CS6534 – Guided Study
Semester B 2019-2020

Final Report

Submitted on 04-05-2020

Topic: GNN for Hotel Recommendation

Student: DAI Jingzhi
No. xxxxxx
Supervisor: Dr. SONG Linqi

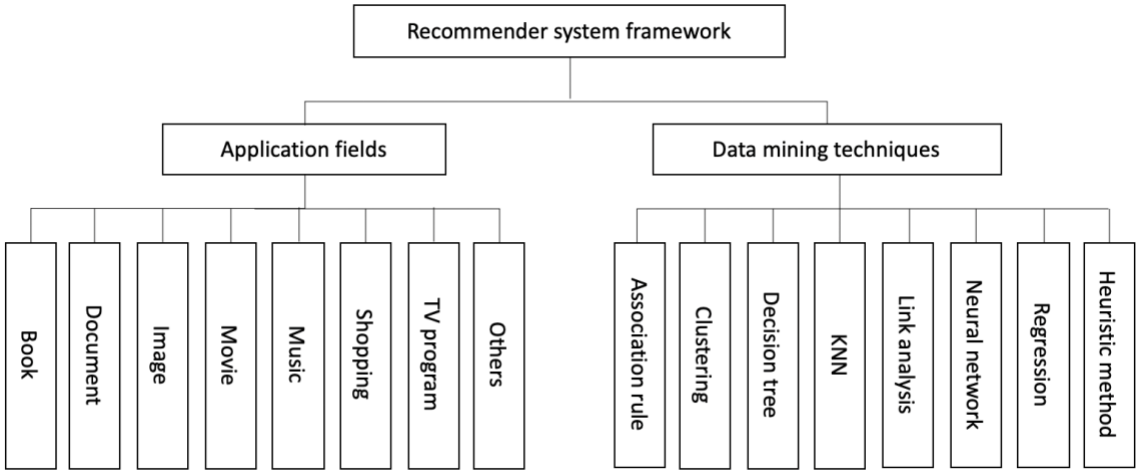
Contents

1 Introduction

1.1 Introduction of recommender system

In the social background of big data, the so-called personalized recommendation is to recommend information and products to users according to their interest characteristics and purchase behaviours. With the fast development of the technologies in electronic commerce field and huge expansion of the e-commerce scale, as well as the increase in the amount and type of consumer goods, users need more time to choose the goods they like, which has directly caused information overload problem. In this environment, recommender system is particularly important and able to save a lot of time for the users. Personalized recommendation systems find the users' personalized needs and interests according to the analysis of the user's behaviour, and then recommend the corresponding information and products to the user. The recommendation system mainly depends on the algorithms of data mining and machine learning. To summarize, the reasons why we need recommender systems include solving the information overload problem to help people save time and efforts in choosing suitable goods, efficiently increasing the user stickiness for a trade and helping merchants make better recommendations while saving time and labor expenses.

[1] The following figure is the recommender system framework:

