, multi-head attention is used. The multi-head attention employs $k$ independent attention heads and concatenates the results, as mentioned in [59]:

$$\vec{h}'_i = \|_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_{uj}\right) \tag{4.5}$$

The item space aggregator is identical to the user space aggregator in architecture, with only the dimensions of features being different due to aggregating another entity. The item space aggregator finds the important items for the user by following the steps mentioned above, assuming the item feature inputs are $H_i = \{\vec{h_{i1}}, \vec{h_{i2}}, \ldots, \vec{h_{iM}}\}$ with M being the total number of nodes(items) and $F_i$ being the features of each node. The output of the item aggregator is the feature set $H'_i = \{\vec{h'_{i1}}, \vec{h'_{i2}}, ..., \vec{h'_{iM}}\}$ with each node having $F'_i$ features.

Finally, after aggregating the user space and item space information, the resulting feature vectors are concatenated and fed into a feed–forward network with Leaky

ReLu nonlinearity. The final feature vector $H = \{\vec{h_1}, \vec{h_2}, ..., \vec{h_N}\}$ with each node having $F'$ features, the output features can be calculated using the formula below:

$$h'_i = LeakyReLu\left(W\left(h'_u \parallel h'_i\right)\right) \tag{4.6}$$

The node aggregators can be stacked for multiple layers and for different spaces (both user and item) to search the network deeper and produce richer features. However, it is noteworthy to mention that adding more aggregation layers expands the search radius of the neural network exponentially. It can cause performance issues without adding too much important information.

### 4.0.3 Predictions

One of the significant advantages of this model is that the users and items are aggregated similarly but independently. This process simplifies the process of social recommendations to the choice of predictor specific to the problem. For example, if the objective is to predict the rating of an item given by a user, the aggregated features of the target user and item can be fed to a feed-forward network to predict the ratings. On the other hand, to predict whether two users are in the same trust network or not, the predictor's input would be the aggregated features of the target users. Likewise, if the objective is to measure the similarity between two items, or whether or not they could be bundled, the predictor's input would be the aggregated features of the target items.

In this work, the specific case of rating prediction is chosen as the prediction problem. Therefore, the user and item aggregated information is fed to a feed-forward network to predict the ratings. Assuming $O_u$ is the final aggregated user features, and $O_i$ are the final aggregated item features, the rating is predicted using the formula below:

$$r = LeakyReLu\left(W\left(O_u \parallel O_i\right)\right) \tag{4.7}$$

# 5. BALANCED RANDOM WALKS

The main objective in language modeling is to estimate the likelihood of a specific sequence of words appearing in a corpus. Equivalently, we can perform a series of walks on a graph to turn them into a sequence of nodes. By that definition, our problem can be formally defined as estimating the probability of observing vertex $v_i$ given all the previous vertices visited:

$$\Pr\left(v_i | \left(v_1, v_2, \cdots, v_{i-1}\right)\right) \tag{5.1}$$

Assuming our network graph is $G = (V, E)$, we define the mapping $\Phi = V \epsilon v \longrightarrow R^d$ as mentioned in [8]:

$$\Pr\left(v_i | \left(\Phi\left(v_1\right), \Phi\left(v_2\right), \cdots, \Phi\left(v_{i-1}\right)\right)\right) \tag{5.2}$$

Calculating the Equation 5.2 mentioned above is computationally expensive, so two assumptions are made to have the calculations more feasible. First, instead of using the context to predict a node, it uses one node to predict the context. Secondly, the context is composed of the adjacent nodes (both left and right) with any order, meaning that there are no differences between the adjacent nodes in the context. This simplifies our algorithm to Equation 5.3.

$$\underset{\Phi}{\text{maximize}} \quad \log \Pr\left(\left\{v_{i-w}, \cdots, v_{i+w}\right\} \setminus v_i | \Phi\left(v_i\right)\right) \tag{5.3}$$

In the following sections, in Section 5.1, we describe how node sequences are generated using balanced random walks. Section 5.3 presents how the model learns the representations based on the input sequences.

### 5.1 Walk Prediction

As mentioned in the previous sections, node occurrences in random walks diverge from the degree distribution based on the graph topology. Initially, we calculate the frequency of each node in a random walk to adapt our walking strategy. The first order of the adjacency matrix of a graph shows the possible destinations after a single walk. Normalizing each row of this matrix shows the probability of moving to a node from a chosen source node. By summing over the column of this normalized adjacency matrix results in a vector, we can calculate the frequency of a destination node through a random walk of length 1.

The same process can be repeated for higher powers of the adjacency matrix, such as $A^n$, outputs a similar frequency vector except that the destination is reached after $n$ walks. Through adding these vectors together, it shows the frequency of passing through each node in $n$ walks.

**Algorithm 1** Node frequency measurement in a random walk

**Input:** Graph G=(V, E, W),

**Output:** f node frequencies

1: $A$=Adjacency Matrix(G)
2: $A_{norm}$ = normalize(A) over rows
3: $A^n$ = normalize(A) over rows
4: $W = normalizeddegree_{histogram}(G)$
5: **for** $i = 1$ to walk length **do**
6: $\quad A^n = matmul(A^n, A_{norm})$
7: $\quad W = W + \sum_{<i>} A^n_{i,j}$
8: **end for**
9: $W_{norm}$ = normalize(W) over rows
10: **return** $W_{norm}$

The adjacency matrix of a graph shows possible transitions from each node to another. By adding all the values in a column, we can acquire the number of times that a node was walked on in a single step. The second order of adjacency matrix has a similar effect except that the values are calculated for two steps. Continuing for higher orders of the adjacency matrix provides us with a simulation of the walker trajectory in a random walk. Finally, summing over all the values for each node in every order provides us with an estimate of the number of times that a specific node is walked on.

Having the estimation for each specific node, we can observe that the number of times each node is walked on is different from the degree of the node. Although this calculation adds some overhead to the random walk algorithm, the calculation is done only once for the graph and the sparsity of the adjacency matrix to keep this overhead to the minimum.

To choose the initial node, the probability of initially choosing a node to walk on is proportional to their degree, this ensures that the initial values follow the Zipf distribution. The algorithm 1 measures the frequency of each node in a random walk using the graph adjacency matrix.

Assuming the graph has E edges and V nodes, the adjacency matrix is $O(E)$ space complexity because of the sparse nature of it, however, the $A^n$ is a space complexity between $O(E)$ and $O(V^2)$ and this complexity gravitates towards the latter since the matrix becomes dense as $n$ increases. The space required for storing the frequencies, on the other hand, is $O(V)$ and therefore, comparatively trivial. The process of matrix multiplication, addition and normalization altogether have a time complexity of $O(N)$ with $N$ being the number of non-zero elements in the matrix. So for $A^n$, the time maximum complexity is $O(V^2)$.

### 5.2 Balanced Random Walk

After finding the frequency of the nodes in a random walk, these frequencies are compared with the degree distribution to modify the node preferences when walking. The nodes with a higher frequency compared to their degree are at a penalty; conversely, the nodes with a lower frequency compared to their degree are preferred during the walk. The algorithm 2 performs the comparisons and carries out the balanced walk.

**Algorithm 2** Balanced Random Walk

**Input:** Graph G=(V, E, W), $W_{norm}$

**Output:** W walks

1: **for all** nodes **do**
2: $coefficient[node] = degree[node]/W[node]$
3: **end for**
4: **for** $i = 1$ to number of walks **do**
5: $weight[node] = degree[node]$
6: initial node = choose random node with given weight
7: $weight[node] = coefficient[node]$
8: walk = random weighted walk from node with the given weight
9: append walk to walks
10: **end for**
11: **return** walks

The initial value of $A_0$ ensures that the short walks are close to the power-law while the following weights ensure that the walks do not diverge from this initial value. It is also worth noting that random walks are random, and the variance of the node frequencies increase as the walk becomes longer. Since our objective is to ensure the walks follow the power-law on average, the balanced walks produce a more consistent output than the random walk.

