TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

THESIS NO: 075MSCSK010

POI Recommendations with the Use of Knowledge Graph Neural Networks

by
Nabin Paudyal

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SYSTEM AND KNOWLEDGE ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
LALITPUR, NEPAL

August, 2021

POI Recommendations with the Use of Knowledge Graph Neural Networks

by

Nabin Paudyal

075MSCSK010

Thesis Supervisor

Assoc. Prof. Dr. Arun Kumar Timalsina

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer System and Knowledge Engineering

Department of Electronics and Computer Engineering

Institute of Engineering, Pulchowk Campus

Tribhuvan University

Lalitpur, Nepal

August, 2021

COPYRIGHT©

DECLARATION

I declare that the work hereby submitted for Master of Science in Computer System and Knowledge Engineering (MSCSKE) at IOE, Pulchowk Campus entitled **"POI Recommendations with the Use of Knowledge Graph Neural Networks"** is my own work and has not been previously submitted by me at any university for any academic award.

I authorize IOE, Pulchowk Campus to lend this thesis to other institution or individuals for the purpose of scholarly research.

Nabin Paudyal
075MSCSK010
Date: August, 2021

RECOMMENDATION

The undersigned certify that they have read and recommended to the Department of Electronics and Computer Engineering for acceptance, a thesis entitled **"POI Recommendations with the Use of Knowledge Graph Neural Networks"**, submitted by **Nabin Paudyal** in partial fulfillment of the requirement for the award of the degree of **"Master of Science in Computer System and Knowledge Engineering"**.

.....

Supervisor:
Assoc. Prof. Dr. Arun Kumar Timalsina,
Department of Electronics and Computer Engineering,
Institute of Engineering, Tribhuvan University

.....

External Examiner: Kumar Pudashine,
Senior Section Chief, Network and Security,
Agricultural Development Bank Ltd

.....

Committee Chairperson:
Assoc. Prof. Dr. Nanda Bikram Adhikari,
Department of Electronics and Computer Engineering,
Institute of Engineering, Tribhuvan University

Date: August, 2021

DEPARTMENTAL ACCEPTANCE

The thesis entitled **"POI Recommendations with the Use of Knowledge Graph Neural Networks"**, submitted by **Nabin Paudyal** in partial fulfillment of the requirement for the award of the degree of **"Master of Science in Computer System and Knowledge Engineering"** has been accepted as a bonafide record of work independently carried out by him in the department.

.....

Prof. Dr. Ram Krishna Maharjan
Head of the Department
Department of Electronics and Computer Engineering
Pulchowk Campus
Institute of Engineering,
Tribhuvan University,
Nepal.

ACKNOWLEDGEMENT

First and foremost, I want to express my appreciation to the Department of Electronics and Computer Engineering, IOE, Pulchowk Campus, for allowing me to work on the thesis as part of our Masters of Science in Computer System and Knowledge Engineering (MSCSKE) program.

I'd like to express my heartfelt gratitude to my thesis supervisor, **Assoc. Prof. Dr. Arun Kumar Timalsina**, for his invaluable guidance, timely feedback and continuous encouragement during the course of this thesis work. I appreciate his direction and assistance throughout the thesis, as well as during the period we collaborated on the project prior to thesis.

I am also grateful to **Assoc. Prof. Dr. Nanda Bikram Adhikari** sir, our program coordinator, for his general assistance throughout the thesis lifetime, beginning with proposal submission all the way to final defense.

I'd also like to extend my sincerest of gratitudes towards external examiner **Kumar Pudashine** sir for his invaluable suggestions and feedback which has helped a lot in improving the quality of this thesis work.

I'm also grateful to my classmates for their continuous encouragement and helpful suggestions over the course of the thesis.

ABSTRACT

Recommendation of the most relevant travel attractions for travellers when they are visiting a new place is a very important problem. The ability to recommend the most relevant tourist attractions that would be of most interest to the visitors can help increase the profits for the business and also provide a better traveler experience on the visitor's part. Some of the most common techniques used for recommending travel attractions are collaborative filtering based ones which rely on the information such as places visited by groups of other travelers that share maximum similarity to the visitor. For the purpose of this thesis work, the goal was to devise a method to generate the most relevant travel attraction recommendations for users drawing upon other models that produced excellent results in recommendation for other items like movies and books in the past. In that regard, Knowledge Graphs were combined with Graph Neural Networks in the ensuing KGNN. Knowledge Graphs represent the relationship between entities in the form of a graph while Graph Neural Networks are the form of Neural Networks that apply over data structured as graphs. The experiments were carried out using the three different KGNN approaches - KGCN, KGNN-LS and KGAT, on the publicly available location based social network check-in datasets for Foursquare and Gowalla. The performance improvement of 24.19%, 14.51% and 22.58% were achieved respectively for the three methods for top 5 recommendations on Foursquare in terms of F1-score when compared to the best performing baseline of Rank-GeoFM. Similarly, the improvements were 13.20%, 14.86% and 28.30% and 18.27%, 25.58% and 30.23% respectively for top 10 and 20 recommendations for Foursquare. The performance improvement on Gowalla meanwhile was of 19.35%, 29.03% and 43.54% respectively for top 5 recommendations, 10.29%, 129.41% and 29.41% for top 10 recommendations and 6.77%, 15.25% and 25.42% respectively for top 20 recommendations.

Keywords: Recommender Systems, Knowledge Graphs, Travel Attractions Recommendation, Graph Neural Neworks

TABLE OF CONTENTS

COPYRIGHT iii

DECLARATION iv

RECOMMENDATION v

DEPARTMENTAL ACCEPTANCE vi

ACKNOWLEDGEMENT vii

ABSTRACT viii

TABLE OF CONTENTS ix

LIST OF FIGURES xii

LIST OF TABLES xiv

LIST OF ABBREVIATIONS xv

1 INTRODUCTION 1

1.1 Background and Motivation 1

1.2 Problem Definition 4

1.3 Objectives 4

1.4 Scope of the Work 4

1.5 Originality of this Work 5

1.6 Organisation of thesis work 5

2 LITERATURE REVIEW 6

3 METHODOLOGY 12

3.1 Machine Learning on Graphs 12

3.1.1 Graphs 12

3.1.2 Knowledge Graphs (KG) 13

3.1.3 Graph Neural Networks (GNN) 14

3.2 Problem Formulation 21
3.3 Knowledge Graph Convolutional Networks (KGCN) 22
3.4 Knowledge aware Graph Neural Networks with Label Smoothness Regularization (KGNN-LS) 23
3.5 Knowledge Graph Attention Network (KGAT) 24
3.6 System Architecture 25
3.7 Learning Algorithm 27
4 EXPERIMENTAL SETUP 29
4.1 Dataset 29
4.1.1 Foursquare 29
4.1.2 Gowalla 29
4.2 Environment and Tools 30
4.2.1 Google's Knowledge Graph Search API 30
4.2.2 Pytorch 30
4.2.3 Pytorch Geometric 30
4.2.4 Matplotlib 30
4.2.5 Networkx 31
4.3 Baselines 31
4.3.1 PMF 31
4.3.2 GeoMF 32
4.3.3 Rank-GeoFM 32
4.4 Parameters 32
4.4.1 Neighbour sampling size (K) 33
4.4.2 Embedding dimension (d) 33
4.4.3 Depth of receptive field (H) 33

4.4.4 Aggregation function (σ) . . . 33
4.5 AUC Results . . . 34
4.5.1 AUC Results for KGCN . . . 34
4.5.2 AUC Results for KGNN-LS . . . 35
4.5.3 AUC Results for KGAT . . . 36
5 RESULTS **38**
5.1 Loss Plot . . . 38
5.1.1 Foursquare . . . 39
5.1.2 Gowalla . . . 40
5.2 Visualizations . . . 42
5.3 Top 5 Results . . . 45
5.4 Top 10 Results . . . 46
5.5 Top 20 Results . . . 47
6 EVALUATION **51**
6.1 Evaluation Metrics . . . 51
6.2 Results of Evaluation . . . 52
6.2.1 Results on Foursquare Dataset . . . 53
6.2.2 Results on Gowalla Dataset . . . 54
7 CONCLUSION AND FUTURE WORK **57**
7.1 Conclusion . . . 57
7.2 Future Work . . . 58
REFERENCES **62**
APPENDIX A **63**

LIST OF FIGURES

1.1 An example of a typical knowledge graph [1] 3

3.1 An example of a typical graph [2] 13

3.2 An example of a travel knowledge graph [3] 14

3.3 Message passing in GNNs [4] 16

3.4 Architecture of Graph Convolutional Networks (GCN) [5] 19

3.5 Attention Passing on Graph Attention Networks (GAT) [6] 21

3.6 The Framework for KGCN [7] 23

3.7 Overview of KGNN-LS model 24

3.8 Overview of KGAT model [8] 25

3.9 Architecture to learn KG Embeddings [9] 26

3.10 Architecture to learn User-POI Embeddings [10] 26

5.1 Loss Plot for KGCN (Foursquare) 39

5.2 Loss Plot for KGNN-LS (Foursquare) 39

5.3 Loss Plot for KGAT (Foursquare) 40

5.4 Loss Plot for KGCN (Gowalla) 40

5.5 Loss Plot for KGNN-LS (Gowalla) 41

5.6 Loss Plot for KGAT (Gowalla) 41

5.7 Visualization for User checkins 42

5.8 Initial POI Embeddings 43

5.9 Final POI Embeddings 44

5.10 Top 5 Recommendations for KGCN 45

5.11 Top 5 Recommendations for KGNN-LS 45

5.12 Top 5 Recommendations for KGAT 45
5.13 Top 10 Recommendations for KGCN 46
5.14 Top 10 Recommendations for KGNN-LS 46
5.15 Top 10 Recommendations for KGAT 47
5.16 Top 20 Recommendations for KGCN 48
5.17 Top 20 Recommendations for KGNN-LS 49
5.18 Top 20 Recommendations for KGAT 50

6.1 Precision on Foursquare dataset 53
6.2 Recall on Foursquare dataset 54
6.3 F1-score on Foursquare dataset 55
6.4 Precision on Gowalla dataset 55
6.5 Recall on Gowalla dataset 56
6.6 F1-score on Gowalla dataset 56

LIST OF TABLES

4.1 Foursquare metrics 29

4.2 Gowalla metrics 30

4.3 AUC Results for different K for KGCN 34

4.4 AUC Results for different d for KGCN 34

4.5 AUC Results for different H for KGCN 34

4.6 AUC Results for different σ for KGCN 35

4.7 AUC Results for different K for KGNN-LS 35

4.8 AUC Results for different d for KGNN-LS 35

4.9 AUC Results for different H for KGNN-LS 36

4.10 AUC Results for different σ for KGNN-LS 36

4.11 AUC Results for different K for KGAT 36

4.12 AUC Results for different d for KGAT 37

4.13 AUC Results for different H for KGAT 37

4.14 AUC Results for different σ for KGAT 37

LIST OF ABBREVIATIONS

API	Application Programming Interface
CNN	Convolutional Neural Network
GeoMF	Geographical Matrix Factorization
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GDL	Graph Deep Learning
GNN	Graph Neural Network
KGNN-LS	Knowledge aware Graph Neural Networks with Label Smoothness Regularization
KG	Knowledge Graph
KGAT	Knowledge Graph Attention Network
KGCN	Knowledge Graph Convolutional Network
LSBN	Location Based Social Network
POI	Point of Interest
PMF	Probabilistic Matrix Factorization
Rank-GeoFM	Ranking based Geographical Factorization Method
RNN	Recurrent Neutal Network

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Today, the internet is the chief source to find ideas for vacation and purchase appropriate travel packages. Selecting places to visit is a frequently encountered issue by users. From the viewpoint of a traveler, "What to do in [destination]" is one of the most frequently searched travel related question. Before deciding the place/s to visit, a significant amount of time and work is required. Meanwhile, finding the best places of visit for travelers is a big source of business for travel service providers. Expedia and TripAdvisor, two major travel and tourism websites, provide visitors with information on tourist attractions which is their core business. Some of the most critical challenges for travel business owners are what to market to end customers and how to sell to them [11].

With the rise of platforms like Youtube, Amazon, and Netflix over the last few decades, recommender systems have become more relevant in our lives day by day. Recommender systems have become practically omnipresent in our lives, from e-commerce platforms that indicate things that would interest customers to online advertisements that advise the relevant contents to users based on their tastes. Recommender systems, to put it simply, are algorithms that propose relevant goods to consumers. Depending on the use case, the things could be anything from movies to books to products to anything else.

In some industries, recommender systems are critical because their effective implementations have the ability to produce significant revenue and give businesses a competitive advantage. Consider the fact that Netflix conducted a contest a few years ago with the purpose of building a recommender system that was superior to its own algorithm, and the winner received a prize of $1 million USD. [12]

The ultimate goal of any recommendation system is to predict a user's interest in a new product based on the user's past preferences, personalized needs, and the product's specific properties and traits, in order to recommend the most appropriate product to use for the user, increase the user's level of satisfaction, and enable the user to make efficient decisions. The key utility of a recommender system is its capacity to generate the most appropriate decision for users without requiring them to express their requirements explicitly.

In today's world of big data, the use of traditional recommendation systems to handle data mining-related problems is quite limited. The use of knowledge graphs, based on the assumption of huge data, is a considerably more efficient method for creating recommender systems.

A knowledge graph is essentially a directed labeled graph with clearly defined labels. Nodes, edges and labels make up a directed labeled graph. People, companies, departments, books, cars, and other entities can all operate as nodes. An edge connects two nodes, and captures the relationship of interest between the nodes. Friendship between two individuals, network connection between two computers, and employee-employer relationship between a firm and an employee are all instances of such connections. The labels capture the meaning of the relationship. A friendship between two persons is an example of this type of relationship.

A knowledge graph can be thought of as a model reflecting an area of knowledge that has been created by subject matter experts using various machine techniques. It gives all of the data a clear structure and a generic interface, as well as the ability to construct smart and multilateral relationships between all of our databases. The knowledge graph sits on top of our existing popular databases and connects all of our available data – both structured and unstructured – at scale. Knowledge graphs are organized as a virtual data layer on top of the main data layer.