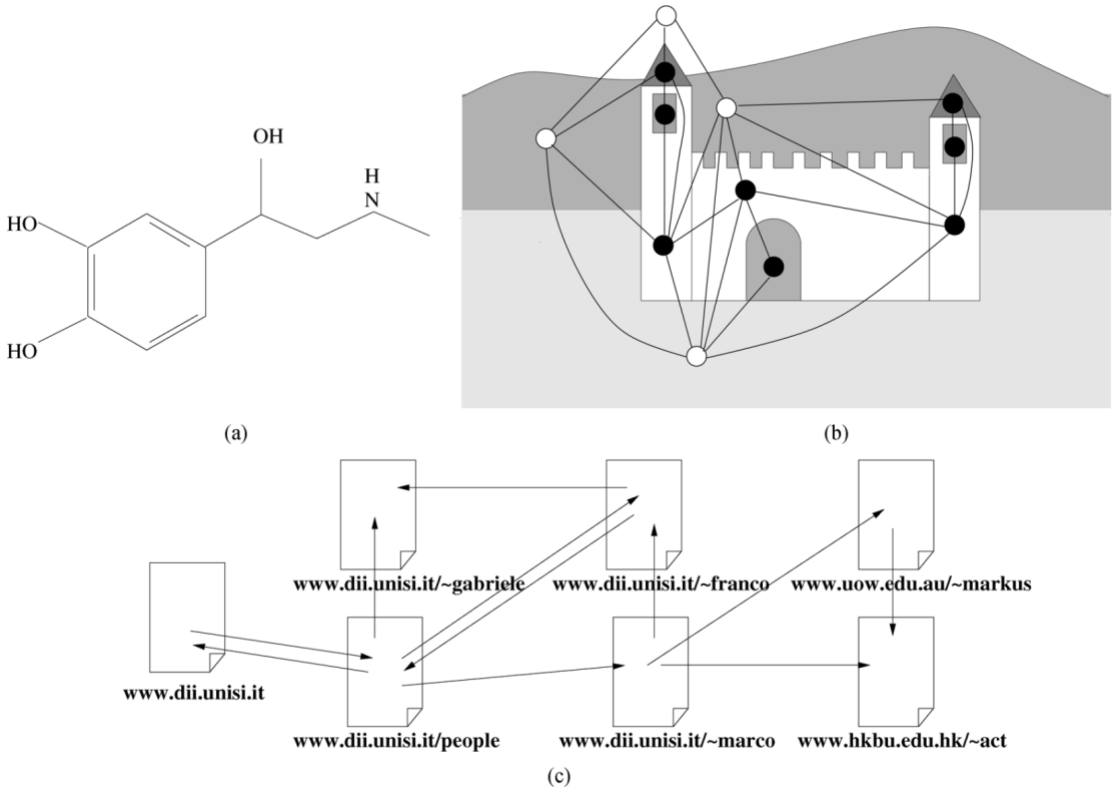


The recommendation system can be regarded as a special form of information filtering system, which mainly includes "collaborative filtering recommendation", "content-based recommendation", "recommendation based on association rules", "recommendation based on knowledge reasoning" and "combination-based recommendation". How to apply GNN algorithms to recommendation systems is a new field.

1.2 Introduction to GNN

1.2.1 Brief introduction

Graph is a kind of data structure. The general graph structure consists of vertices and edges. A graph G can be described by the set of vertices V and edges E. G=(V,E). The edges can be directed or undirected. The vertices can be also called nodes. The node vector of a graph can be represented by X(v), where v∈V. [2]These are some applications in which the data can be describe as graphs: (a) a chemical compound, (b) an image, and (c) a subset of the web.

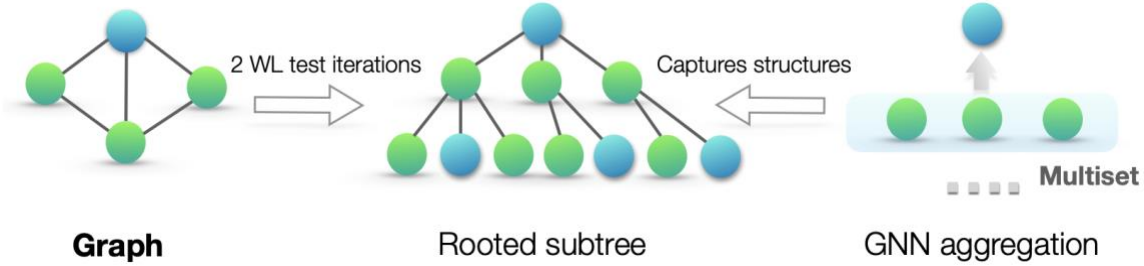


Graph neural network should be some models, methods and applications of deep learning in graph structure data. Nodes contain entity information and edges contain relationship information. It uses graph structure and node feature X(v) to learn the representation vector H (V) of one node or the representation vector h(G) of the whole graph.

[3]The latest GNNs all follow the strategy of "neighbourhood aggregation", in which we update the representation vectors by aggregating the representation vectors of its neighbouring nodes. After K iterations, the k-th layer of GNN is:

$a_v^{(k)} = AGGREGATE^{(k)}(\{h_u^{(k-1)}: u \in N(v)\}),\ h_v^{(k)} = COMBINE^{(k)}(h_v^{(k-1)}, a_v^{(k)})$

The following picture can represent an overview of the theoretical framework:



[4]At present, many learning tasks need to deal with the data of graph structure, such as physical system modelling, molecular fingerprints learning, protein interface prediction and diseases classification. All of these require that the model can learn relevant knowledge from the input of graph structure. Due to the convincing performance and high interpretability, graph neural network has been used in many fields such as social network, recommendation system, biology, medicine and so on, making great contribution to social progress.

1.2.2 Application of GNN in Recommendation System

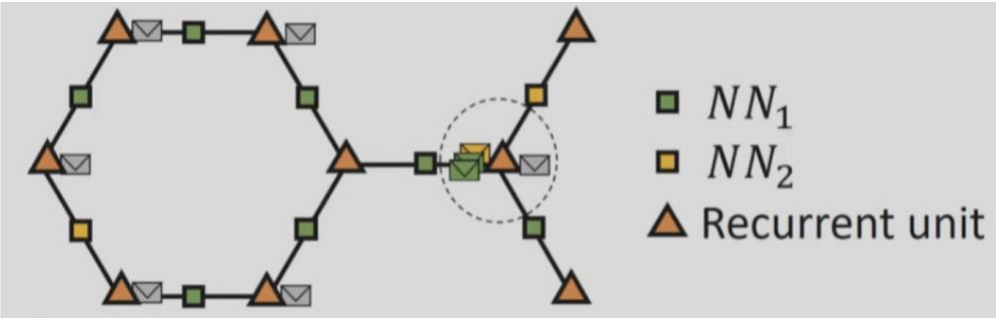
- There are 4 main types of GNN used for recommender systems:

1) GNN is used for the graph composed of user-item interactions with no additional information.
2) GNN is used for knowledge graph.
3) GNN is used for user social network.
4) GNN is used for graph built from user sequential behaviours.