### 5.3 Learning Features

After performing the balanced random walks, the strings of nodes have been created. First, these strings are shuffled to randomize their order, and then they fed to the Skip Gram model to extract the representations of the nodes. The training objective of the original Skip Gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document. Applying full softmax in training is not efficient for large networks because it evaluates all the outputs to obtain the results; therefore, other methods have replaced it.

Hierarchical Softmax is a computationally efficient approximation of Softmax. Instead of evaluating all the output nodes, hierarchical Softmax only evaluates the log base 2 of the outputs, resulting in significant computational gains. Noise Contractive Estimate (NCE) loss is an alternative to hierarchical Softmax. NCE loss reduces the language model estimation problem to the problem of estimating the parameters of a probabilistic binary classifier that uses the same parameters to distinguish samples from the empirical distribution from samples generated by the noise distribution. Essentially, this is more computationally attainable in large networks than Softmax since it subsamples frequent words instead of considering all the words in the vocabulary. The Skip Gram model is concerned with learning high-quality vector representations, so we are free to simplify NCE as long as we maintain the vector representation quality. The negative sampling algorithm is then defined as:

$$\log \sigma \left( {v'_{n_O}}^{\top} v_{n_I} \right) + \sum_{i=1}^{k} E_{n_i \sim P_n(n)} \left[ \log \sigma \left( -{v'_{n_i}}^{\top} v_{n_I} \right) \right] \tag{5.4}$$

Where $P_n$ is the noise distribution, $n_1, n_2, ..., n_i$ is the sequence of nodes and $\sigma(x) = 1/(1+exp(x))$. In language modeling, in very large corpora, the most frequent words can easily occur hundreds of millions of times (e.g., “in”, “the”, and “a”). Such words usually provide less information value than rare words. The same phenomenon happens in social networks, a few users possess most of the edges in the network;

[h]

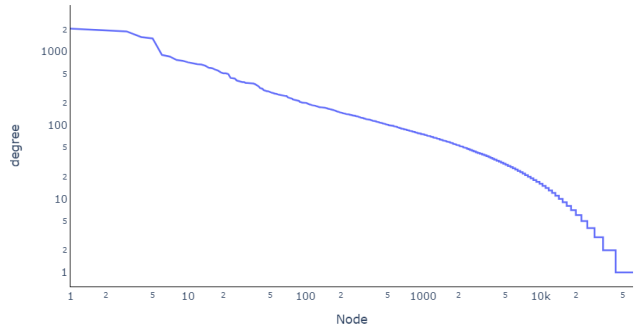


Fig. 5.1. Node frequency in CourseNetworking user-follower network in log-log scale.

Table 5.1.
Binary operators for learning edge features from node embeddings.

| **Operator** | **Symbol** | **Definition** |
|---|---|---|
| **Average** | $\oplus$ | $[f(u) \oplus f(v)]_i = \frac{f_i(u)+f_i(v)}{2}$ |
| **Hadamard** | | $[\mathrm{f(u)\ f(v)}]_i = f_i(u) * f_i(v)$ |
| **Weighted-L1** | $\|\cdot\|\overline{1}$ | $\|f(u) \cdot f(v)\|_{\overline{1}i} = \|f_i(u) - f_i(v)\|$ |
| **Weighted-L2** | $\|\cdot\|\overline{2}$ | $\|f(u) \cdot f(v)\|_{2i} = \|f_i(u) - f_i(v)\|^2$ |

therefore, they don't contain valuable information in most cases. To counter the imbalance between the rare and frequent nodes, assuming $f(n_i)$ is the frequency of the $i$'th node, a simple subsampling approach was used which the nodes in the training set is discarded by:

$$P\left(n_i\right) = 1 - \sqrt{\frac{t}{f\left(n_i\right)}} \tag{5.5}$$

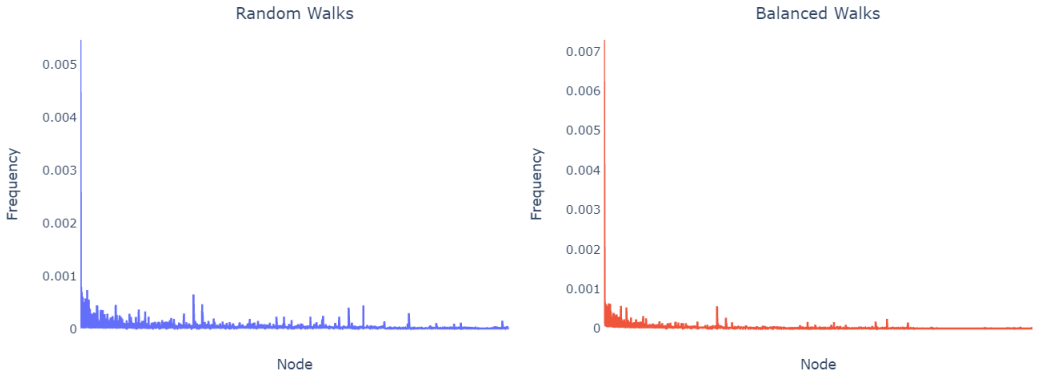


Fig. 5.2. Node frequencies in a Random Walk (left) and Balanced Walk (right) of length 8. (The nodes are sorted by degree.)

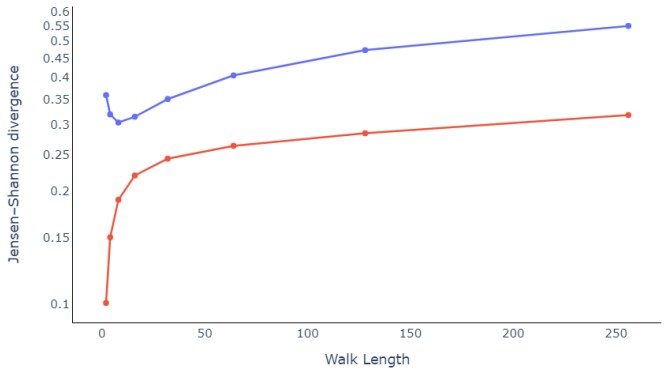


Fig. 5.3. Jensen-Shannon divergence of random and balanced walks based on the walk length.

The context size is one of the primary parameters of the training, nodes in the context window can be paired together and fed into the network. Having a larger window allows our model to understand the more distant features in our network better. But the larger the context size, the more data it requires to train the model. It may not be necessary for most of the networks.

In a directed graph, each edge contains information about the directions too, therefore, the direction of the context is very helpful and depends on the type of data under study.

# 6. EXPERIMENTAL RESULTS

The experimental results are divided to two sections, HeteroGraphRec and Balanced Random Walks. The former shows the experimental setup for the new Neural Network architecture that is introduced, while the latter studies the Balanced Random Walk node embedding in independent sets of experiments.

## 6.1 HeteroGraphRec

### 6.1.1 Experimental Setup

#### Datasets

We test our HeteroGraphRec model on three real-world datasets: Ciao [60], Douban [61] and Epinions [62]. The details of these datasets are listed in Table 6.1.

- **Ciao:** Ciao is a real-world dataset published by Guo et al. [15] which contains movie rating records of users and trust relations between them.
- **Douban:** Douban Movie is a Chinese website that allows Internet users to share their comments and viewpoints about movies. Users can post short or long comments on movies and give them marks. The dataset was published by Zhong et al. [64] in 2014.
- **Epinions:** Epinions is another social network in which users rate items and issue trust statements. It was published on the LIBREC website.