Multiple databases may be required in practice to generate knowledge graphs in

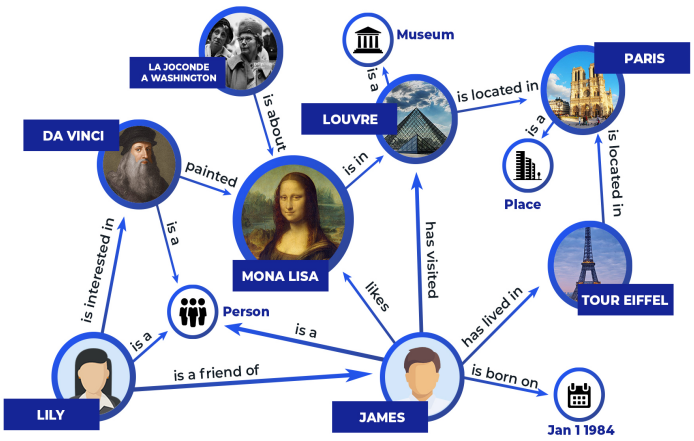

Figure 1.1: An example of a typical knowledge graph [1]

the industry. The building and management of knowledge graphs can be secured by the use of components such as ontology editors and taxonomies, graph mappers, validation, entity extractors, visualization and search tools, and so on. Despite the fact that graph databases are normally managed by data engineers in collaboration with highly qualified semantic web professionals, the Semantic Middleware interfaces assist individuals in interacting with the knowledge graph. As a result, those with less technical skills, in addition to business and knowledge professionals, can profit from the system's utilization [3].

Links inside the graph can be used to uncover the relationship between objects and users as a path, which can give rich and relevant information for interactions between people and items. Such connectivity is capable of not only revealing the semantics of relations and entities, but also of understanding a user's interests. The use of knowledge graphs to improve recommender system performance [13] is an active area of research. Several hybrid systems were developed in the past to increase performance by combining content-based and collaborative filtering methods in a blended approach. Recently, research has emphasized on the use of external knowledge graphs (KGs) for recommendations in order to improve the suggestive capabilities.

1.2 Problem Definition

The problem dealt with in this thesis work is the problem of POI recommendation. In this thesis work, POI recommendation has been treated as a supervised machine learning problem. As with any supervised machine learning problem, real valued output is required for inputs in the past observed data. This information in this case is available in the form of user's checkin to specific POIs. Based on the information which is available for a user's preference to POIs in the past, the goal is to learn the preference for POIs in the future. The POIs might be the ones the user interacted with in the past or entirely new ones as well. The user's preference for POIs is available as their checkin information on publicly available location based social network datasets. With this information available, the goal is to learn the probability of checkin to any other POIs for the user in the future.

1.3 Objectives

The objectives of this thesis work are:

1. To combine Knowledge Graphs (KG) and Graph Neural Networks (GNN) to alleviate data sparsity problems and devise a better approach for the next POI recommendation problem.

2. To evaluate the performance of two GCN based methods: Knowledge Graph Convolutional Networks (KGCN) and Knowledge aware Graph Neural Networks with Label Smoothness Regularization (KGNN-LS) and GAT based approach: Knowledge Graph Attention Network (KGAT) for the next POI recommendation problem.

1.4 Scope of the Work

Choosing the best places to visit is a common problem faced by travelers while creating an itinerary best suited to the interest of the traveler and recommending the places in which he/she is most interested is a very common problem for travel

business providers. In this thesis work, the goal is to find the most optimal method to recommend POIs in which the travelers would be most interested in based upon the checkin information available for the traveler while visiting other places in the past. The methods used are KG aware and GNN based which learn the user and POI embeddings based on user's interests for POIs with specific relationships. The datasets used for the thesis work are based on the two popular location based social networks - Foursquare and Gowalla.

1.5 Originality of this Work

Knowledge Graph aware methods have been widely used in the recommendation of other items such as movies, music and books to users with exciting results. End-to-end graph based machine learning has also been gaining wide popularity as an effective tool to solve the problem of recommendation for users. This thesis is the first work that tries to use KG aware graph machine learning to solve the problem of POI recommendation.

1.6 Organisation of thesis work

The thesis is divided into seven different sections. A brief introduction along with problem definition, as well as the motivation behind carrying out research in this sector, is discussed in chapter one. The next chapter explores the past efforts in the field of POI recommendation and also in domain of recommendation in general. The overall architecture of the system along with discussion around the fundamental principles of the underlying technology is discussed in the third section i.e. the methodology section. The fourth chapter discusses the experimental setup covering the different tools and datasets used. The fifth chapter presents the different sets of results obtained from our experiments. In the sixth chapter, evaluation is performed by comparing our methodology with existing baseline methods. In the seventh and final chapter, conclusions drawn from this thesis work along with some avenues to carry out further work in this domain are presented.

CHAPTER 2

LITERATURE REVIEW

In 2012, Google presented the idea of knowledge graphs [14]. The purpose of knowledge graphs is to define different concepts and entities that exist in the physical world, as well as to represent the relationships that exist between them. In a knowledge graph, each idea and entity has a universally unique ID, the intrinsic properties of the entity are represented by each of the attribute-value pairs, and the two entities are connected and their relationship is specified by a label.

Knowledge graphs were quickly adopted by prominent researchers in the domain of recommendation systems in a variety of fields, yielding impressive results. Oramas et al. [15] used the knowledge graph to make recommendations in the field of sound and music. Lu [16] et al. built a knowledge tree consisting of international tourism attractions using data from DBpedia, Geonames, and Wikidata in order to recommend tourist attractions. Noia et al. [17] used a knowledge graph to generate book recommendations.

The primary idea behind using knowledge graphs to address recommendation problems is to identify the features included in a knowledge graph as quickly as possible. Network representation learning has lately been a major area of study in the field of machine learning. Learning a low-dimensional representation for each network node while keeping the original structural information is the goal of network representation learning. This method has proven to be a very successful means of learning the features of the graph.

Handmade feature vectors and learned representations, both types of vector representations for graphs and relational structures, enable the use of traditional data analysis tools as well as machine learning approaches for the structures. Various ways for constructing embeddings have been researched in the disciplines of machine

learning and knowledge representation [18]. Vector embeddings, on the other hand, have gained virtually little theoretical interest. As a result of its capacity to bridge the gap between the "discrete" world of relational data and the "differentiable" world of machine learning, vector embeddings offer a lot of research promise. Apart from the binary relations in knowledge graphs, very little work has been done on relational data embeddings.

The majority of present research focuses on the suggestion of point of interests (POIs) through the usage of social networks based on location. This area has seen a lot of research [19]. Existing works have used variety of data types, including contextual data, social information, classifications, and tags. Using social information essentially entails making use of information about places frequented by a user's friends. It has been shown that there is a relatively minimal overlap of destinations among friends, and that leveraging this social information to improve the performance of suggestions is of little use.

Contextual data is useful in the same way as spatial information because people tend to visit sites that are close to each other [20]. Because many people visit several locations at different times and frequently visit the same locations within the same time period, temporal information can also be useful. The utilization of various context-based data, such as weather and travel speed, has also been extensively studied [21].

Tags and categories can be used to collect semantic data about POIs as well. By mining tags and categories connected to POIs, the authors of [22] address the usage of an aggregated latent Dirichlet allocation model to learn about the topics that users are interested in. It should be observed, however, that in the vast majority of situations, the tags are either absent, inaccurate, or of little use for the intended purpose. "Me and Ann, Easter 2012 trip to Europe" [23] is one such example.

The most popular dataset in the Linked Open Data cloud is DBPedia. DBPedia

contains a wealth of information on tourism attractions. On the other hand, extracting a travel knowledge graph is a challenging operation. To begin with, the DBpedia ontology lacks an ontology class that encompasses all of the cities that exist, albeit there is the dbo:City class. However, a large number of cities lack the requisite data for this category. dbr:Paris is an excellent example of this. On the other hand, there is also lack of an ontology class that has information of all of the tourist attractions. Even the most common types of travel attractions, such as dbo: Museum, dbo:Park, and dbo:Monument, are dispersed throughout the ontology and are not instances of a larger class like "Travel attraction". Another issue is that the aforementioned classes could not be identified as a tourist attraction. For example, dbr:Parc Montsouris, one of Paris' most well-known parks, lacks the type information dbo:Park as well as any information about its subclasses.

Due to the rise of social networks that rely on users' geolocation, recommendation for POIs has emerged as a major research challenge. However, due to a lack of checkin data, a form of data for implicit feedback, there is a significant problem among popular methodologies that have been adopted for POI suggestion. Li, Xutao, and colleagues presented Rank-GeoFM, a geographical factorization approach based on ranking, for the purpose of suggesting POIs in [24], which addresses these two issues. POIs with and without checkins both played a role in learning the ranking, and so the problem of data sparsity was addressed. Furthermore, their model was able to incorporate different contextual information, such as temporal and geographical influence. A stochastic gradient descent based method was used for learning. Experiments on public datasets were conducted extensively to validate the suggested technique for user-POI and user-time-POI settings. In both scenarios, the experimental findings showed that the proposed strategy outperformed exisiting state-of-the-art methods.

Knowledge graphs in recommender system applications have a lot of potential for improving explainability and accuracy. However, the majority of commonly used methods consider the complete knowledge network. Because graphs are rarely complete in real-world environments, and often miss entities, facts, and relations,

this can lead to poor performance. As a result, it's vital to account for incomplete graphs when utilizing recommender systems. In [25], Yixin, Cao, and colleagues suggested a hybrid technique for knowledge graph completion and recommendation. They provided information about graph relations in order to comprehend why a user would love them, unlike previous graph-based recommendation algorithms. The authors' technical contribution was a new translation-based recommendation model that took various user preferences into account when translating a user to an item, and then jointly trained the model with the completion model by mixing different types of transfer schemes. Their solution surpassed existing graph-based state-of-the-art recommendation systems in extensive testing on two publicly available benchmark datasets.

In [26], Song et. al, explored the use of recommender systems along with knowledge graphs, for the purpose of effectively addressing the problems of cold start and data sparsity. The authors explored generation of recommendations by discovering most important paths between users and items. To be precise, the problem was formulated as a decision process in sequence, where the initial state is the user, and the actions are the walks on the level of graphs. The rewards were then shaped using the prevalent state-of-the-art techniques and the policy gradient methods were used to train the policy function. By performing experiments on publicly available datasets, the authors showed that their method was not only able to provide better recommendations but also offer reasonable explanations.

In [27], Zhang et al. investigated the use of heterogeneous knowledge base information to improve recommender systems. By utilizing the knowledge base, three different components were built to extract semantic representations for things from structural content, textual content and visual content, respectively. TransR, a heterogeneous approach of network embedding, was used to extract structural representations of items while accounting for the heterogeneity of nodes as well as relationships. To acquire textual and visual representations for the items, two separate deep learning-based embedding algorithms, stacked denoising auto-encoders in combination with stacked convolutional auto-encoders, were used. Finally, the

Collaborative Knowledge Base Embedding (CKE) integrated framework was utilized to build latent collaborative filtering representations as well as semantic item representations from the knowledge base. The authors' strategy outperformed numerous widely accepted state-of-the-art methods for suggestion.

To address the problem of cold start and data sparsity in CF-based recommender systems, researchers typically collect attributes for users and objects and create complicated algorithms to make the best use of this side information. The qualities are generally linked to one another, establishing a knowledge network (KG). Hongwei et al. introduced Knowledge Graph Convolutional Networks (KGCN) in their research [7], a framework for efficiently capturing relatedness between items by understanding their relationships on the KG in an end-to-end way. To identify high-order structural and logical information in the KG, neighbourhood sampling was carried out for each KG entity as the receptive field. The neighborhood information was then combined together with bias while computing the entity's representation. The model was used to generate recommendations for three separate datasets of music, books, and movies, and the findings revealed that their approach outperformed existing state-of-the-art baselines in recommendation.

The power of knowledge graphs lies in their inherent ability in capturing structural information and relatedness between entities because of which they are a powerful source of side information for improving recommendations. However, a major issue with popular strategies is that they rely on explicit feature engineering and aren't suitable for end-to-end training. Hongwei et al. proposed using Knowledge-aware Graph Neural Networks with Label Smoothness regularization (KGNN-LS) to deliver optimal suggestions in their paper [28]. Their method computed item embeddings for specific users first by using a trainable function that could capture the most essential linkages in the user's knowledge network. After that, the graph was converted into a weighted graph for a particular user and then a GNN was used in order to compute item embeddings with personalization. To incorporate better bias for induction, their method used the label smoothness assumption which assumes that two things in the KG that appear together are substantially

more likely to have equal user relevance scores than items that do not appear together. In cold-start scenarios with minimal user-item interactions, the technique was found to provide outstanding outcomes.

Moving beyond the trivial simulation of user-object interactions, taking contextual knowledge into account is critical for more precise, diversified, and explicable suggestions. Traditional techniques, such as the factorization machine (FM), represent this as a supervised learning problem, assuming that each interaction is a single instance with stored side information. Because they don't evaluate the relationship between things, these strategies aren't very good at handling collaboration between diverse users. In their paper [8], Xiang et al. investigated the use of a knowledge graph (KG), which combines items and attributes to break down the interaction assumption independently. In a hybrid KG and user-item graph structure, higher-order relations — those that link things to one or more attributes — play a key role in recommendation. They proposed the Knowledge Graph Attention Network (KGAT), a technique that can describe high-order relationships in a KG in an end-to-end manner.

CHAPTER 3

METHODOLOGY

3.1 Machine Learning on Graphs

Machine learning on graphs has recently gained a lot of traction in a variety of fields, including social networks, knowledge graphs, recommender systems, and even life science. The ability of graphs to model the dependencies between nodes in a graph allows for a breakthrough in graph analysis research.

End-to-end deep learning paradigms like CNN, RNN, and autoencoders have given fresh life to machine learning applications including object detection, machine translation, and speech recognition. Deep Learning is effective at detecting hidden patterns in Euclidean data (images, text, videos).

However, applications based on data collected from non-Euclidean domains, which are represented as graphs with intricate interactions and interdependencies between items, have yet to be fully explored. That's where graph-based machine learning comes in. Graph Deep Learning (GDL) is a new field of research with a lot of promise for learning and interpreting graph data.

3.1.1 Graphs

The graph is the fundamental part of a GNN. A graph is a data structure that is made up of two kinds of elements: nodes and edges. G = (V, E), where V represents the set of nodes and E represents the edges between them, is the formal definition of a graph G. The edges are directed when there are directional dependencies between nodes. They are undirected if such directional relationships do not exist.

A graph is usually associated with an adjacency matrix A. A is a square matrix

that represents connectivity between nodes in a network. The dimension of A is (n x n) for a network with n nodes. In some cases, a feature set, such as a user profile, is used to represent the nodes. The feature matrix X for each node in the graph has a dimension of (n x f) when each node in the graph is represented by f features.

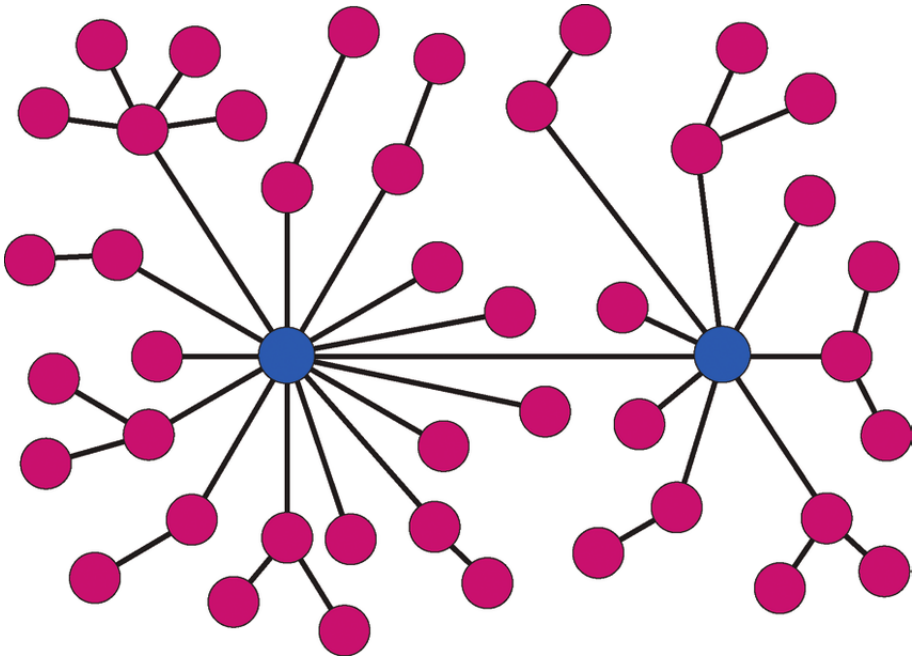

Figure 3.1: An example of a typical graph [2]

3.1.2 Knowledge Graphs (KG)

A knowledge graph is composed of triples of entity-relation-entity (h,r,t). The head and tail are defined by h and t, respectively, while the relationship between the head and tail is represented by r. The link between the things represented in the network is described by a knowledge graph. A triple *(Big Ben, location.location.city, London)*, for example, illustrates the relationship that *Big Ben* is one of the places in *London*.

In this thesis work, the travel knowledge graph has been constructed using Google's Knowledge Graph Search API. We make use of the following relationships for a POI in the KG.

1. containsPlace

2. address

3. aggregateRating

4. type

5. isAccessibleForFree

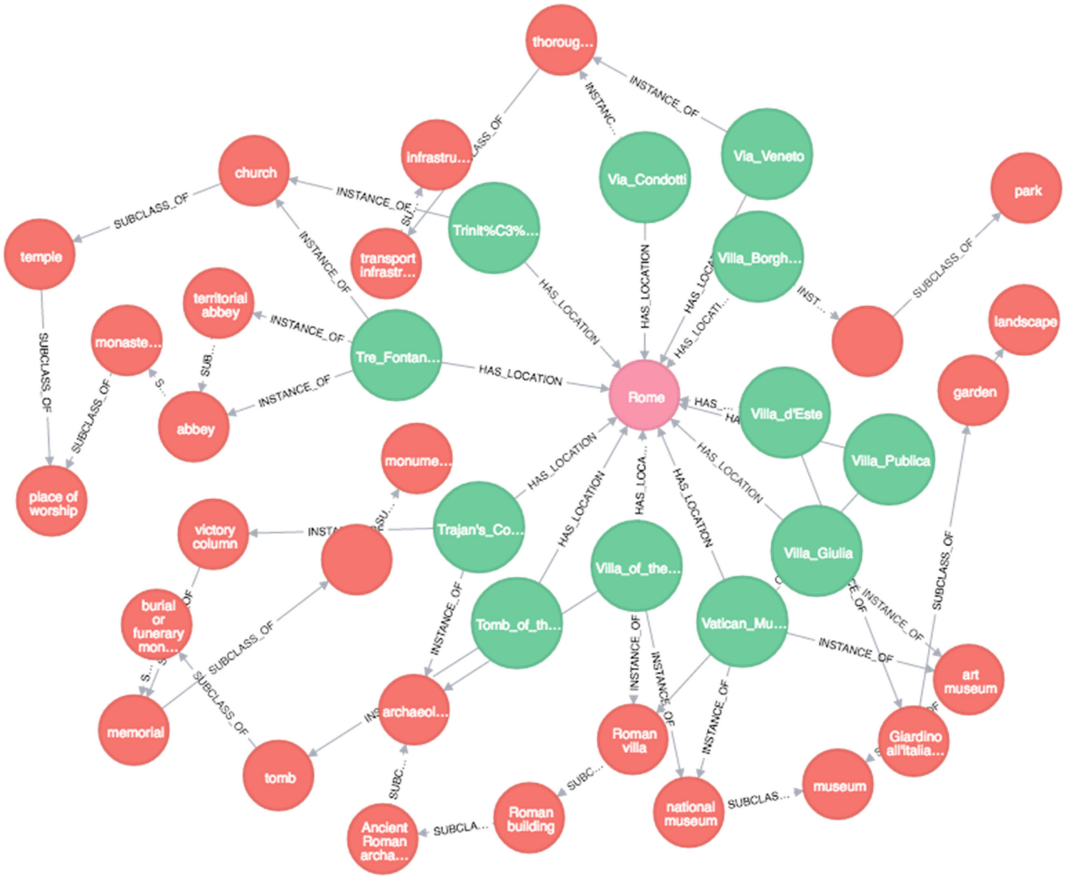

Figure 3.2: An example of a travel knowledge graph [3]

3.1.3 Graph Neural Networks (GNN)

A Graph Neural Network is a form of Neural Network that works with the graph structure directly. Node classification is a typical use of GNN. Every node in the graph has a label, and our goal is to predict node labels lacking ground-truth data.

Each node v in the node classification issue is specified by its feature x_v and connected with a ground-truth label t_v. The purpose for graph G, which is partially

labeled, is to use the nodes with labels to forecast the nodes without labels. Each node is indicated by a d-dimensional h_v vector containing neighborhood information, as learned by the learning method. Specifically,

$$h_v = f(x_v, x_{co[v]}, h_{ne[v]}, x_{ne[v]}) \tag{3.1}$$

where $x_{co[v]}$ denotes the features of the edges that are connected with v, $h_{ne[v]}$ denotes the embedding of the nodes that form the neighbourhood of v, and $x_{ne[v]}$ denotes the neighbouring node features for v. The transition function f is used to project these inputs onto a d-dimensional space. The Banach fixed point theorem can be used to write the preceding equation as an iteratively update process because h_v requires a unique solution. Message forwarding or neighborhood aggregation are terms used to describe this activity.

$$H^{t+1} = F(H^t, X) \tag{3.2}$$

H and X stand for the aggregation of all the h and x, respectively.

To calculate the output of the GNN, the state h_v as well as the feature x_v is passed to an output function g.

$$o_v = g(h_v, x_v) \tag{3.3}$$

Both f and g are fully connected feed-forward neural networks.

The loss L1 can be formulated as:

$$loss = \sum_{i=1} (t_i - o_i) \tag{3.4}$$

which can be optimized via gradient descent.

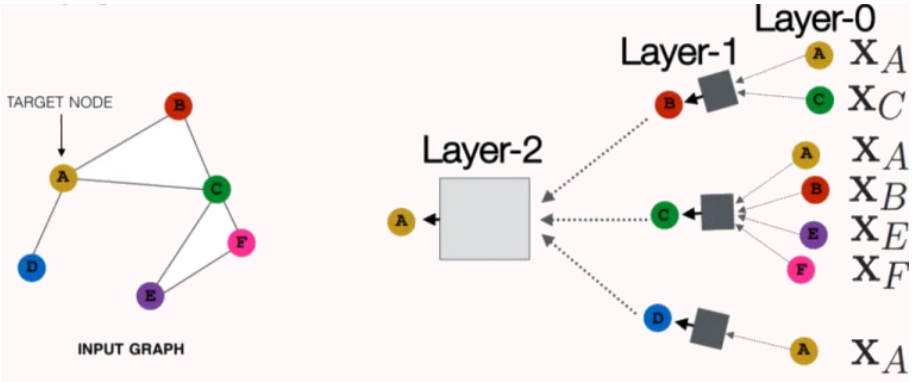

Figure 3.3: Message passing in GNNs [4]

3.1.3.1 Graph Convolutional Networks (GCN)

Images can be interpreted as graphs of pixels with connection to other pixels, but they maintain a fixed structure at all times. A Convolutional Neural Network (CNN) shares weights across neighbours based upon certain assumptions. For example, a 3 x 3 area of pixels can be evaluated as a "neighborhood". The assumptions upon which the CNNs are based depend on 2-dimensional data of regular nature, which is also referred to as Euclidean data.

But social media networks, molecular structure representations, or map addresses aren't of two-dimensional nature. They neither have a fixed size nor any structure. There are several issues while cramming non-Euclidean data into CNN's, because their usefulness ends just right about there.

In effect, the main problem involved in the embedding of features represented as vertices and edges is the problem of arbitrary usage of space, and an absence of any Euclidean distance separating the neighbors. Because of these factors, the problem needs to be approached with different sets of assumptions.

The convolution operation in neural networks is a way of sharing weights among neighbors. First, in order to determine the neighbors, some data needs to be provided.

In a typical neural network architecture, the evaluation of weights determines the feature representation for the next hidden layer using the forward propagation and feature representation of the current layer along with its bias. However, in the context of graph convolutional networks, the adjacency matrix also comes along in the equation. A non-linear activation is used as well, similar to classical neural network models.

The majority of graph neural network models in use today follow a standard and widely understood architecture. The name given to these models is Graph Convolutional Networks (GCNs); convolutional because the filter parameters are generally shared across all sites in the graph.

The purpose of these models is to learn a feature function on a graph G = (V, E) that accepts the following as input:

- Every node I has a feature description x_i, which can be represented as an N x D feature matrix X. (The number of nodes is N, while the total number of input characteristics is D.)
- The graph structure is represented as a matrix, commonly in the form of an adjacency matrix A.

and producing a node-level output Z (an N x F feature matrix, where F is the total number of output features for each node). With the addition of some form of pooling mechanism, the outputs at the graph level may then be represented.

After that, each layer of the neural network may be expressed as a nonlinear function.

$$H^{l+1} = f(H^l, A) \tag{3.5}$$

with H^0 =X and H^L = Z (or z for graph-level outputs), and L the number of layers. The only difference between the models is how f is chosen and specified.

As an example, consider the following simple layer-wise propagation rule:

$$f(H^l, A) = \sigma(A(H^l, W^l)) \tag{3.6}$$

where W^l is a weight matrix for the l-th neural network layer and σ is a non-linear activation function similar to ReLU.

However, there are two major flaws in this straightforward model: For each node, multiplication with A means adding all the feature vectors of all neighboring nodes together, but not the node itself (as long as there are no self-loops in the graph). This can be solved by ensuring that the graph contains no self-loops, which can be done by simply adding the identity matrix to A.

The second big disadvantage is that A is rarely normalized, therefore multiplying with it completely affects the scale of the feature vectors. The solution to this problem is to normalize A so that all of the rows add up to one, i.e. to use $D^{-1}A$, where D is the diagonal node degree matrix. Taking the average of surrounding node features now equates to multiplying with $D^{-1}A$.

In practice, symmetric normalization, i.e. $D^{-1/2}AD^{-1/2}$, makes dynamics more interesting (because it's no longer just averaging neighboring nodes). In their study, Kipf & Welling [29] developed the following propagation rule:

$$f(H^l, A) = \sigma(\bar{D}^{-1/2}\bar{A}\bar{D}^{-1/2}H^l W^l) \tag{3.7}$$

with $\bar{A} = A + I$, where $\bar{D}$ refers to A's diagonal node degree matrix, and I is the identity matrix.

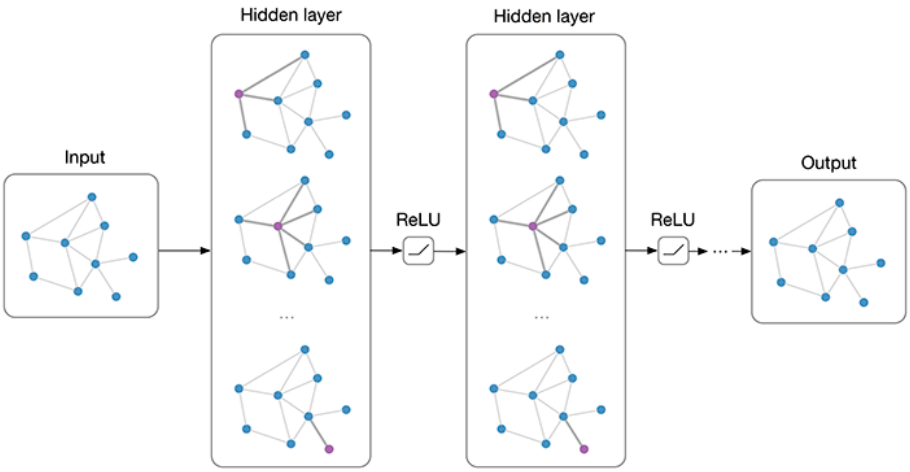

Figure 3.4: Architecture of Graph Convolutional Networks (GCN) [5]

3.1.3.2 Graph Attention Networks (GAT)

Let's look at a network with n nodes and a set of features for the node ($\vec{h}_1$, $\vec{h}_2$, ..., $\vec{h}_n$), along with an adjacency matrix A, where $A_{ij} = 1$ if i and j are linked; else, 0. A graph convolutional layer computes a set of new node features ($\vec{h}'_1$, $\vec{h}'_2$, ..., $\vec{h}'_n$) based on the input attributes as well as the graph topology.

A weight matrix W specifies a shared node-wise feature transformation for each graph convolutional layer (such that a higher-level representation can be created). The vectors $\vec{g}_i$ are normally recombined in some fashion at each node after that, resulting in the vectors $\vec{g}_i = \mathrm{W}\vec{h}_i$ as a result.

A graph convolutional operator can be considered as an aggregate of characteristics across neighbourhoods in general to meet the localisation property. The attributes of node i's output are as follows: Given the neighborhood of node i is called N_i (usually consists of all of i's first-order neighbors, including i),

$$\vec{h}_i = \sigma(\sum_{j \varepsilon N_i} \alpha_{ij} \vec{g}_j) \tag{3.8}$$

where α_{ij} is the weighting factor (importance) of node j's attributes to node i and σ is an activation function.

The GAT layer improves on the GCN layer's basic aggregation function by using attention coefficients to assign different priority to each edge.

$$z_i^l = W^l h_i^l \tag{3.9}$$

The lower layer embedding h_i is transformed linearly in the preceding equation, and W is the learnable weight matrix. This transformation aids in the transformation of input features into high-level and dense features by providing appropriate expressive power.

$$e_{ij}^l = LeakyReLU((\vec{a}^l)^T(z_i^l||z_j^l)) \tag{3.10}$$

A pair-wise unnormalized attention score between two neighbors is computed using the aforementioned technique. It initially concatenates the two nodes' z embeddings, where || signifies concatenation. The dot product of such concatenation and a learnable weight vector is then computed. Finally, a LeakyReLU is applied to the dot product result.

$$\alpha_{ij}^l = \frac{exp(e_{ij}^l)}{\sum_{k\varepsilon N_i} exp(e_{ik}^l)} \tag{3.11}$$

The attention scores on each node's incoming edges must then be normalized using a softmax function. In a probability distribution, the softmax encodes the outcome of the previous step. As a result, the attention scores across different nodes are much more comparable.

$$h_i^{l+1} = \sigma(\sum_{j\varepsilon N_i} \alpha_{ij}^l z_j^l) \tag{3.12}$$

The GCN aggregation, in which neighbor embeddings are aggregated together and scaled by attention scores, is comparable to the preceding equation. The fundamental result of this scaling process is that each neighbor node learns a different contribution.