These social networks allow users to rate items from 1 to 5 and add other users to their trust network. To model these social networks as a heterogeneous graph, we

Table 6.1.
General statistics about the size of the social networks including their users, items, and the connections between them.

| Dataset | Epinions | Ciao | Douban |
|---|---|---|---|
| #users | 49,289 | 17,589 | 24,562 |
| #items | 22,173 | 16,121 | 46,643 |
| #ratings | 138,207 | 62,452 | 244,936 |
| #social | 487,183 | 40,133 | 513,010 |

must first create the item-item connections. One way to develop such connections is to connect all the items that are in the same category. However, this approach could create dense graphs for these specific datasets, making the model computationally expensive. Hence, we made a reasonable compromise and grouped items based on both their categories and the ratings that they received from users to create a sparse graph where the items connected had a strong similarity. We demonstrate that even a few meaningful item connections like this can considerably improve the accuracy of the predictions.

### Evaluation Metrics

The performance of HeteroGraphRec is evaluated by the measured error between the actual rating and the predicted rating. The two performance metrics used for this evaluation are Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). The objective of the model is to minimize these error values. It is also worth noting that even small reductions in these metrics can significantly improve top-k recommendations [65].

**Baselines**

In this research, our HeteroGraphRec model is compared with GraphRec and three other groups of methods including traditional recommender systems, traditional social recommender systems, and deep neural network based recommender systems. The compared baselines are listed below:

- **PMF [66]:** Models latent factors of users and items by applying Probabilistic Matrix Factorization on the user-item matrix.
- **SoRec [16]:** Performs co-factorization on two user-item adjacency matrix and user-user adjacency matrix.
- **SoReg [11]:** Constrains the matrix factorization framework by modeling social network information as regularization terms.
- **SocialMF [67]:** Incorporates the trust information into the matrix factorization model and propagates it in the network.
- **TrustMF [17]:** Considers trust as a directional property, and using matrix factorization, maps the users into two spaces of truster space and trustee space.
- **NeuMF [68]:** Introduces Neural Network-based Collaborative Filtering approach, which enhances the performance of Collaborative Filtering by replacing the inner product with a neural architecture. We adjust its loss to the squared loss for the rating prediction task since the original implementation is for recommendation ranking task.
- **DeepSoR [25]:** Proposes a deep neural network-based model to learn nonlinear features of each user from social relations and to integrate into probabilistic matrix factorization for rating prediction problem.
- **GCMC+SN [39]:** Presents a graph auto-encoder framework based on differentiable message passing on the bipartite interaction graph. To incorporate

social network information into GCMC, they utilize the Node2Vec [8] to generate user embedding as user side information instead of using the raw feature social connections $T \in R^{NxN}$ directly due to the raw features being very large and inefficient in size as mentioned before.

- **ScAN [41]:** Employs a co-attention neural network, which learns the influence value between the user and her friends from the historical data of the interaction between the user/her friends and an item. The social data in this network is initialized using Node2Vec as well.

- **GraphRec [42]:** Utilizes attention mechanism to intelligently aggregate information on both user and item space. the information about the user's interaction history and trust network is collected from the user space, and information about the users that used the same items are collected in the item space.

The compared models above fall into three major categories. PMF and NeuMF are pure collaborative filtering models with no social network information for rating prediction, and three baselines of DeepSoR, GCMC+SN, and GraphRec are neural network-based social recommenders.

### HyperParameter Setting

We split our dataset into training, validation, and test sets with two different percentage values. These values were once set as 80%, 10%, 10%, and another time as 60%, 20%, 20% respectively. The neural network has two stacked node aggregators for both user and item space, and the weights of the network were initialized with the Xavier Initialization method [69]. Leaky ReLu was chosen as the activation function. Early stopping strategy with a patience value of 10 was adopted for the training process. The experimental results are repeated for ten random seed initialization and are statistically significant, with a p-value of less than 0.05. The default value for user embedding size and item embedding size is 64 and 128 respectively.

Table 6.2.
MAE Performance of HeteroGraphRec compared to other recommender systems

| 2* Dataset | Algorithms | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMF | SoRec | SoReg | SocialMF | TrustMF | NeuMF | DeepSoR | GCMC+SN | ScAN | GraphRec | **HeteroGraphRec** |
| Ciao(60%) | 0.956 | 0.8489 | 0.8967 | 0.8394 | 0.7622 | 0.8294 | 0.7896 | 0.7741 | 0.7629 | 0.7566 | 0.7414 |
| Ciao(80%) | 0.8975 | 0.8548 | 0.8519 | 0.8307 | 0.7628 | 0.7993 | 0.7685 | 0.7511 | 0.7495 | 0.7406 | 0.7304 |
| Douban(60%) | 0.9701 | 0.8542 | 0.9114 | 0.8517 | 0.7730 | 0.8613 | 0.8009 | 0.7880 | 0.7816 | 0.7638 | 0.7571 |
| Douban(80%) | 0.9289 | 0.8612 | 0.8745 | 0.8348 | 0.7806 | 0.8131 | 0.7841 | 0.7752 | 0.7638 | 0.7521 | 0.7427 |
| Epinions(60%) | 1.0211 | 0.9086 | 0.9412 | 0.8965 | 0.8550 | 0.9097 | 0.8520 | 0.8602 | 0.8548 | 0.8476 | 0.8367 |
| Epinions(80%) | 0.9918 | 0.8916 | 0.9169 | 0.8794 | 0.8376 | 0.9028 | 0.8229 | 0.8607 | 0.8441 | 0.8192 | 0.81037 |

Table 6.3.
RMSE Performance of HeteroGraphRec compared to other recommender systems

| 2* Dataset | Algorithms | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | PMF | SoRec | SoReg | SocialMF | TrustMF | NeuMF | DeepSoR | GCMC+SN | ScAN | GraphRec | **HeteroGraphRec** |
| Ciao(60%) | 1.1948 | 1.0719 | 1.102 | 1.0637 | 1.0513 | 1.0851 | 1.0488 | 1.0209 | 1.0184 | 1.0093 | 0.9980 |
| Ciao(80%) | 1.1197 | 1.0704 | 1.0811 | 1.0572 | 1.0422 | 1.0655 | 1.0348 | 1.0252 | 1.0207 | 0.9813 | 0.9721 |
| Douban(60%) | 1.2241 | 1.1024 | 1.1943 | 1.0718 | 1.0681 | 1.0977 | 1.0553 | 1.0361 | 1.0257 | 1.0105 | 1.006 |
| Douban(80%) | 1.1546 | 1.0924 | 1.0958 | 1.0623 | 1.0590 | 1.0852 | 1.0472 | 1.0410 | 1.507 | 1.449 | 0.9962 |
| Epinions(60%) | 1.2814 | 1.1496 | 1.1877 | 1.1380 | 1.1526 | 1.1627 | 1.1205 | 1.0981 | 1.0968 | 1.0907 | 1.0755 |
| Epinions(80%) | 1.2156 | 1.1582 | 1.1634 | 1.1478 | 1.1484 | 1.1504 | 1.0961 | 1.0692 | 1.0687 | 1.0647 | 1.0483 |

### 6.1.2 Rating Prediction Performance

We compared our HeteroGraphRec model to the baselines mentioned above in Tables 6.2 and 6.3. Our model outperforms all the previous baselines in both MAE and RMSE performance. These results confirm that PMF has the lowest accuracy since it performs Collaborative Filtering on ratings only. NeuMF outperforms PMF, which suggests that neural networks can have improvements over Collaborative Filtering methods.