To let α_{ij} be implicitly specified, self-attention is applied over the node features. This is prompted by the fact that, as the Transformer architecture has demonstrated, self-attention is sufficient for state-of-the-art machine translation outcomes.

In general, α_{ij} is generated as a consequence of the attentional mechanism a:$R^N \times R^N \rightarrow$R, which calculates unnormalized coefficients e_{ij} across node pairs i,j taking into account their characteristics:

$$e_{ij} = (\vec{h}_i, \vec{h}_j) \tag{3.13}$$

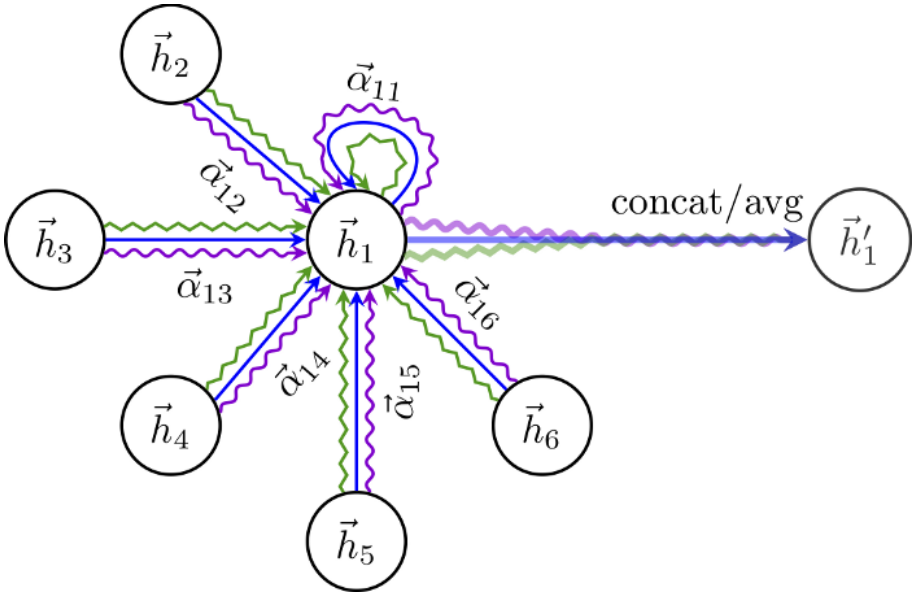

Figure 3.5: Attention Passing on Graph Attention Networks (GAT) [6]

3.2 Problem Formulation

In this thesis, the goal is to tackle the issue of a knowledge-graph-aware POI recommendation problem. A set of M users represented by U = $\{u_1, u_2, ..., u_M\}$ and a set of N POIs represented by V = $\{v_1, v_2, ..., v_N\}$ are provided in a standard POI recommendation situation. A user-POI interaction matrix $Y \varepsilon R^{MxN}$ is also supplied, which is defined by user's feedback, check-in information in the context of our work. Given a knowledge graph G = (h, r, t), where h ε E, r ε R and r

ε E, where E denotes the collection of entities and R denotes the collection of relationships, the goal is to learn $\bar{y}_{uv}$ = F(u, v : Y, G), where $\bar{y}_{uv}$ indicates the likelihood that user u will be interested in POI v.

3.3 Knowledge Graph Convolutional Networks (KGCN)

KGCN is a mechanism to capture high-order structural relatedness between entities in a knowledge graph. A potential pair of user u and POI (entity) v is given. The set of entities directly connected to v is denoted by N (v), while the relation between entities e_i and e_j is denoted by r_{e_i,e_j}. To calculate the score between a user and a relation, the function g : R^d x R^d→R (e.g., inner product) is utilized:

$$\pi_r^u = g(u, r) \tag{3.14}$$

where d is the dimension of representations, while u ε R^d and r ε R^d are the representations of user u and relation r, respectively. In general, u_r denotes the relevance of the relationship r to the user u. One visitor might be more interested in the free POIs, while another would be more concerned with the POI's "category."

To characterize the topological closeness structure of item v, we compute the linear combination of v's neighborhood:

$$v_{N(v)}^u = \sum_{e\varepsilon N(v)} \bar{\pi}_{r_{v,e}e}^u \tag{3.15}$$

where $\bar{\pi}_r^u$ is the user-relation score normalized.

$$\bar{\pi}_{r_{v,e}}^u = \frac{exp(\pi_{r_{v,e}}^u)}{\sum_{e\varepsilon N(v)} exp(\pi_{r_{v,e}}^u)} \tag{3.16}$$

and e is the entity e's representation. User-relation scores act as personalized filters when computing an entity's neighborhood representation since we aggregate the neighbors with bias with respect to these user-specific scores.

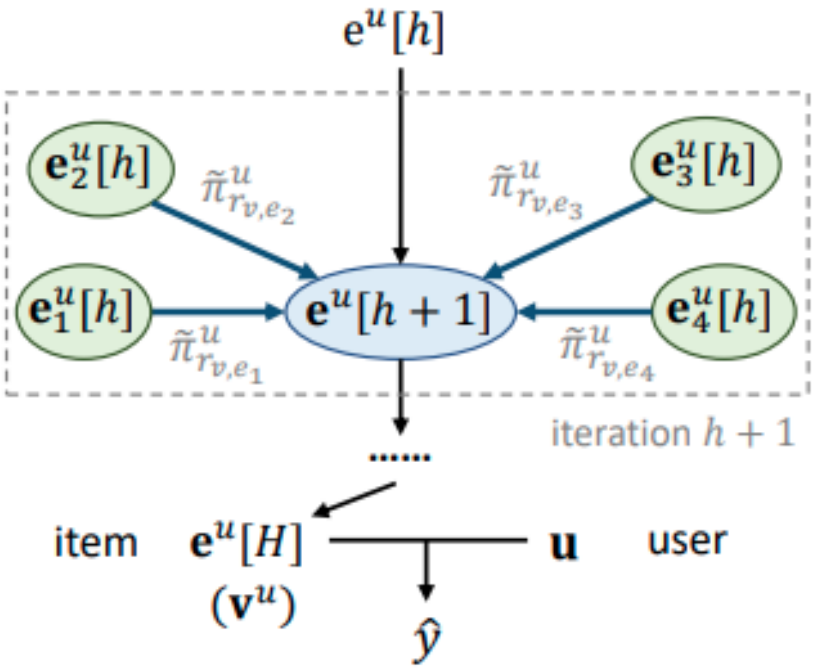

Figure 3.6: The Framework for KGCN [7]

3.4 Knowledge aware Graph Neural Networks with Label Smoothness Regularization (KGNN-LS)

In traditional GNNs, the edge weights of the input graph are fixed, but in KGNN-LS (which contains possible function g parameters and feature vectors of users and relations), they are learnable and require supervised training like W. Because user-item interactions are the primary source of supervised signal outside of GNN layers, enhancing the model's fitting ability will ultimately lead to overfitting. In addition, edge weights are important in graph representation learning. As a result, more edge weight regularization is needed to aid in the learning of entity representations and to generalize to unseen interactions more successfully.

Let's consider what an ideal set of edge weights might be. Consider a function of labels with a real value l_u : E $\rightarrow$ R on G, which is compelled to take a particular value $l_{u(v)} = y_{uv}$ at node v ε V $\subseteq$ E. In our context, $l_{u(v)} = 1$ if the user u considers the item v interesting and interacts with it, otherwise $l_{u(v)} = 0$. The label smoothness assumption states that we should expect close items in the KG to

have comparable relevance labels. This is why the energy function E was chosen:

$$E(l_u, A_u) = \frac{1}{2} \sum_{e_i \varepsilon E, e_j \varepsilon E} A_u^{ij} (l_u(e_i) - l_u(e_j))^2 \tag{3.17}$$

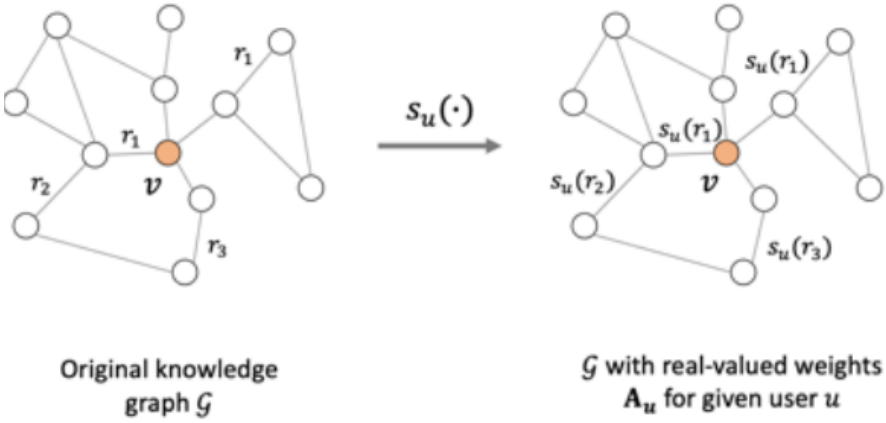

Figure 3.7: Overview of KGNN-LS model

3.5 Knowledge Graph Attention Network (KGAT)

Entities and relations can be parameterized as vector representations while the graph structure is preserved using knowledge graph embedding. TransR is used on CKG in KGAT. To be precise, each entity is embedded and relation is optimized by the principle of translation i.e. $e_h^r + e_r \approx e_t^r$, if a triplet (h, r , t) exists in the graph. Herein, e_h, e_t ε R_d and e_r ε R^k are the embeddings for h, t, and r respectively; and e_h^r, e_t^r are the representations that are projected for e_h and e_t in the relation r's space. As a result, for a given triplet (h, r, t), the plausibility score (or energy score) is calculated as follows:

$$g(h, r, t) = ||W_r e_h + e_r - W_r e_t||_2^2 \tag{3.18}$$

where W_r ε R^{kxd} is the relation r matrix for transformation, that projects entities from the d-dimensional entity space into the k-dimensional space of relations. g(h, r, t) with a lower score indicates that the triplet has a higher chance of being correct, and vice versa.

(h, r, t) is implemented using a mechanism of relational attention, which can be defined in the following manner:

$$\pi(h, r, t) = (W_r e_t)^T tanh(W_r e_h + e_r) \tag{3.19}$$

where tanh is the nonlinear activation function we've chosen. As a result, the distance between e_h and e_t in the relation r's space influences the attention score, with closer entities propagating more information.

Following that, we use the softmax function to normalize the coefficients across all triplets related to h:

$$\pi(h, r, t) = \frac{exp(\pi(h, r, t)}{\sum_{(h,r',t')\varepsilon N_h} exp(\pi(h, r', t'))} \tag{3.20}$$

As a result, the final attention score can be used to determine which neighbor nodes should be given more attention to capture collaborative signals. The attention flow shows which areas of the data to focus on when propagation forward is employed, which can be thought of as arguments for the recommendation.

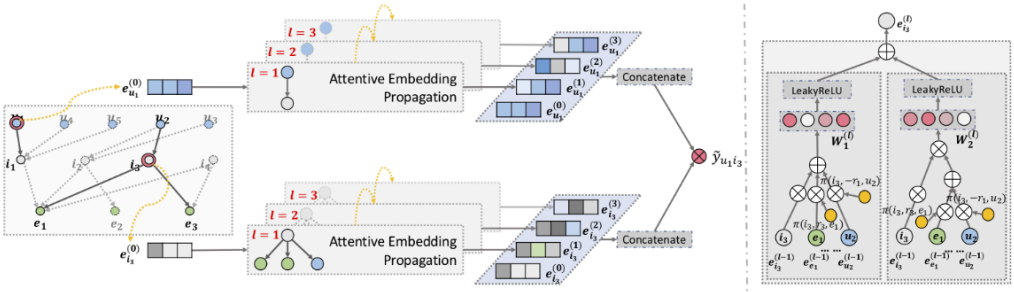

Figure 3.8: Overview of KGAT model [8]

3.6 System Architecture

The goal of this thesis work is to combine user-POI interaction matrix with knowledge graph constructed for the POIs to learn user and POI embeddings that are

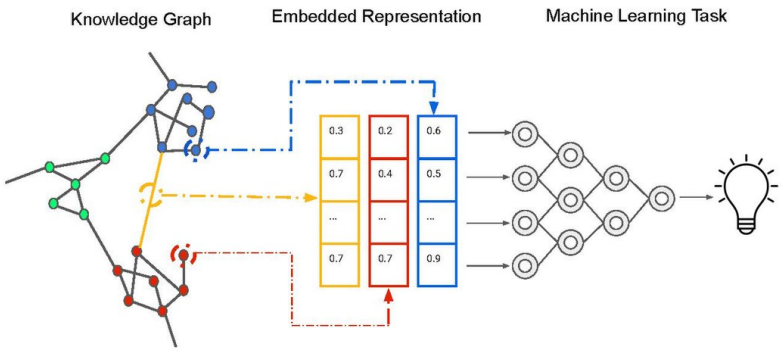

Figure 3.9: Architecture to learn KG Embeddings [9]

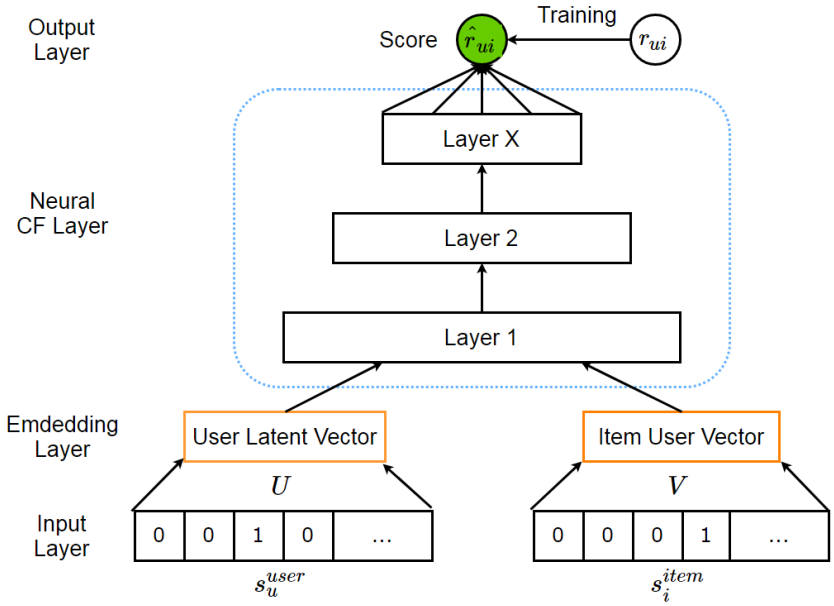

Figure 3.10: Architecture to learn User-POI Embeddings [10]

enriched with contextual information in order to generate effective POI recommendations for the users. As illustrated in the two diagrams above, the overall system involves learning of relevant KG embeddings for the POIs for a particular user and passing those through multiple layers atypical of an end-to-end deep learning architecture to learn user and POI embeddings. A simple approach like cosine similarity can then be used to obtain the POIs with nearest embeddings with the user embeddings in order to obtain the most relevant recommendations for a particular user.