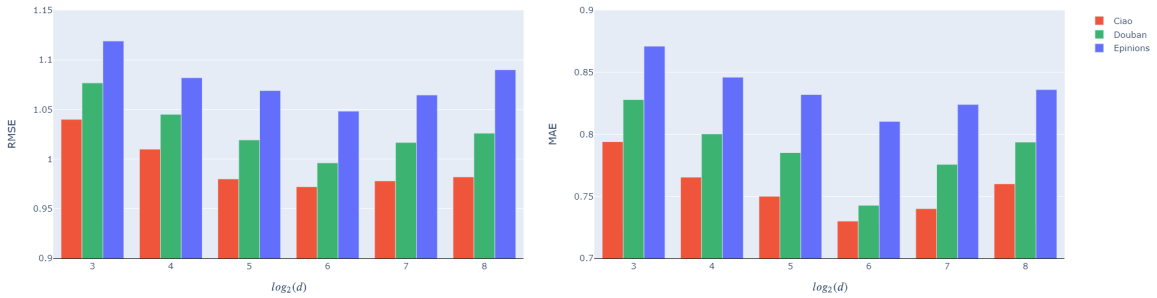


Fig. 6.1. Sensitivity of the model to user embedding dimension size.

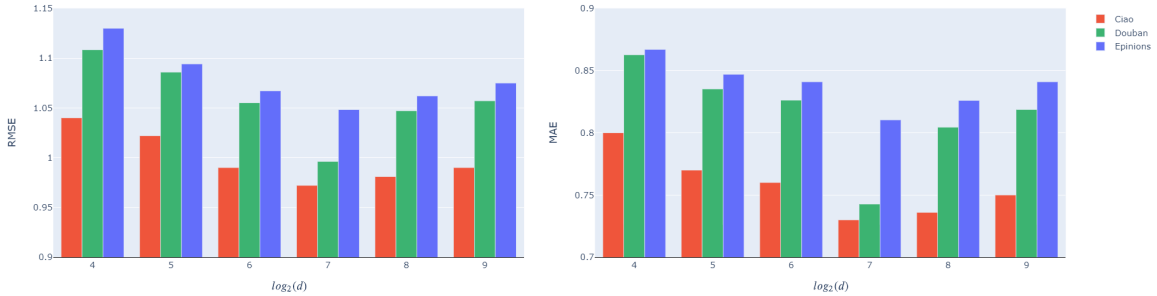


Fig. 6.2. Sensitivity of the model to user embedding dimension size.

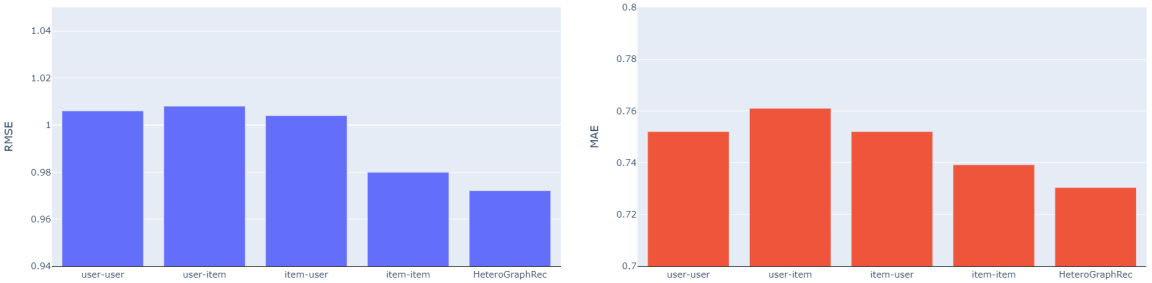


Fig. 6.3. Sensitivity of the model to each of its four components in the Ciao
dataset.

SoRec, SoReg, SocialMF, and TrustMF perform matrix factorization on not only the ratings but also the social info. Hence, they are more accurate than PMF as well.

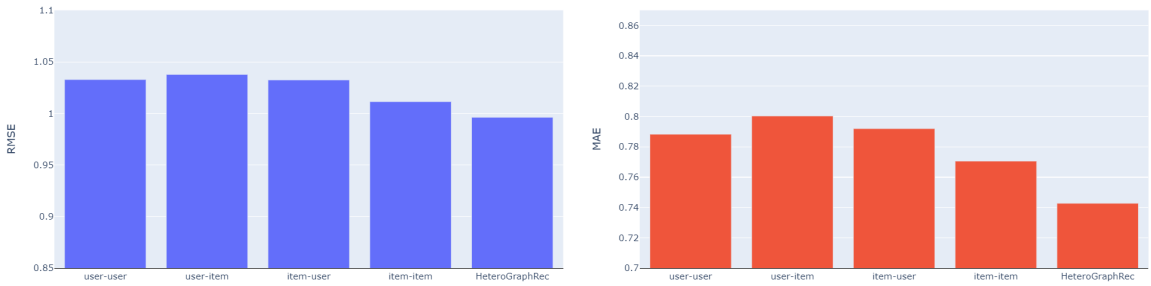


Fig. 6.4. Sensitivity of the model to each of its four components in the Douban
dataset.

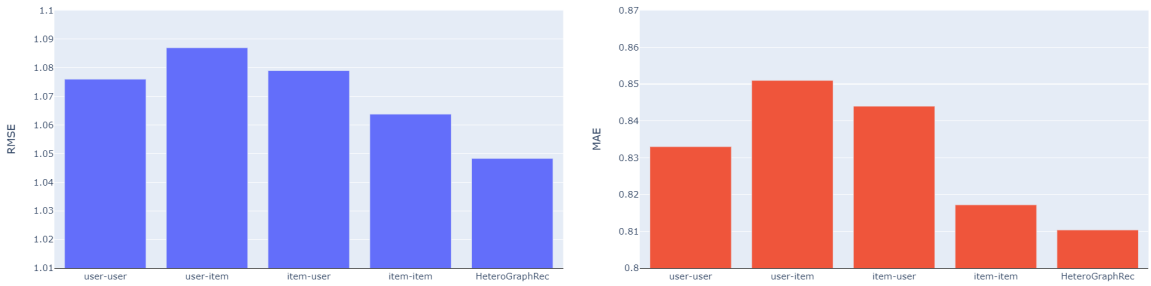


Fig. 6.5. Sensitivity of the model to each of its four components in the Epinions dataset.

DeepSoR and GCMC+SN, similar to NeuMF, utilize the neural network structure to boost the performance of the model. The performance of GCMC+SN is a testament to the potency of GNNs in graph prediction tasks.

ScAN surpassed the previously mentioned baselines, which shows the importance of attention mechanism in neural recommenders. GraphRec's performance exceeded even ScAN because of the additional item space aggregations using a similar attention mechanism.

To sum up, our model consistently outperforms all the baselines, as mentioned above, due to the inherent strength of attention networks. The improved social mod-

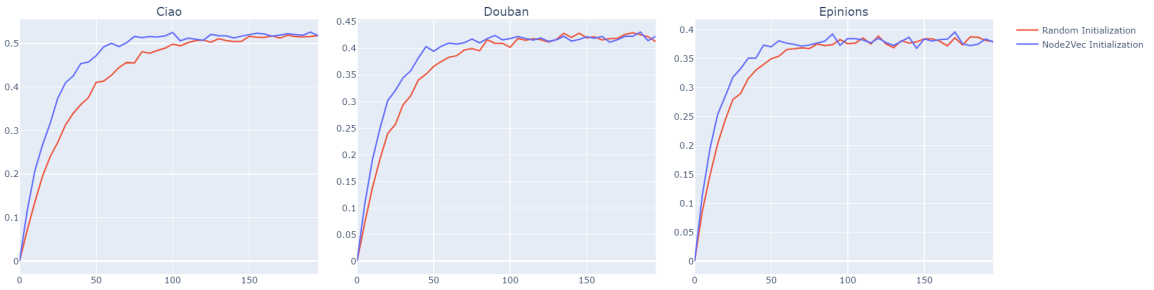


Fig. 6.6. Performance of HeteroGraphRec@10 throughout the training process with using random and node2vec initializations.

eling provided more profound insight into the data and allowed for more dimensions for information collection.

### 6.1.3 Model Analysis

#### Embedding Dimensions

We measured the sensitivity of our model to the sizes of user and item embedding size in Figures 6.1 and 6.2 respectively. The x-axis is the log of the embedding dimension. All the other parameters are set to default except the one understudy.

The performance of the model drops at smaller embedding sizes due to the embeddings being unable to contain all the necessary information related to the nodes. Increasing the number of embedding dimensions beyond some point increases the time required for the training. The reason is that the model may face complications in convergence due to the increasing number of variables that need to be tuned – the larger number of items required larger embedding dimensions than the users to contain their information. Epinions was the most sensitive dataset to the embedding dimension size, while Ciao relied on accurate tuning as the model's performance varied less when changing the values.

### Model Components

In order to demonstrate the interpretability of the model, we evaluated the affect of each component demonstrated in Figure 4.3. The functional effects of each component are detailed as below:

- **user-user:** This component measures the influence of a user's fiend or trust network in their preferences. This information represents the user-user connections in the model.
- **user-item:** This component of the model quantifies the value of the user's interaction history and how much it identifies the user's preferences. .
- **item-user:** This component analyzes the effect of other users that have interacted with the same item, which is almost equivalent to Collaborative Filtering, as it collects preferences from other users.
- **item-item:** This component shows the amount of information the neural network can gain from an item by referring to other items with similar characteristics.

We disabled one component at a time, then measured the accuracy of the model to view the effects of a specific component in deriving the final output. Figures 6.3 to 6.5 show the performance of the model with one of the four components being disabled. For example, RMSE or MAE of user-user means the performance of the model when the user-user component is disabled. The figures show the comparison of each case with the complete model's performance. The significance of each component for deriving the final output varies from one data set to another. However, it is consistent with all data sets that the most substantial components, in this case, were user-user and user-item connections, which show the importance of trust network and the users. The item-user component is closely following the previous two components in terms of significance, which means that the model heavily relies on this aspect of the network when making predictions. This result also explains that the algorithms

that only consider user-user and user-item connections can perform relatively well. Although the item-item component shows a smaller change in the performance when it is disabled, the results still demonstrate the effectiveness of this component.

The changes in the performance of the components were relatively marginal, meaning that there is some overlap between the information that each component provides. The difference in performance was comparatively lower in the RMSE results, possibly due to the RMSE dampening smaller error values. This difference is most evident in the item-item results for Douban compared to the final HeteroGraphRec result.

The removal of the item-item component essentially converts the HeteroGraphRec model into GraphRec. Therefore, we can infer that even though item-item connections provide the least amount of information than the other three components. However, including the item-item connections can still boost the performance of the network and contribute to the final output of the model.

### Node Feature Initialization

Proper weight initialization is one of the determining factors in the convergence rate of the model. In this research, we compared two different initialization methods to view their impact on the convergence rate of the HeteroGraphRec model. The first initialization method is the random initialization from a Gaussian distribution with mean 0 and variance 1. The second method pre-trains the model using Node2Vec [8], to obtain embedding vectors for users and items, then performs the training using the acquired features as input vectors to the HeteroGraphRec model.

The performance of the above-noted initialization methods can be measured through the hit ratio in the top-K list [70] [71]. Hit ratio is the number of correct predictions (hits) in the top-K recommendations, and this value is calculated as follows:

$$HR@K = \frac{\text{Number of Hits@K}}{|GT|} \tag{6.1}$$

Where, the denominator $|GT|$ is the number of all test sets, and the numerator is the sum of the number of test sets present in each user's top-K list.

Figure 6.6 shows the result of the top-k recommendations for k=10 throughout the training process. The results confirm that using pre-trained feature vectors as initializations for the model can considerably improve the model's convergence rate. Among the datasets, Ciao had the highest accuracy for the top-10 recommendations, followed by Douban and then Epinions. These results are consistent with the RMSE and MAE evaluations of the model.

## 6.2 Balanced Random Walks

In this section, we provide the details of our datasets, then we evaluate the walking strategy, compare the performance of the whole model against other state-of-the-art algorithms, and evaluate the parameter sensitivity of the proposal BalNode2Vec. The experimental results are repeated for 10 random seed initialization and are statistically significant with a p value of less than 1%.

### 6.2.1 Dataset

We use five different data sets in this research, which 4 of them are publicly available. One of the data sets is generated from an academic and social networking site – CourseNetworking [49]. The user-follower network of the CourseNetworking is extracted from this academic and social networking site to illustrate the performance of the BalNode2Vec. There are 61151 nodes and 313923 edges. The nodes of this graph represent users, and an edge between a source and target node represents the follower and followee, respectively. The distribution of the edges closely follows the pattern of Zipf degree distribution, as shown in Figure 5.1. Therefore, the network graph is scale-free so that the BalNode2Vec model can be used to learn the representations of this network. This network consists of nodes with different "roles" like student or

Table 6.4.
Area Under Curve (AUC) scores for link prediction using the Average binary operator.

| **Algorithm** | **CourseNetworking** | Reddit Hyperlink | PPI | arXiv | Facebook |
|---|---|---|---|---|---|
| **BalNode2Vec** | **0.7480** | **0.7644** | **0.7097** | **0.7157** | **0.7570** |
| **DeepWalk** | 0.7324 | 0.7488 | 0.6647 | 0.6853 | 0.7360 |
| **Walklet** | 0.7127 | 0.7503 | 0.6451 | 0.6740 | 0.7470 |
| **LINE** | 0.7054 | 0.7096 | 0.6429 | 0.6843 | 0.7217 |
| **Node2Vec** | 0.7431 | 0.7422 | 0.6772 | 0.6932 | 0.7443 |
| **GraRep** | 0.7453 | - | 0.6819 | 0.7049 | 0.7518 |

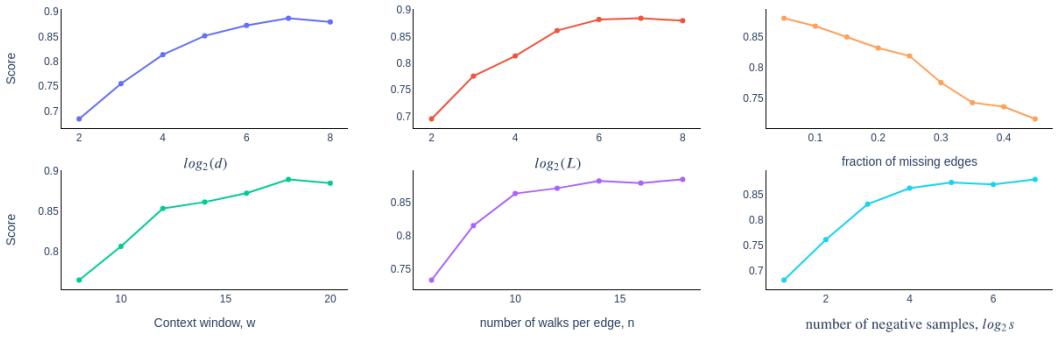


Fig. 6.7. Area Under Curve (AUC) scores for link prediction on the CourseNetworking dataset.

teacher, which means that a large number of source and destination nodes exist in this network.