An end-to-end deep learning pipeline is used to learn the user and POI embeddings.

The embeddings are then used to predict the probability of the user actually being interested in the POI visit. The real value is obtained with the user's actual checkin information for the POI in the past. The loss is computed by comparing the actual and calculated values. The loss is then propagated backwards in the deep learning model. The weights and biases for the model are updated in every iteration until the loss is minimized. When the loss is relatively small and has more or less stabilized when comparing the values between two successive epochs, the final result is obtained for the model. The model is then used to make POI recommendations for specific users in the future.

The number of hidden layers in our architecture is based on the depth of receptive field (H) which is discussed in subsequent sections. Binary Cross Entropy (BCE) Loss has been used as the loss function while Adaptive Moment (Adam) has been used as the optimizer in our training pipeline.

3.7 Learning Algorithm

The following points describe the stages involved in training for the knowledge graph aware recommender system.

1. Get all of the neighbors' incoming messages. The incoming messages for neighbours are computed in the form of embeddings using the POI knowledge graph which calculates the embeddings based on an user's preferences for specific relationships of the POI in the KG.

2. By performing the necessary aggregation function, combine those messages into a single message.

3. With a learnable weight matrix, perform matrix multiplication for the neighborhood message.

4. With a learnable weight matrix, multiply the initial node message by a matrix.

5. Combine steps 3 and 4.

6. Apply a ReLU activation function to the aggregation.

7. Repeat for as many levels as necessary until convergence is achieved. The output of the last layer is the outcome.

CHAPTER 4

EXPERIMENTAL SETUP

4.1 Dataset

The datasets used in this thesis work for testing and training purposes in this thesis work are the two location based social network based datasets from the platforms: Foursquare and Gowalla.

4.1.1 Foursquare

This dataset [30] comprises global-scale Foursquare check-in data gathered over a long period of time (in between the time period of 18 months from April 2012 to September 2013). User checkins from 415 cities in 77 countries are included. The 415 cities with the most Foursquare check-ins, each with at least 10,000 check-ins are included in the dataset.

Table 4.1: Foursquare metrics

Entities	**Count**
Checkins	1,94,108
Users	2,321
POIs	5,596
Sparsity	99.18%

4.1.2 Gowalla

Users check-in to report their presence on Gowalla, a location-based social networking site. The friendship network is undirected and the data was acquired using their public API [31]. It comprises 196,591 nodes and 950,327 edges. The user check-ins from February 2009 to October 2010 are included in the data.

Table 4.2: Gowalla metrics

Entities	Count
Checkins	4,56,967
Users	10,162
POIs	24,237
Sparsity	99.88%

4.2 Environment and Tools

The experiments were carried out on the Google Colab GPU runtime on the cloud. The tools that were used for our experiments are listed below.

4.2.1 Google's Knowledge Graph Search API

Google's Knowledge Graph Search API was used for the construction of the POI Knowledge Graph.

4.2.2 Pytorch

Pytorch was used as the end-to-end machine learning framework for storing the user and POI data as tensors and using a series of layers to compute losses and optimize gradients to obtain the best performing model across several epochs.

4.2.3 Pytorch Geometric

Pytorch Geometric, which is developed based on Pytorch, was used as the library for graph deep learning, to create and train our Knowledge Graph Neural Networks in an optimal manner.

4.2.4 Matplotlib

Matplotlib was used for certain visualizations such as showing loss plots, user and POI embeddings, etc.

4.2.5 Networkx

Networkx was used as the visualization tool for showing connectivity between nodes and connection maps for user-POI interaction.

4.3 Baselines

The performance of our proposed methodology was evaluated by comparing the results obtained with the three methods against the results produced by the existing baseline methods in identical setup. The model training and evaluation was carried out in the identical environment for baselines as well as existing methodologies using the same location based social network datasets for Foursquare and Gowalla.

The performance of the methodologies proposed in this work was compared with existing state-of-the-art baselines to assess their effectiveness. Below is a brief overview of the three baseline approaches used to compare performance.

4.3.1 PMF

PMF [32], i.e. Probabilistic Matrix Factorization, is a widely used factorization method common for other user-item recommendation problems as well.

Many present approaches to collaborative filtering are incapable of dealing with huge datasets or users with few feedbacks. The PMF model grows linearly with the amount of observations and, more crucially, works well on datasets that are huge, sparse, and highly unbalanced. The PMF model can be expanded to incorporate an adaptive prior on model parameters, demonstrating how model capacity can be adjusted automatically. For consumers who have only a few POI interactions, the model generalizes much better.

4.3.2 GeoMF

GeoMF [33], i.e. Geographical Matrix Factorization, is an established method used in POI recommendation.

For POI recommendation, this method uses weighted matrix factorization, which is more suited to collaborative filtering with implicit feedback. Furthermore, researchers on the LBSNs have observed a spatial clustering phenomena in human mobility behavior, i.e individual visiting places tend to cluster together, and shown its efficiency in POI suggestion, therefore the method adds it into the factorization model. In particular, activity area vectors of users and impact area vectors of POIs are used to supplement the latent components of users and POIs in the factorization model.

4.3.3 Rank-GeoFM

Rank-GeoFM, i.e. Ranking based Geographical Factorization Method, is the current state-of-art for POI ecommendation.

The method assumes that check-in frequency reflects consumers' visiting preferences and the factorization is learnt by accurately ranking the POIs. POIs with and without check-ins will both contribute to learning the ranking in this model, alleviating the data sparsity problem. Furthermore, the model may easily incorporate many forms of background information, such as spatial and temporal influences. The factorization is learned using a stochastic gradient descent approach.

4.4 Parameters

There are different parameters, the choice of value for which highly impacts the overall performance of the system. They are listed below.

4.4.1 Neighbour sampling size (K)

This refers to the number of neighbours that are used when information from the neighbouring nodes are aggregated. Since most of the nodes have a very large number of neighbours, aggregating all the neighbours is neither feasible nor effective since it leads to the problem of overfitting. So the value of K needs to be chosen such that it both captures essential neighbourhood information as well as prevents the problem of noise.

4.4.2 Embedding dimension (d)

This refers to the dimension of the vector that represents a node. The vector representation for the node is obtained by performing convolution operations on graphs like GraphConv to generate the representation for the node without actually having to perform feature engineering. The dimensions are generated in such a manner that the original topological structure of the graph is preserved.

4.4.3 Depth of receptive field (H)

The depth represents the ability of the model to capture long term relationships in the KG. When this value is set to 1, only the relationship between the entities that are directly connected is captured. When this value is set to a greater number, the model can also capture distant interests of the user. This also needs to be chosen very carefully as too large values might capture even the relationships that are of no interest to the user.

4.4.4 Aggregation function (σ)

Aggregation function is the approach taken while combining information coming from the neighbouring nodes. One approach is to simply add the neighbourhood information onto the receiving node. The other approach is to concatenate this information together. Another approach is to directly take a representation for the neighbourhood such as maximum value among neighbouring nodes.

4.5 AUC Results

Area Under Curve (AUC) represents the probability that the model will rank a randomly drawn positive sample (i.e. POI recommendation on which the visitor was actually interested) higher than a randomly drawn negative sample (i.e. POI recommendation on which the visitor was not interested). We computed AUC scores for different combinations of the parameters to decide upon the right values to take while performing the final set of experiments.

4.5.1 AUC Results for KGCN

Table 4.3: AUC Results for different K for KGCN

K	Foursquare	Gowalla
2	0.791	0.782
3	0.794	0.786
4	0.795	0.788
5	0.798	0.792
6	0.794	0.789

Table 4.4: AUC Results for different d for KGCN

d	Foursquare	Gowalla
4	0.789	0.775
8	0.793	0.780
16	0.797	0.782
32	0.793	0.796
64	0.790	0.784
128	0.789	0.778

Table 4.5: AUC Results for different H for KGCN

H	Foursquare	Gowalla
1	0.724	0.682
2	0.746	0.713
3	0.738	0.724
4	0.724	0.712

Table 4.6: AUC Results for different σ for KGCN

σ	**Foursquare**	**Gowalla**
sum	0.794	0.672
concat	0.790	0.654
neighbour	0.682	0.642

Based on the AUC results obtained, the optimal values for Foursquare were found to be K = 5, d = 16, H = 2 and σ = sum(). On the other hand, the optimal values for Gowalla based on the AUC results are K = 5, d = 32, H = 3 and σ = sum().

4.5.2 AUC Results for KGNN-LS

Table 4.7: AUC Results for different K for KGNN-LS

K	**Foursquare**	**Gowalla**
2	0.823	0.812
3	0.842	0.822
4	0.848	0.834
5	0.852	0.842
6	0.846	0.832

Table 4.8: AUC Results for different d for KGNN-LS

d	**Foursquare**	**Gowalla**
4	0.812	0.782
8	0.816	0.792
16	0.824	0.812
32	0.832	0.824
64	0.848	0.842
128	0.826	0.836

Table 4.9: AUC Results for different H for KGNN-LS

H	**Foursquare**	**Gowalla**
1	0.742	0.721
2	0.752	0.732
3	0.782	0.724
4	0.767	0.713

Table 4.10: AUC Results for different σ for KGNN-LS

σ	**Foursquare**	**Gowalla**
sum	0.812	0.724
concat	0.798	0.782
neighbour	0.712	0.722

Based on the AUC results obtained, the optimal values for Foursquare were found to be K = 5, d = 64, H = 3 and σ = sum(). On the other hand, the optimal values for Gowalla based on the AUC results are K = 5, d = 64, H = 2 and σ = sum().

4.5.3 AUC Results for KGAT

Table 4.11: AUC Results for different K for KGAT

K	**Foursquare**	**Gowalla**
2	0.813	0.792
3	0.825	0.798
4	0.832	0.813
5	0.834	0.816
6	0.823	0.814

Table 4.12: AUC Results for different d for KGAT

d	**Foursquare**	**Gowalla**
4	0.823	0.792
8	0.826	0.806
16	0.832	0.814
32	0.831	0.812
64	0.827	0.808
128	0.816	0.802

Table 4.13: AUC Results for different H for KGAT

H	**Foursquare**	**Gowalla**
1	0.702	0.708
2	0.712	0.711
3	0.724	0.717
4	0.718	0.713

Table 4.14: AUC Results for different σ for KGAT

σ	**Foursquare**	**Gowalla**
sum	0.813	0.792
concat	0.802	0.786
neighbour	0.791	0.712

Based on the AUC results obtained, the optimal values for Foursquare were found to be K = 5, d = 16, H = 3 and σ = sum(). On the other hand, the optimal values for Gowalla based on the AUC results are K = 5, d = 16, H = 3 and σ = sum().

CHAPTER 5

RESULTS

As the topic of our thesis is the next POI recommendation for a traveller, the results are in the form of top-N attraction recommendations for the user. The results are obtained by sorting attractions on the basis of their interest scores for a user. The top N recommendations are generated for the varying values of N set at 5, 10 and 20. For the purpose of this thesis work, top attractions are recommended for users within the radius of 1 square kilometers of the place they are currently located in.

The results obtained for different values of N for the three different graph learning algorithms used in this work are discussed in the sections below.

5.1 Loss Plot

As in any typical deep learning based system, the training of our pipeline was carried out until the losses were reasonably low. The changes in the value of losses with epochs are illustrated in the loss plots below.

5.1.1 Foursquare

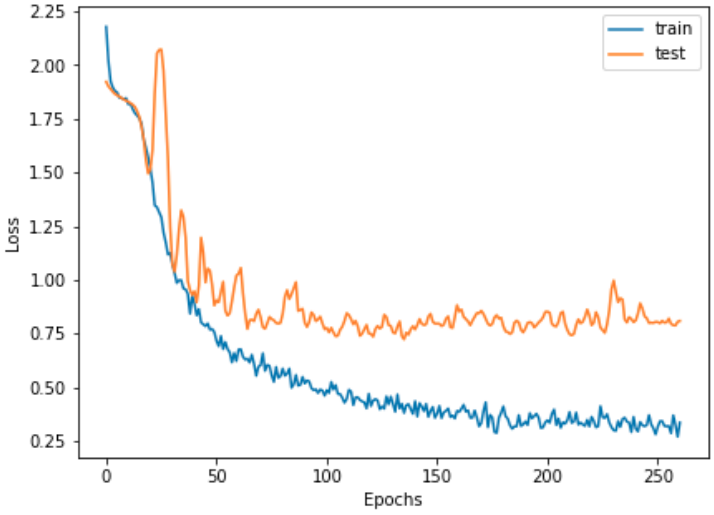

Figure 5.1: Loss Plot for KGCN (Foursquare)

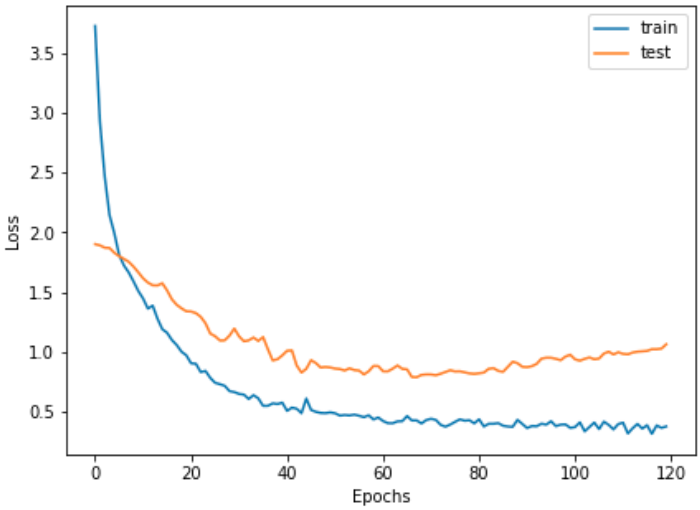

Figure 5.2: Loss Plot for KGNN-LS (Foursquare)

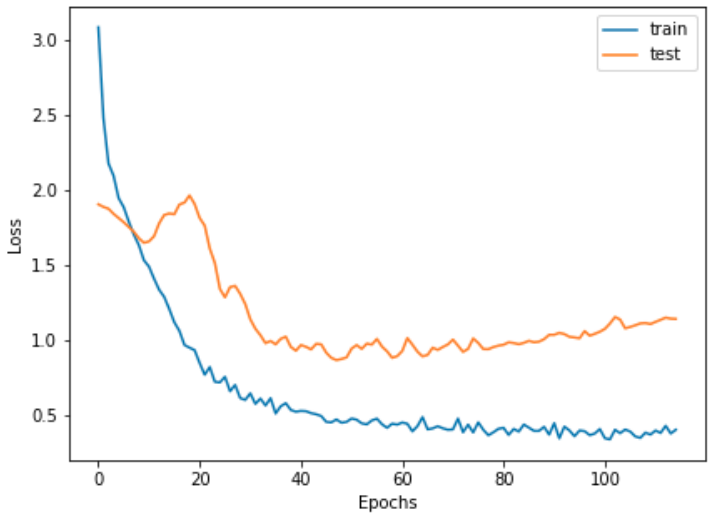

Figure 5.3: Loss Plot for KGAT (Foursquare)

From the loss plots presented above, we can see that the loss convergence for Foursquare is much quicker while training for KGAT compared to training for KGCN and KGNN-LS methods.

5.1.2 Gowalla

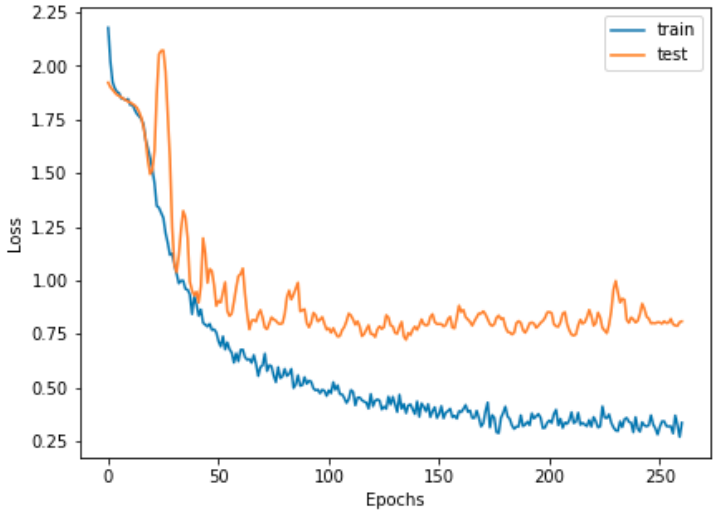

Figure 5.4: Loss Plot for KGCN (Gowalla)

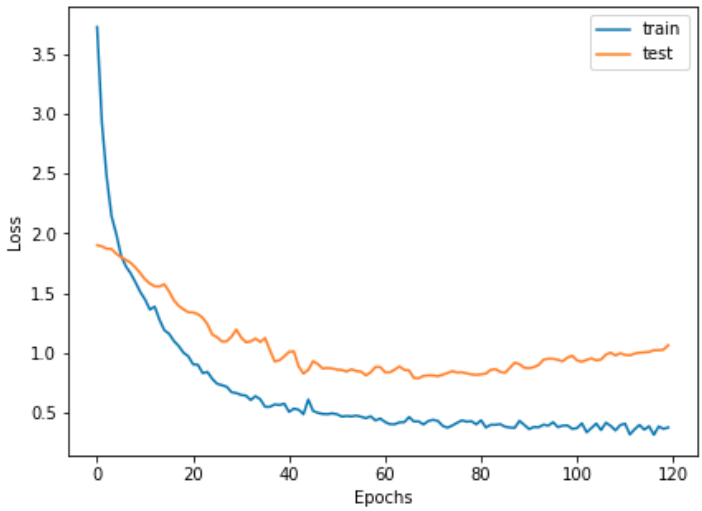

Figure 5.5: Loss Plot for KGNN-LS (Gowalla)

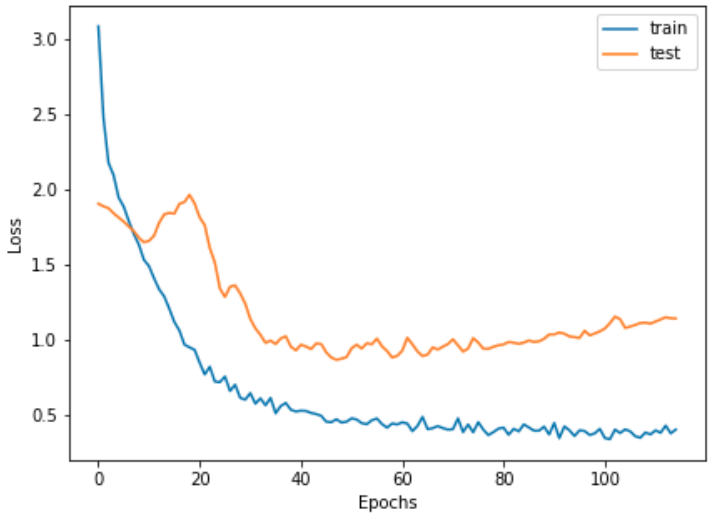

Figure 5.6: Loss Plot for KGAT (Gowalla)

From the loss plots presented above, we can see that the loss convergence for Gowalla is much quicker while training using the KGAT when compared to training using the KGCN and KGNN-LS approaches.

5.2 Visualizations

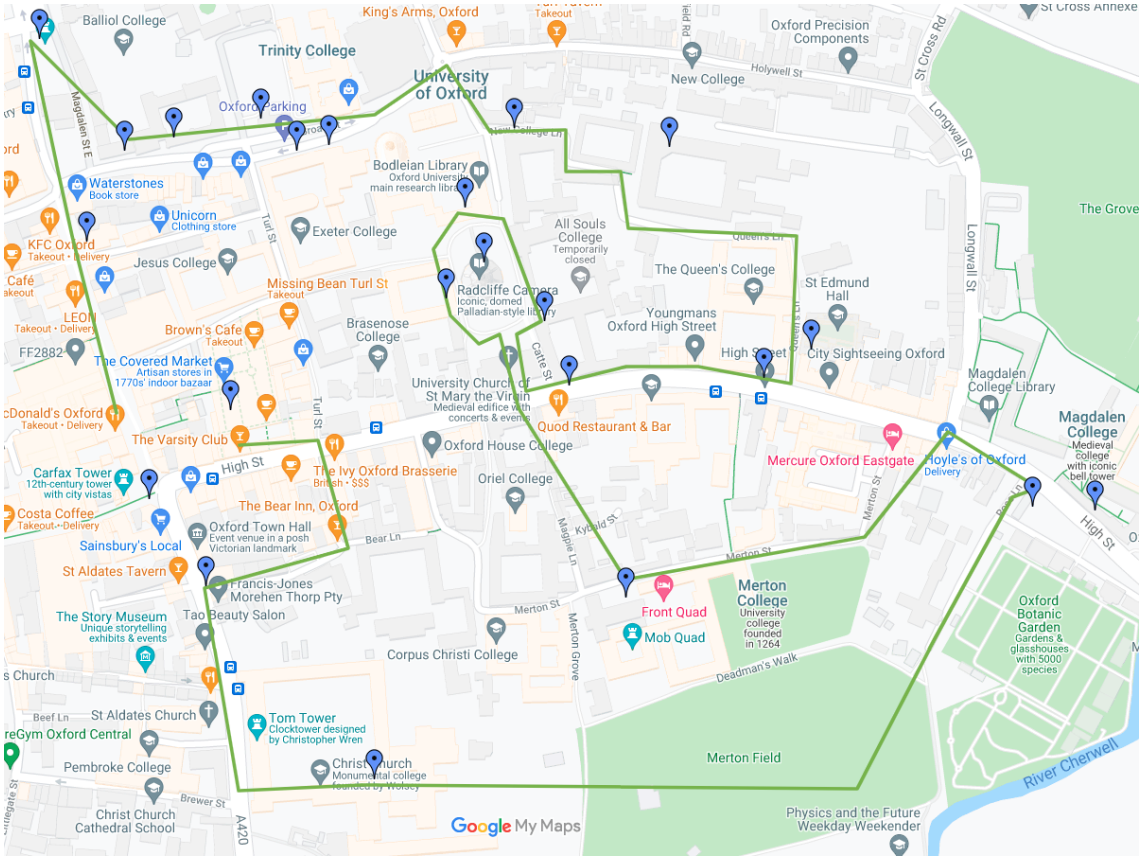

Figure 5.7: Visualization for User checkins

The above map shows the visualization for a random user's check in information on Foursquare on Google maps. The two places which are both visited by the user are connected in the graph for the user. This means for two POIs i and j, the corresponding value of $A_{ij} = 0$ given the adjacency matrix A, if both of them have been visited by the user. However, this value is 0 if the user hasn't visited both places. This is an important attribute as common checkin information determines the adjacency of the POIs in the graph which is ultimately exploited while training the Graph Neural Network.

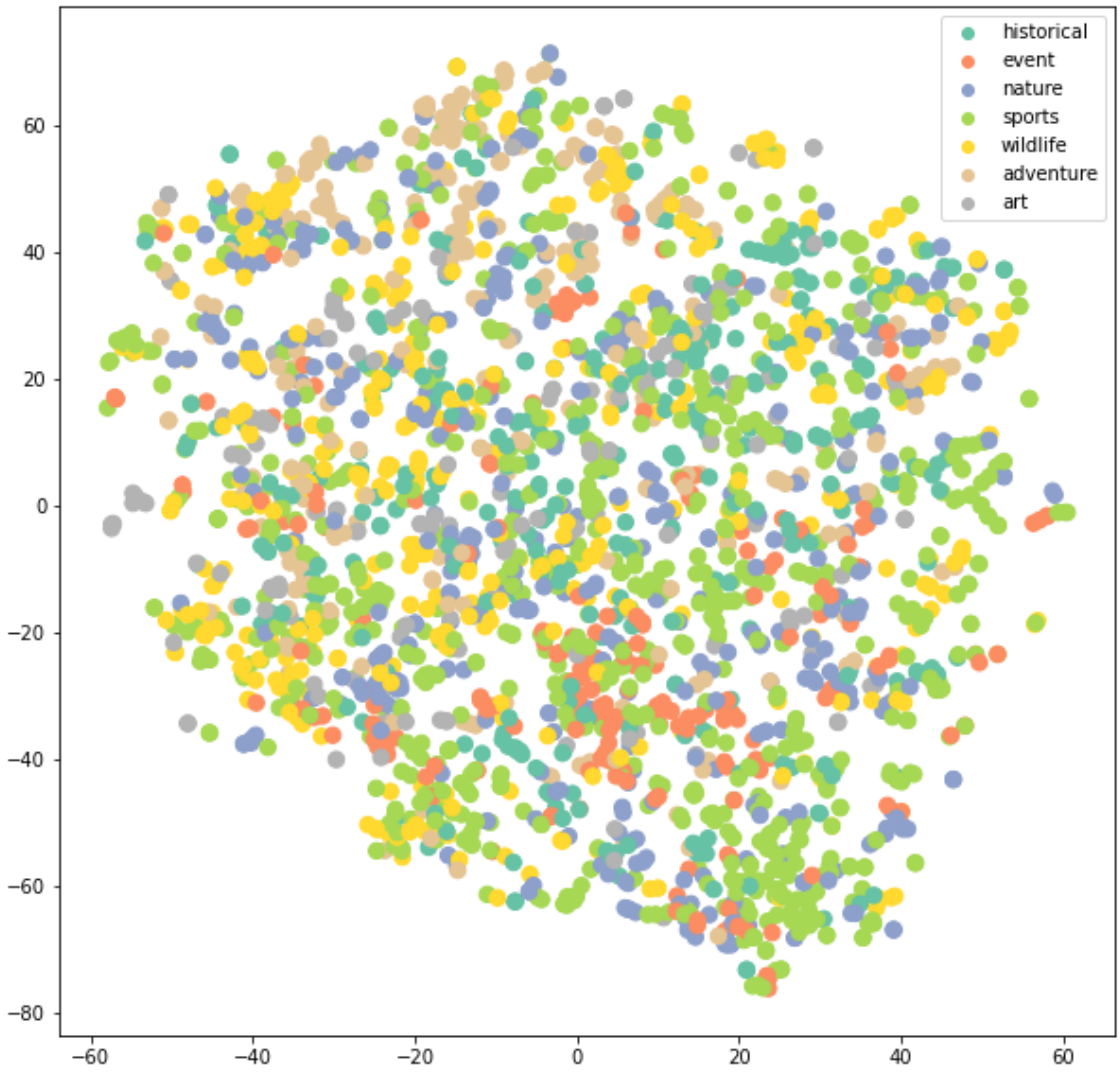

Figure 5.8: Initial POI Embeddings

The above scatter plot for POI embeddings shows the low level embeddings for potential POIs for visit for a randomly selected user within a city. The POIs are illustrated using different colors based on their category. The above plot shows seven different types of POIs which are scattered in the TSNE plot. The results are obtained when applying a single GCN layer directly without any training mechanism in place.