The other data sets used in this research include the following:

Table 6.5.
Area Under Curve (AUC) scores for link prediction using the Hadamard binary operator.

| **Algorithm** | **CourseNetworking** | Reddit Hyperlink | PPI | arXiv | Facebook |
|---|---|---|---|---|---|
| **BalNode2Vec** | **0.8837** | **0.9792** | **0.7796** | **0.9541** | **0.971** |
| **DeepWalk** | 0.8452 | 0.9530 | 0.7481 | 0.9354 | 0.9591 |
| **Walklet** | 0.8430 | 0.9618 | 0.7129 | 0.9110 | 0.9480 |
| **LINE** | 0.8247 | 0.9436 | 0.7244 | 0.887 | 0.9323 |
| **Node2Vec** | 0.8539 | 0.9627 | 0.7563 | 0.9374 | 0.9620 |
| **GraRep** | 0.8651 | - | 0.7580 | 0.9418 | 0.9567 |

- Reddit Hyperlink [72]: The hyperlink network represents the directed connections between two subreddits (a subreddit is a community on Reddit). The network is extracted from publicly available Reddit data of 2.5 years from Jan 2014 to April 2017. There are 55,863 nodes and 858,490 edges.

- Protein-Protein Interactions (PPI) [73]: In the PPI network for Homo Sapiens, nodes represent proteins, and an edge indicates a biological interaction between a pair of proteins. The network has 19,706 nodes and 390,633 edges.

- arXiv ASTRO-PH [72]: This is a collaboration network generated from papers submitted to the e-print arXiv where nodes represent scientists, and an edge is present between two scientists if they have collaborated in a paper. The network has 18,722 nodes and 198,110 edges.

Table 6.6.
Area Under Curve (AUC) scores for link prediction using the Weighted-L1 binary operator.

| **Algorithm** | **CourseNetworking** | Reddit Hyperlink | PPI | arXiv | Facebook |
|---|---|---|---|---|---|
| **BalNode2Vec** | **0.8793** | **0.9754** | **0.7701** | **0.9463** | **0.9676** |
| **DeepWalk** | 0.8312 | 0.9499 | 0.7455 | 0.9334 | 0.9517 |
| **Walklet** | 0.8368 | 0.9670 | 0.7114 | 0.9048 | 0.9483 |
| **LINE** | 0.8219 | 0.9428 | 0.7211 | 0.8947 | 0.9315 |
| **Node2Vec** | 0.8547 | 0.9684 | 0.7647 | 0.9352 | 0.9579 |
| **GraRep** | 0.8697 | - | 0.7671 | 0.9388 | 0.9497 |

- Facebook [72]: In the Facebook network, nodes represent users, and edges represent a friendship relation between any two users. The network has 4,039 nodes and 88,234 edges.

### 6.2.2 Walking Strategy

Figure 5.2 compares the node distribution by using the random walk and a balanced walk. As shown from the results, the balanced walk reduces the sampling error and generates samples that follow the power-law more closely and, as a result, produces a sequence that is more suitable for applying the Skip Gram algorithm.

To numerically measure the difference between the node frequencies in a walk, and the degree distribution, which is the target distribution, we used the statistical distance methods. The Jensen-Shannon divergence is used in this evaluation. The results of figure 5.3 show that the predictions have successfully managed to balance

Table 6.7.
Area Under Curve (AUC) scores for link prediction using the Weighted-L2 binary operator.

| **Algorithm** | **CourseNetworking** | Reddit Hyperlink | PPI | arXiv | Facebook |
|---|---|---|---|---|---|
| **BalNode2Vec** | **0.8724** | **0.9620** | **0.7514** | **0.9481** | **0.9648** |
| **DeepWalk** | 0.8349 | 0.9476 | 0.7428 | 0.9294 | 0.9516 |
| **Walklet** | 0.8407 | 0.9596 | 0.7150 | 0.9127 | 0.9429 |
| **LINE** | 0.8258 | 0.9428 | 0.7213 | 0.8920 | 0.9317 |
| **Node2Vec** | 0.8511 | 0.9644 | 0.7428 | 0.9341 | 0.9606 |
| **GraRep** | 0.8687 | - | 0.7601 | 0.9447 | 0.9542 |

the node representations during the walk. Because of the initial condition of choosing nodes based on their degree, the BalNode2Vec has a minimal error in very short walks and even in longer walks, it manages reduce the rate of divergence significantly, thus it allows for a longer walk length in order to capture deeper network connections.

### 6.2.3 Link Prediction

To evaluate the link predictions, typically, we use the output embedding to predict the future edges in the graph [8]. This experiment is carried out by selecting 50% of the edges at random, with the constraint that the residual graph must be connected, and then performing the training on the residual graph and predicting the removed links. The output node embeddings are then converted to edge embeddings using binary operators mentioned in 5.1. The edge embeddings are compared with the removed edges of the graph to measure a model's performance.

In this research, we compare the proposed BalNode2Vec against the most recent state-of-the-art algorithms DeepWalk [6], Node2Vec [8], LINE [7], GraRep [74], and Walklet [36]. Tables 6.4 to 6.7 show the Area Under the Curve (AUC) score for each binary operator mentioned in Table 5.1. We could not measure the performance of GraRep on the Reddit Hyperlink dataset due to out of memory error. This is because GraRep is not scalable on graphs with near one million edges. Based on the results, BalNode2Vec performs better than all the other methods on every data set. The performance improves more significantly when dealing with networks that have less reciprocal connections like CourseNetworking and Reddit Hyperlink datasets. When we look at operators individually (Table 5.1), BalNode2Vec outperforms the others in all different operators. Node2Vec works better than the other methods except on Reddit Hyperlink involving the Average Binary operator in which DeepWalk performs better than Node2Vec. When Hadamard operator is used, the performance of the BalNode2Vec shows more improvement than the other methods across all datasets.

### 6.2.4 Parameter Sensitivity

The proposed BalNode2Vec has several parameters related to the sampling strategy and network training that need to be fine-tuned for each network. With the same procedure for link prediction, Figure 6.7 shows the sensitivity of the model to various parameters. All the parameters are set to default value except the one being tested. The default values for this experiment were: walk per edge: 12, embedding dimension $d$=128, context window $w$=18 and walk length $l$=64.

We measure the AUC score against the number of features $d$, the walk length $l$, and context window size $w$ and how they affect the performance. We observe that performance tends to saturate once the dimensions of the representations reach around 100. Similarly, we observe that increasing the number and length of walks improves performance. The context window size, $w$, also improves performance and training depth at the cost of increased training time. However, the performance differences

are not that large when $w$ is close to 20. The number of negative samples also has a diminishing effect on the accuracy of the model, and after some point (around 32), it is not computationally efficient to increase the number of negative samples.

### 6.2.5 Embedding Visualization

The final embeddings generated for the CourseNetworking dataset are generated using this method and then clustered using the K-means clustering technique with K=6. In order to visualize the nodes and their respective cluster, one of the dimensionality reduction techniques need to be used to bring the embedding dimensionality back to 2. The reason the training is not done with d=2 is that our goal is to evaluate the performance of the model in the production scale, which has a high embedding size, and training with a lower dimension undermines our ability to visualize the actual performance of the model. Two dimensionality reduction techniques were considered for the visualization, t-SNE and PCA.