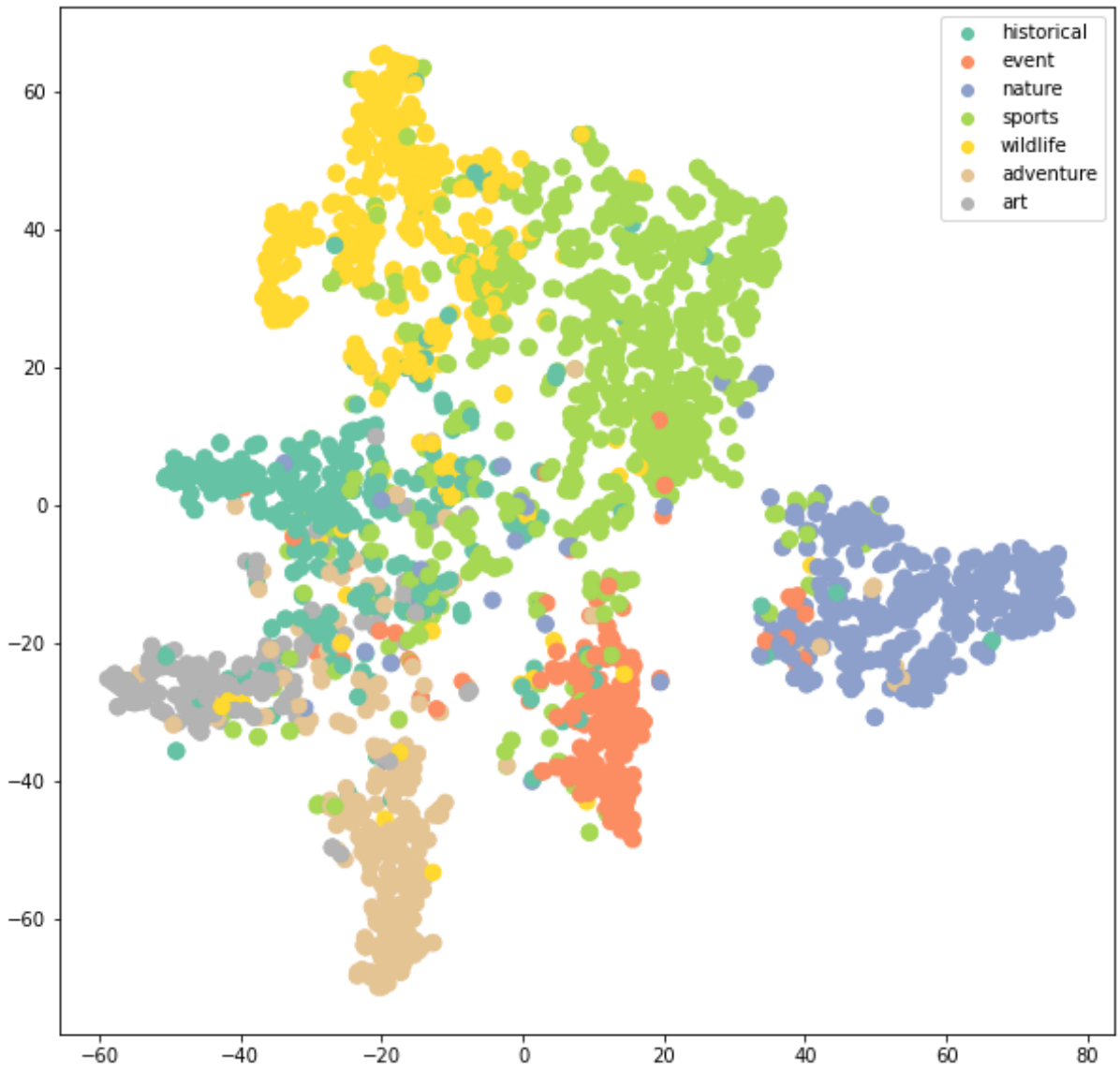

Figure 5.9: Final POI Embeddings

The above scatter plot is the final result for embeddings of potential POIs that the random user can visit in a city at the end of the training pipeline. As seen in the above diagram, POIs of similar categories are clustered together with each other in the TSNE plot. The basic idea behind using KG aware GNNs for POI recommendation is to obtain optimal user and POI embeddings which lie together in the embedding space based on their similarity, which is exactly evident from the above TSNE plot.

5.3 Top 5 Results

The top 5 recommendations for a visitor currently at Amsterdam Avenue, New York (40°48'40.3"N 73°57'28.0"W) while using the three different methods are presented below.

```
User id: 156391

Top 5 Recommendations:

Riverlink Park (0.234)
Guy Park (0.213)
Schoharie Crossing State Historic Site (0.202)
Our Lady of Matyrs Shrine (0.187)
Renaissance Harlem (0.172)
```

Figure 5.10: Top 5 Recommendations for KGCN

```
User id: 156391

Top 5 Recommendations:

Schoharie Crossing State Historic Site (0.284)
Our Lady of Matyrs Shrine (0.272)
Kirk Douglas Park (0.246)
Shuttleworth Park (0.237)
Old Fort Johnson (0.218)
```

Figure 5.11: Top 5 Recommendations for KGNN-LS

```
User id: 156391

Top 5 Recommendations:

Guy Park (0.263)
Old Fort Johnson (0.247)
General Grant National Memorial (0.238)
Riverlink Park (0.226)
Carmansville Playground (0.218)
```

Figure 5.12: Top 5 Recommendations for KGAT

5.4 Top 10 Results

The top 10 recommendations for a visitor currently at Central Park, New York (40.7831° N, 73.9712° W) while using the three methods are presented below.

```
User id: 127026

Top 10 Recommendations:

American Museum of Natural History (0.272)
Theodore Roosevelt Park (0.252)
The Obelisk (0.237)
The Mall and Literary Walk (0.232)
Belvedere Castle (0.218)
Wagner Cove (0.205)
Cherry Hill (0.201)
Delacarte Theater (0.198)
The Great Lawn (0.195)
The Ramble (0.192)
```

Figure 5.13: Top 10 Recommendations for KGCN

```
User id: 127026

Top 10 Recommendations:

The Ramble Cave (0.324)
Belvedere Castle (0.308)
Shakespeare Garden (0.293)
Theodore Roosevelt Park (0.272)
The Mall and Literary Walk (0.271)
Arthur Ross Pinetum (0.241)
Bethesda Terrace (0.230)
The Great Lawn (0.228)
The Ramble (0.226)
The Metropolitan Museum of Art (0.218)
```

Figure 5.14: Top 10 Recommendations for KGNN-LS

```
User id: 127026

Top 10 Recommendations:

Bow Bridge (0.308)
Belvedere Castle (0.298)
Alice in Wonderland (0.287)
Delacarte Theater (0.276)
The Obelisk (0.267)
The Great Lawn (0.262)
American Museum of Natural History (0.253)
Arthur Ross Pinetum (0.241)
Sheep Meadow (0.237)
Ladies Pavilion (0.231)
```

Figure 5.15: Top 10 Recommendations for KGAT

5.5 Top 20 Results

The top 20 recommendations for a visitor currently at Tokyo Tower, Tokyo (35.6586° N, 139.7454° E) with the three methods are presented below.

```
User id: 117325

Top 20 Recommendations:

Imperial Palace (0.272)
Statue of Admiral Perry (0.268)
Tower Daijingu Shrine (0.262)
Konchi-in (0.257)
Prince Shiba Park (0.252)
Megaweb Toyota City Showcase (0.247)
Zojoji Hall (0.245)
Toufuya Ukai (0.242)
Shinko-in (0.237)
Odaiba Marine Park (0.231)
Franciscan Chapel Center (0.226)
Nyoirin Kannon (0.218)
Iikura Park (0.217)
Cafe La Tout (0.211)
Tokyo Joypolis (0.208)
Seiryu-ji Temple (0.197)
Atago Shrine (0.192)
Shiba Toshogu Shrine (0.191)
Place in the Sun (0.185)
Sumida Park (0.181)
```

Figure 5.16: Top 20 Recommendations for KGCN

```
User id: 117325

Top 20 Recommendations:

Imperial Palace (0.312)
Shiba Toshogu Shrine (0.289)
Terrace Dining TANGO (0.282)
Mori Art Museum (0.275)
Franciscan Chapel Center (0.272)
Momiji Waterfall (0.268)
Meiji Jingu (0.263)
Seiryu-ji Temple (0.261)
Kodo-in (0.257)
Shinko-in (0.252)
Jirozaemon Inari Shrine (0.249)
Rikugien Gardens (0.244)
Nyoirin Kannon (0.241)
Cafe La Tout (0.237)
Maxell Aqua Park (0.234)
Tokugawa Estate Mausoleum (0.232)
Odaiba Marine Park (0.226)
Place in the Sun (0.218)
Prince Shiba Park (0.207)
Watarium Art Museum (0.198)
```

Figure 5.17: Top 20 Recommendations for KGNN-LS

```
User id: 117325

Top 20 Recommendations:

Imperial Palace (0.284)
Mori Art Museum (0.278)
Statue of Admiral Perry (0.272)
Momiji Waterfall (0.271)
Konchi-in (0.268)
Meiji Jingu (0.265)
The Land of Gas Founder (0.262)
Franciscan Chapel Center (0.258)
Megaweb Toyota City Showcase (0.252)
Kodo-in (0.221)
Zojoji Hall (0.219)
Toufuya Ukai (0.214)
Jirozaemon Inari Shrine (0.212)
Cafe La Tout (0.211)
Sky Lounge Stellar Garden (0.208)
Terrace Dining TANGO (0.202)
MOS Burger Tokyo Tower (0.198)
Atago Shrine (0.192)
XEX Atago Green Hills (0.191)
Place in the Sun (0.185)
```

Figure 5.18: Top 20 Recommendations for KGAT

CHAPTER 6

EVALUATION

6.1 Evaluation Metrics

Our evaluation of the results is done by computing the Precision, Recall and F1-scores for Top-N recommendations with different values of N at 5, 10 and 20. The scores are computed by comparing whether the attraction recommended by our model would have been actually visited by the traveler at a given instance in the past.

The different types of scenarios for a recommendation by our model are elaborated below.

True Positive (TP):
These are the attractions that were recommended by the model and also actually visited by the travelers.

True Negative (TN):
These are the attractions that were not recommended by the model and also not visited by the travelers in reality.

False Positive (TP):
These are the attractions that were recommended by the model but actually not visited by the travelers.

False Negative (FN):
These are the attractions that were visited by the travelers which were not actually recommended by our model.

The calculations for different scores are then carried out as:

Precision:

Precision is the measure of the rate at which the model was able to recommend places that were actually visited by the travelers. It measures how often travelers actually visited the places recommended by our model compared to the recommendations generated by the model. It is calculated with the following formula.

$$Precision(P) = \frac{TP}{TP + FP} \tag{6.1}$$

Recall:

Recall is the measure of the rate at which travelers actually visited the places recommended by the model. It measures how often travelers actually visited the places recommended by our model compared to their overall visits. It is calculated with the following formula.

$$Recall(R) = \frac{TP}{TP + FN} \tag{6.2}$$

F1-Score:

F1-Score is the harmonic mean of Precision and Recall values.

$$F1{-}Score(F1) = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.3}$$

6.2 Results of Evaluation

The comparison between the results obtained by our model along with the results obtained from the current existing baselines is presented in the sections below. For this comparison, both the results for existing baselines, reported by authors in [24] as well as the results obtained while running the experiments using the baselines on our environment are included. The legends in the following charts appended by (R) denote reported results while those appended by (O) represent the results

obtained on our own environment.

6.2.1 Results on Foursquare Dataset

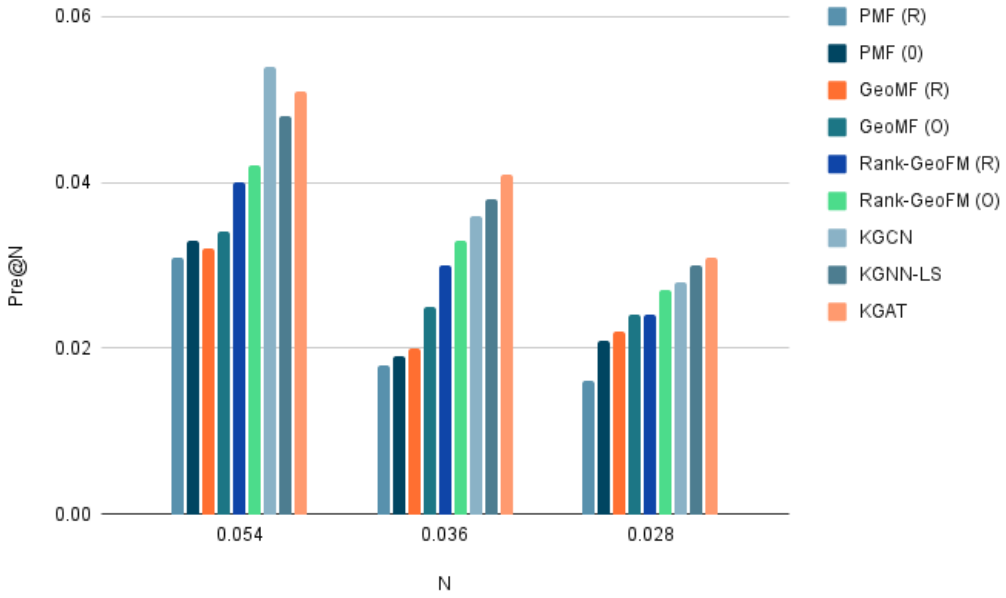

Figure 6.1: Precision on Foursquare dataset

From the experiments carried out on Foursquare dataset, it's observed that in general, KG aware recommender systems perform better than the ones that don't use KG information. While all three approaches perform better than existing baselines, KGAT was seen to provide better results.

As illustrated in the graphs, KGCN outperformed all other methods in terms of Precision and F1-score for top-5 recommendations but apart from that, KGAT was the best performer across all evaluation categories.

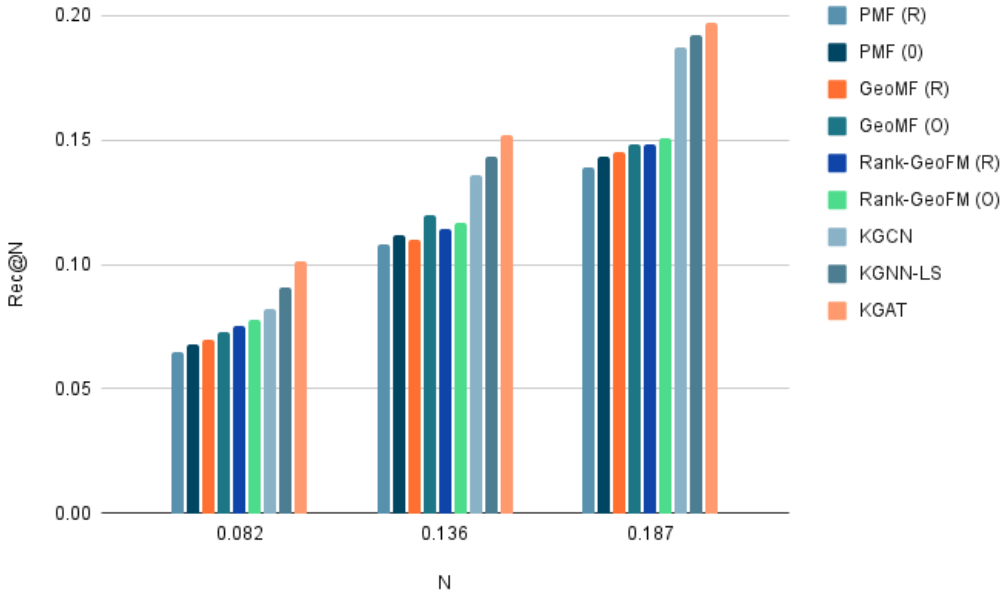

Figure 6.2: Recall on Foursquare dataset

6.2.2 Results on Gowalla Dataset

From the experiments carried out on Gowalla, it's observed that in general, KG aware recommender systems perform better than the ones that don't use KG information. While all three approaches perform better than existing baselines, KGAT was seen to provide better results.

As illustrated in the graphs, KGNN-LS yielded better results compared to all other methods in terms of Precision and F1-score for top-10 recommendations but apart from that, KGAT was the best performer across all other evaluation categories.

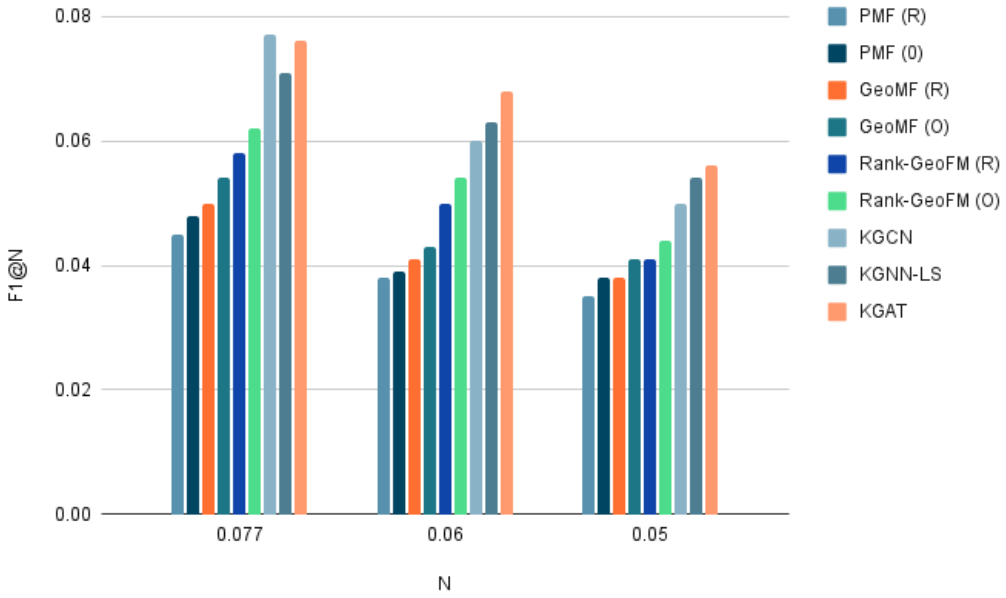

Figure 6.3: F1-score on Foursquare dataset

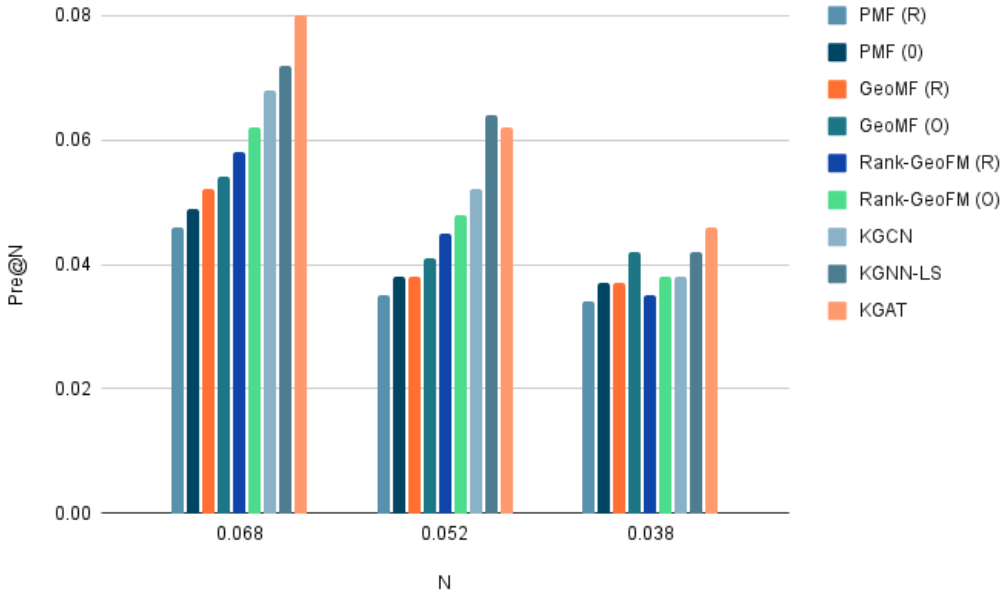

Figure 6.4: Precision on Gowalla dataset

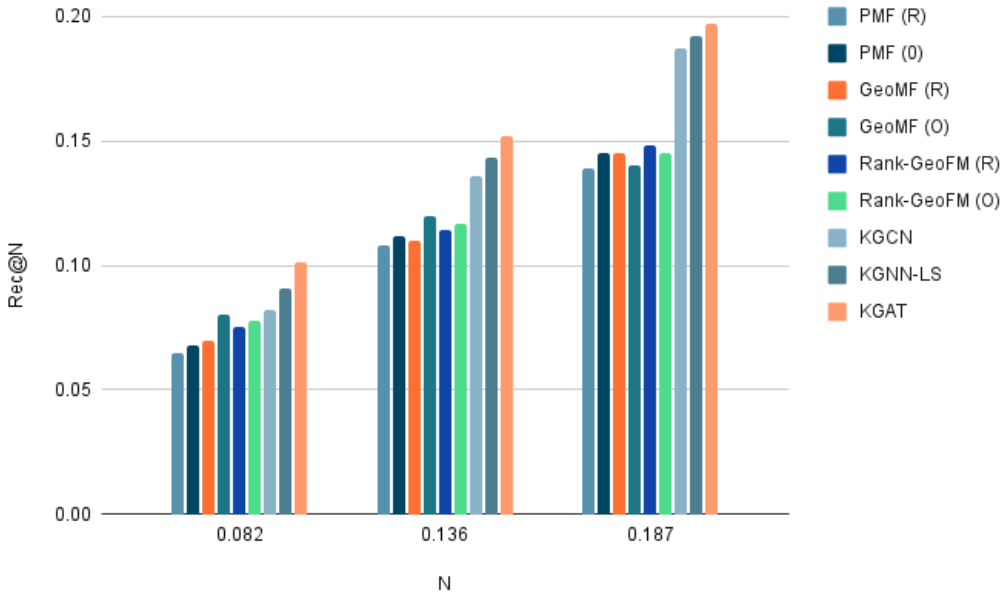

Figure 6.5: Recall on Gowalla dataset

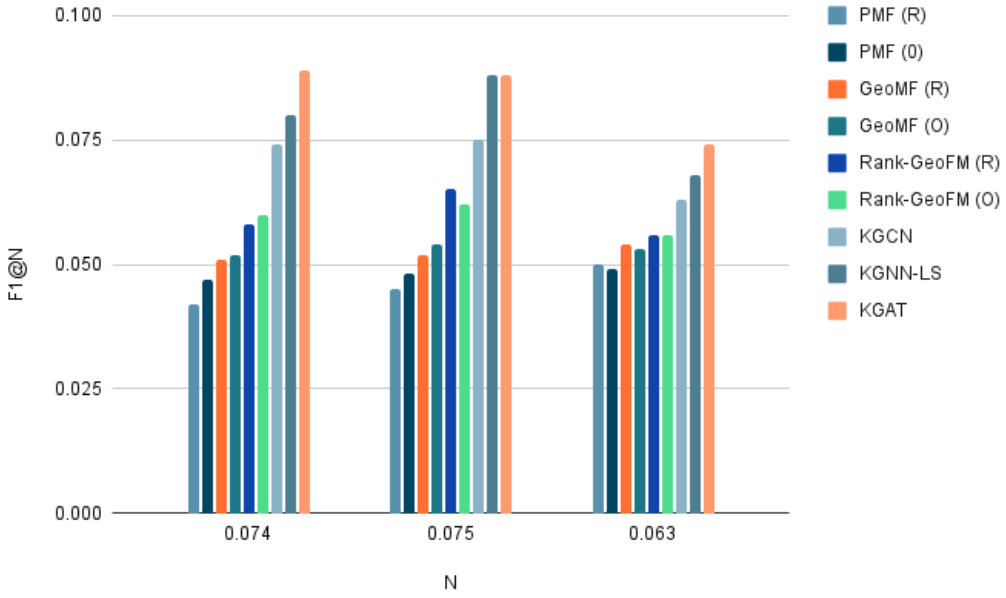

Figure 6.6: F1-score on Gowalla dataset

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

Based on the experiments and results obtained in this work, it can be seen that all the three knowledge graph aware recommendation methods used in general outperform the existing baselines in the field of travel recommendation. Although the other two methods i.e. KGCN and KGNN-LS have also performed better than KGAT in some instances, mostly KGAT is better at generating recommendations as per our experimental results.

Compared to the best performing baseline of Rank-GeoFM, performance improvement of 24.19%, 14.51% and 22.58% was observed respectively for KGCN, KGNN-LS and KGAT on the Foursquare dataset for top 5 results in terms of F1-score. The performance improvement observed for top 10 results meanwhile was 13.20%, 14.86% and 28.30% respectively. Finally, the improvement observed was respectively 18.27%, 25.58% and 30.23% respectively in the case of top 20 results. The performance improvement achieved seemed to improve considerably with the increasing number of recommendations generated.

Similarly, when compared to the best performing baseline of Rank-GeoFM, performance improvement of 19.35%, 29.03% and 43.54% was observed respectively for KGCN, KGNN-LS and KGAT on the Gowalla dataset for top 5 results in terms of F1-score. The performance improvement observed for top 10 results meanwhile was 10.29%, 29.41% and 29.41% respectively. Finally, the improvement observed was respectively 6.77%, 15.25% and 25.42% respectively in the case of top 20 results. Just opposite to the Foursquare dataset, the performance improvement noticeably decreased when the number of recommendations generated was increased in the case of Gowalla dataset. However, the performance improvement achieved on

Gowalla was in general better than the Foursquare dataset. Considering the fact that Gowalla dataset was much sparse when compared with Foursquare, we can deduce that our model provides much better performance improvement on sparse datasets compared to the existing baseline methods.

7.2 Future Work

Several avenues have been identified as well to improve upon the findings of this work.

1. The focus in this thesis as well as most of the existing research has centred on extracting contextual information for items (POIs in our work) from the KG. Extraction of such information for users can be an area of work to explore upon.

2. Extraction of contextual information for both users and items from a heterogeneous KG representing both users and items in an end-to-end manner can be an area of work for further investigation.

3. The model proposed in this paper is universal and can be applied to generate recommendations on the local context of Nepal as well by preparing appropriate user-POI interaction dataset and corresponding knowledge graph with rich POI related information. So construction of user-POI interaction matrix along with KG for Nepal's local context can also be a promising area for exploration.

REFERENCES

[1] Yashu Seth. Introduction to question answering over knowledge graphs, 2019. https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/.

[2] Igor Belykh, Martin Hasler, M. Lauret, and Henk Nijmeijer. Synchronization and graph topology. *International Journal of Bifurcation and Chaos*, 15, 11 2005.

[3] Chun Lu, Philippe Laublet, and Milan Stankovic. Travel attractions recommendation with knowledge graphs. In Eva Blomqvist, Paolo Ciancarini, Francesco Poggi, and Fabio Vitali, editors, *Knowledge Engineering and Knowledge Management*, pages 416–431, Cham, 2016. Springer International Publishing.

[4] Nikita Sharma. Introduction to graph neural networks, 2020. https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/.

[5] Thomas Kipf. Graph convolutional networks, 2016. https://tkipf.github.io/graph-convolutional-networks/.

[6] Yuexin Huang and Hailong Sun. *Best Answerers Prediction With Topic Based GAT In Q & A Sites*, page 156–164. Association for Computing Machinery, New York, NY, USA, 2020.

[7] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. In *The World Wide Web Conference*, WWW '19, page 3307–3313, New York, NY, USA, 2019. Association for Computing Machinery.

[8] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. KGAT: knowledge graph attention network for recommendation. *CoRR*, abs/1905.07854, 2019.

[9] Wikipedia contributors. Knowledge graph embedding, 2021. https://commons.wikimedia.org/wiki/File:Knowledge_Graph_Embedding.pdf.

[10] Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *CoRR*, abs/1707.07435, 2017.

[11] Duan Hong Liu Qiao, Li Yang. Knowledge graph construction techniques. *Journal of Computer Research and Development*, 53(3):582, 2016.

[12] Andreas Töscher and Michael Jahrer. The bigchaos solution to the netflix grand prize. 01 2009.

[13] Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Trans. Intell. Syst. Technol.*, 8(1), September 2016.

[14] Amit Singhal. Introducing the knowledge graph: things, not strings. 2012. https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[15] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. Sound and music recommendation with knowledge graphs. *ACM Trans. Intell. Syst. Technol.*, 8(2), October 2016.

[16] Cunchao TU, Cheng YANG, Zhiyuan Liu, and Maosong SUN. Network representation learning: an overview. *SCIENTIA SINICA Informationis*, 47:980–996, 08 2017.

[17] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013, 01 2013.

[18] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 32–36, New York, NY, USA, 2017. Association for Computing Machinery.

[19] Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. Recommendations in location-based social networks: A survey. *GeoInformatica*, 19, 07 2015.

[20] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, page 17–23. AAAI Press, 2012.

[21] Matthias Braunhofer, Mehdi Elahi, Francesco Ricci, and Thomas Schievenin. *Context-Aware Points of Interest Suggestion with Dynamic Weather Data Management*, volume 2014, pages 87–100. 01 2014.

[22] Bin Liu and Hui Xiong. *Point-of-Interest Recommendation in Location Based Social Networks with Topic and Location Awareness*, pages 396–404. 05 2013.

[23] Igo Ramalho Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. On planning sightseeing tours with tripbuilder. *Information Processing & Management*, 51(2):1–15, 2015.

[24] Xutao Li, Gao Cong, Xiao-Li Li, Tuan-Anh Nguyen Pham, and Shonali Krishnaswamy. Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 433–442, New York, NY, USA, 2015. Association for Computing Machinery.

[25] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The World Wide Web Conference*, WWW '19, page 151–161, New York, NY, USA, 2019. Association for Computing Machinery.

[26] Weiping Song, Zhijian Duan, Ziqing Yang, Hao Zhu, Ming Zhang, and Jian Tang. Explainable knowledge graph-based recommendation via deep reinforcement learning. *CoRR*, abs/1906.09506, 2019.

[27] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In

Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, page 353–362, New York, NY, USA, 2016. Association for Computing Machinery.

[28] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 968–977, New York, NY, USA, 2019. Association for Computing Machinery.

[29] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 09 2016.

[30] Dingqi Yang. Foursquare dataset. https://sites.google.com/site/yangdingqi/home/foursquare-dataset, [Online; accessed September 5, 2021].

[31] Jure Leskovec. Gowalla dataset. https://snap.stanford.edu/data/loc-gowalla.html, [Online; accessed September 5, 2021].

[32] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1257–1264, Red Hook, NY, USA, 2007. Curran Associates Inc.

[33] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. Geomf: Joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 831–840, New York, NY, USA, 2014. Association for Computing Machinery.

APPENDIX A

Turnitin Similarity Index

Updated Final Thesis Report/Updated Final Thesis Report/075MSCSK_010_Nabin.pdf

ORIGINALITY REPORT

21%
SIMILARITY INDEX

PRIMARY SOURCES

1	flipkarma.com Internet	284 words — 3%
2	deepai.org Internet	139 words — 1%
3	www.groundai.com Internet	132 words — 1%
4	arxiv.org Internet	112 words — 1%
5	petar-v.com Internet	93 words — 1%
6	tkipf.github.io Internet	77 words — 1%
7	citeseerx.ist.psu.edu Internet	63 words — 1%
8	"ECAI 2020", IOS Press, 2020 Crossref	62 words — 1%
9	www.kdd.org Internet	52 words — 1%