Principal component analysis (PCA) [75] simplifies the complexity in high-dimensional data while retaining trends and patterns. It does this by transforming the data into fewer dimensions, which act as summaries of features. High-dimensional data are very common in biology and arise when multiple features, such as expression of many genes, are measured for each sample. This type of data presents several challenges that PCA mitigates: computational expense and an increased error rate due to multiple test correction when testing each feature for association with an outcome. PCA is an unsupervised learning method and is similar to clustering1—it finds patterns without reference to prior knowledge about whether the samples come from different treatment groups or have phenotypic differences.

t-SNE [76] is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different

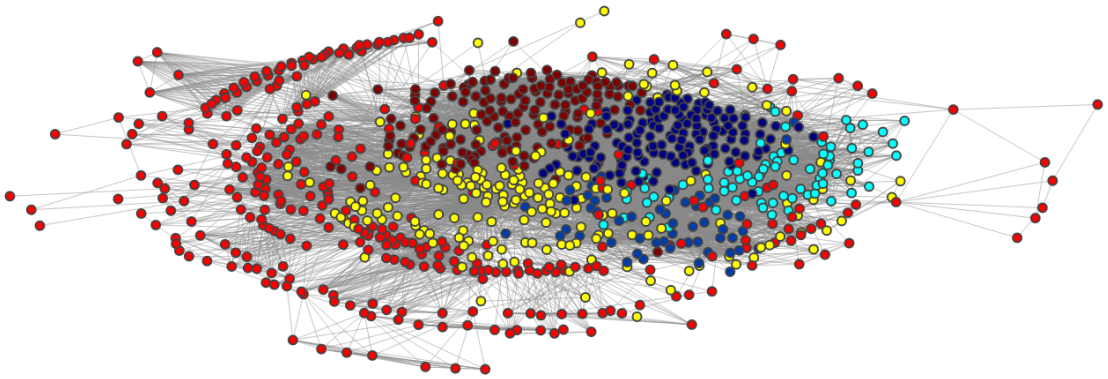

Fig. 6.8. Top 500 most connected users in the CourseNetworking dataset visualized by applying t-SNE over the Balanced Walk Embeddings and colored using K-means clustering.

initializations we can get different results. For discrete probability distributions $P$ and $Q$ defined on the same probability space, $X$, the Kullback–Leibler divergence from $Q$ to $P$ is defined [77] to be:

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right) \tag{6.2}$$

which is equivalent to:

$$D_{\mathrm{KL}}(P\|Q) = -\sum_{x\in\mathcal{X}} P(x)\log\left(\frac{Q(x)}{P(x)}\right) \tag{6.3}$$

t-SNE differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance. We chose t-SNE over PCA specifically because of the non-linearity which inherently exists in the embeddings, additionally, we also used MahalaNobis distance [78] as our metric for measuring the difference between embeddings. The result of the visualization for top 500 most connected users in CourseNetworking can be seen in Figure 6.8.

# 7. CONCLUSION AND FUTURE WORK

In this research, we proposed and evaluated a novel graph-based recommender system, HeteroGraphRec, which uses modern neural network structures to intelligently aggregate both the item-related and user-related information in a social network. This new model of the social network naturally contains item–item connections based on similarity or similar metrics. These connections contain information about the structure of the items in the network. Then, we proposed a neural network architecture that utilizes the various connections and uses the structured input to improve the recommender system's performance. We illustrated the effectiveness of the HeteroGraphRec model by comparing them with previous baselines in graph recommenders. Our HeteroGraphRec model consistently outperforms all the baselines, meaning that the additional item–item dimension is valuable in providing more accurate recommendations.

We also provide a scalable, adaptable, and efficient node embedding method - BalNode2Vec. We investigate the limitations of the random walk sampling strategy for sequence generation for the Skip Gram model and propose the BalNode2Vec model, which is capable of devising a model and improving the performance of the baseline Node2Vec model. By adopting a new sampling direction, BalNode2Vec can analyze the graph structure and modify the path accordingly, hence, ensuring that all the nodes are adequately sampled. We compare the proposed BalNode2Vec model against five different state-of-the-art models on five different data sets. The results demonstrate the BalNode2Vec can provide more consistent and accurate results and performs better than the existing models.

Currently, our model aggregates all the local information related to a particular node in the graph, without constructing the global parameters related to the network. The global parameters can be used to analyze the most popular or the most

well-received items in the entire network, which could help provide accurate recommendations. In the future, we plan to incorporate global parameters in aggregators to improve our model's prediction accuracy further. Another future direction to enhance our model is to consider the dynamic behavior of the social network by intelligently incorporating temporal data into our model.

We also plan to use the embeddings generated using this technique to improve the recommendation engine and provide offer community and role-based results based on these embeddings. As for the algorithm, in the future, our effort will be to make the optimizer and sampler more interactive so that the graph sampling will be based on the optimizer's result, and the sampler would sample what the optimizer needs instead of performing random walks.

# REFERENCES

# REFERENCES


[1] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. IEEE International Joint Conference on Neural Networks, pp. 729734, 2005.

[2] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. IEEE Transactions on Neural Networks, 20(1):61–80, 2009.

[3] T. Kipf and M. Welling. "Semi-supervised classification with graph convolutional networks." ICLR 2017

[4] P. Velickovi c, G, Cucurull, A, Casanova, A, Romero, P, Liò, and Y, Bengio. 2018. Graph Attention Networks. ICLR (2018)

[5] X. Wang, H. Ji, C. Shi, B. Wang, C. Peng, Y. P., and Y. Ye, "Heterogeneous graph attention network," in WWW, 2019.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in KDD, 2015.

[7] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: Large-scale Information Network Embedding. In WWW, 2015.

[8] A. Grover and J. Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In KDD '16. ACM, 855–864.

[9] Y. Goldberg and O. Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. CoRR abs/1402.3722 (2014).

[10] J. Tang et al. Recommendation with social dimensions. 30th AAAI Conference on Artificial Intelligence, AAAI 2016. AAAI press, 2016. p. 251-257. Research output: Chapter in Book/Report/Conference proceeding Conference contribution

[11] H. Ma, D. Zhou, C. Liu, M. R Lyu, and I. King. 2011. Recommender systems with social regularization. In Proceedings of the fourth ACM international conference on Web Search and Data Mining. ACM, 287–296.

[12] J. Tang, X. Hu, H. Gao, and H. Liu. 2013. Exploiting local and global social context for recommendation. In IJCAI, Vol. 13. 2712–2718.

[13] P. V Marsden and N. E Friedkin. 1993. Network studies of social influence. Sociological Methods Research 22, 1 (1993), 127–151.

[14] M. McPherson, L. Smith-Lovin, and J. M Cook. 2001. Birds of a feather: Homophily in social networks. Annual review of sociology 27, 1 (2001), 415–444

[15] G. Guo, et al. "Etaf: An extended trust antecedents framework for trust prediction." Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. IEEE Press, 2014.C

[16] H. Ma, H. Yang, M. R Lyu, and I. King. 2008. Sorec: social recommendation using probabilistic matrix factorization. In Proceedings of the 17th ACM conference on Information and Knowledge Management. ACM, 931–940.

[17] B. Yang, Y. Lei, J. Liu, and W. Li. 2017. Social collaborative filtering by trust. IEEE transactions on pattern analysis and machine intelligence 39, 8 (2017), 1633–1647.

[18] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," arXiv preprint arXiv:1812.04202, 2018, [online] Last Date Accessed: 2020-07-08.

[19] S. Cao, W. Lu, and Q. Xu. Deep neural networks for learning graph representations. In AAAI, 2016.

[20] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. Trans. Neur. Netw., 8(3):714–735, May 1997. ISSN 1045-9227. doi: 10.1109/72.572108, [online] Last Date Accessed: 2020-07-08.

[21] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. IEEE transactions on Neural Networks, 9(5):768–786, 1998.

[22] Z. Zhao, Q. Yang, H. Lu, T. Weninger, D. Cai, X. He, and Y. Zhuang. 2018. Social-aware movie recommendation via multimodal network learning. IEEE Transactions on Multimedia 20, 2 (2018), 430–440.

[23] X. Wang, X. He, L. Nie, and T. Chua. 2017. Item silk road: Recommending items from information domains to social users. In Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 185–194.

[24] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu. 2017. On deep learning for trust-aware recommendations in social networks. IEEE transactions on neural networks and learning systems 28, 5 (2017), 1164–1177.

[25] W. Fan, Q. Li, and M. Cheng. 2018. Deep Modeling of Social Relations for Recommendation. In AAAI.

[26] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin. 2007. Graph embedding and extensions: A general framework for dimensionality reduction. IEEE TPAMI 29, 1 (2007)

[27] P. Goyal, E. Ferrara, Graph Embedding Techniques, Applications, and Performance: A Survey (2018), Knowledge-Based Systems.

[28] W. L. Hamilton et al. 2017. Representation Learning on Graphs: Methods and Applications. IEEE Data Engineering Bulletin on Graph Systems.

[29] O. Barkan and N. Koenigstein, "Item2vec:Neural Item Embedding for Collaborative Filtering." Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[30] J. Yang and J. Leskovec. Overlapping communities explain core-periphery organization of networks. Proceedings of the IEEE, 102(12):1892–1902, 2014.

[31] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. RolX: structural role extraction  mining in large graphs. In KDD, 2012.

[32] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang. A deep learning approach to link prediction in dynamic networks. In ICDM, 2014.

[33] Y. Dong, J. Zhang, J. Tang, N. V. Chawla, and B. Wang. 2015. CoupledLP: Link Prediction in Coupled Networks. In KDD '15. ACM, 199–208.

[34] D. L. Nowell and J. Kleinberg. 2003. The link prediction problem for social networks. In Proceedings of the twelfth international conference on Information and knowledge management (CIKM '03). ACM, New York, NY, USA, 556-559. DOI: https://doi.org/10.1145/956863.956972, [online] Last Date Accessed: 2020-07-08.

[35] Y. Dong, Yuxiao  N. V. Chawla  A. Swami(2017). metapath2vec: Scalable Representation Learning for Heterogeneous Networks. 135-144. 10.1145/3097983.3098036.

[36] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena. Don't walk, skip!: Online learning of multi-scale network embeddings. In Advances in Social Networks Analysis and Mining (ASONAM), 2017.

[37] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In KDD '18. ACM, 974–983.

[38] F. Monti, M. Bronstein, and X. Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In Advances in Neural Information Processing Systems. 3700–3710

[39] R. v d Berg, T. N Kipf, and M. Welling. 2017. Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263 (2017), [online] Last Date Accessed: 2020-07-08.

[40] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Proc. of NIPS, 2017, pp. 1024–1034.

[41] M. Li, K. Tei, and Y. Fukazawa, "An efficient co-attention neural network for social recommendation," in Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI), October. 2019, pp. 34–42.

[42] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin. Graph neural networks for social recommendation. In WWW, 2019.

[43] T. Bui, S. Ravi and V. Ramavajjala. "Neural Graph Learning: Training Neural Networks Using Graphs." WSDM 2018

[44] T. Bansal, D. Juan, S. Ravi and A. McCallum. "A2N: Attending to Neighbors for Knowledge Graph Inference." ACL 2019

[45] I. Goodfellow, J. Shlens and C. Szegedy. “Explaining and harnessing adversarial examples.” ICLR 2015

[46] T. Miyato, S. Maeda, M. Koyama and S. Ishii. “Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning.” ICLR 2016

[47] D. Juan, C. Lu, Z. Li, F. Peng, A. Timofeev, Y. Chen, Y. Gao, T. Duerig, A. Tomkins and S. Ravi “Graph-RISE: Graph-Regularized Image Semantic Embedding.” WSDM 2020

[48] Google Brain Team, “Neural Structured Learning: training with structured signals”, https://www.tensorflow.org/neural_structured_learning. [online] Last Date Accessed: 2020-07-08.

[49] A. Jafari, “CourseNetworking, a global, academic social-networking site with unique, next-generation technology solutions for learning and collaboration for universities”, thecn.com, Cyberlab.iupui.edu, [online], Last Date Accessed: 2020-07-04.

[50] IUPUI CyberLab, “Rumi, an intelligent online personal learning assistant”, rumi.thecn.com, [online], Last Date Accessed: 2020-07-04.

[51] “Coursenetworking, white paper”, https://www.thecn.com/aboutus, 2012, [online] Last Date Accessed: 2019-02-20.

[52] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, Bert: Pretraining of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018, [online] Last Date Accessed: 2020-07-08.

[53] K. Banker, “MongoDB in Action”, Manning Publications Co. Greenwich, CT, USA ©2011, ISBN:1935182870 9781935182870

[54] https://docs.mongodb.com/manual/replication/,[online] Last Date Accessed: 2020-011-15.

[55] S. Gilbert and N. Lynch, “Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services”, ACM SIGACT News, Volume 33 Issue 2 (2002), pg. 51–59. doi:10.1145/564585.564601, [online] Last Date Accessed: 2020-07-08.

[56] M. Shertil, “traditional RDBMS to NOSQL database: new era of databases for big data” (2016).

[57] A. Lakshman, “Cassandra: a decentralized structured storage system”, ACM SIGOPS Operating Systems Review Volume 44 Issue 2, April 2010, Pages 35-40

[58] https://cassandra.apache.org/doc/latest/architecture/index.html,[online] Last Date Accessed: 2020-011-15.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017, [online] Last Date Accessed: 2020-07-08.

[60] https://www.librec.net/datasets.html, [online] Last Date Accessed: 2020-03-28.

[61] https://sites.google.com/site/erhengzhong/datasets, [online] Last Date Accessed: 2020-03-28.

[62] http://www.trustlet.org/downloaded_epinions.html, [online] Last Date Accessed: 2020-03-28.

[63] K. Li, J. Gao, S. Guo, N. Du, X. Li, and A. Zhang. LRBM: A restricted boltzmann machine based approach for representation learning on linked data. In ICDM, 2014

[64] E. Zhong, W. Fan, and Q. Yang. "User behavior learning and transfer in composite social networks." ACM Transactions on Knowledge Discovery from Data (TKDD) 8.1 (2014): 6

[65] Y. Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 426–434

[66] R. Salakhutdinov and A. Mnih. 2007. Probabilistic Matrix Factorization. In 21th Conference on Neural Information Processing Systems, Vol. 1. 2–1.

[67] M. J. and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the fourth ACM conference on Recommender systems. ACM, 135–142.

[68] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua. 2017. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web, WWW. 173–182.

[69] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, 2010

[70] M. Ji, J. Han, and M. Danilevsky. 2011. Ranking-based classification of heterogeneous information networks. In KDD '11. ACM, 1298–1306.

[71] J. Zhang, J. Tang, C. Ma, H. Tong, Y. Jing, and J. Li. 2015. Panther: Fast top-k similarity search on large networks. In KDD '15. ACM, 1445–1454.

[72] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014, [online], Last Date Accessed: 2020-07-01.

[73] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. 2013. B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood, et al. The BioGRID interaction database. Nucleic acids research, 36:D637–D640, 2008.

[74] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In KDD, 2015.

[75] K. Pearson (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space". Philosophical Magazine. 2 (11): 559–572. doi:10.1080/14786440109462720, [online] Last Date Accessed: 2020-07-08.

[76] L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(November):2579-2605, 2008.

[77] D. J.C. MacKay (2003). Information Theory, Inference, and Learning Algorithms (First ed.). Cambridge University Press. p. 34. ISBN 9780521642989.

[78] P. C. Mahalanobis, "On the Generalized Distance in Statistics," Proceedings of National Institute of Sciences (India), Vol. 2, No. 1, 1936, pp. 49-55.