- There are three progressions of GNN in the context of recommender systems [5]:

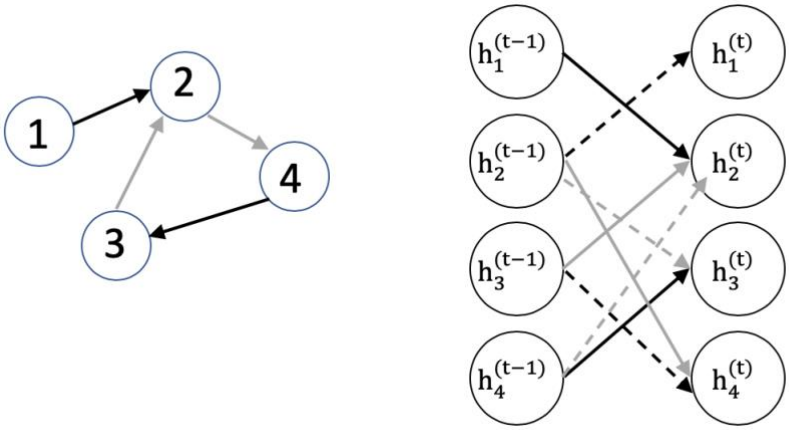
1) Graph Neural Networks

Basically, each node is in a recurrent unit with a grey envelope containing its information. Each edge type has a neural network representation. Obviously, the influence of one node on the other decreases exponentially with distance. So, it's difficult to propagate information for a long distance. While Gated Graph Neural Network overcomes this problem.



2) Gated Graph Neural Network

The main change is that we use gated recurrent unit. We let the recurrence happen for some number of timesteps and let the information backpropagating through time.



But in GGNN, we find it difficult to model when we face flexible and complex node interactions. On top of that, Node-wise Graph Neural Networks was invented.

3) Node-wise Graph Neural Network [6]

NGNN aims to model edge-wise interactions with node-wise parameters. Specifically, when two nodes communicate state information on the edge, the information will be first transformed by a function with specific parameters related to the two nodes. In this way, NGNN is able to address the limitation of GGNN.

1.3 **Motivation**

The main motivation for picking this project is that I want to know how graph neural network algorithm works in hotel recommendation system. It's a new try in the hotel recommender system field, which inspired me to do this project. The recommender system based on graphs takes items and users as nodes. By analysing the content information and the relationship between item and item, user and user, user and item, recommendation system based on graphs can generate high-quality recommendations. Graph neural network provides great potential to hotel recommendation since data in hotel recommender systems can be represented as a user-item graph, while learning latent relationship between users and items is the key point. As we know, hotel preference mainly will be affected by price, facilities and services, but these factors are difficult to qualify by data, a learning system based on graphs can utilize the interaction between users and items to make accurate recommendations in this situation.

1.4 **Potential outcome**

In this study, the data we use is Expedia Hotel Recommendation dataset published on the Kaggle website. [7] The dataset collected over a period of time from 2013 to 2014 contains a variety of features that can give us insights into the process of selecting a hotel. The train set contains 37,670,293 records, and the test set contains 2,528,243 records. But in practice, we might randomly select some samples to do the experiment, about one million records from the train set. In addition, the data set provides some latent features for each destination recorded in the train and test set.

We mainly use neural graph collaborative filtering algorithm as recommendation framework, compared with matrix factorization method. The programming language is Python. We will conduct data pre-processing, data cleaning, data modelling, prediction operations and use Recall Rate as the measure of the quality of recommendation.

2 Related work

2.1 Existing and related solutions

2.1.1 Objective

The objective was to analyse users' hotel-booking data and predict the possibility a user will stay at different hotel clusters. To some degree, it can be regarded as multi-class classification problem. There are much work has been done by classification methods to complete recommendation currently, such as Decision Tree, Random Forest, Naive Bayes, XGBoost, Ensemble Learning and so on, but little work about graph neural network algorithm has been applied to hotel recommendation.

In this study, we only consider the users and items and the interactions between them. The core of our study is applying graph structure to hotel recommendation, so we analyzed different algorithms based on graphs.

The graph-based model is one of the most important part in field of recommender system. In fact, many researchers call neighborhood-based model graph-based model, for neighborhood-based model can be regarded as a simple form of graph-based model.

Before we study the graph-based model, we first need to represent the user's behavior data in the form of graph. The user's behavior data we discuss is composed of binary array, in which each (u, i) represents the user's behavior to item i. This kind of data is easy to be represented by a binary graph.

	$\mathbf{Item_1}$	…	$\mathbf{Item_k}$	…	$\mathbf{Item_n}$
$\mathbf{User_1}$	$R_{1,1}$	…	$R_{1,k}$	…	/
…	…	…	…	…	…
$\mathbf{User_j}$	$R_{j,1}$	…	/	…	$R_{j,n}$
…	…	…	…	…	…
$\mathbf{User_m}$	/	…	$R_{m,k}$	…	$R_{m,n}$

2.1.2 Related solutions

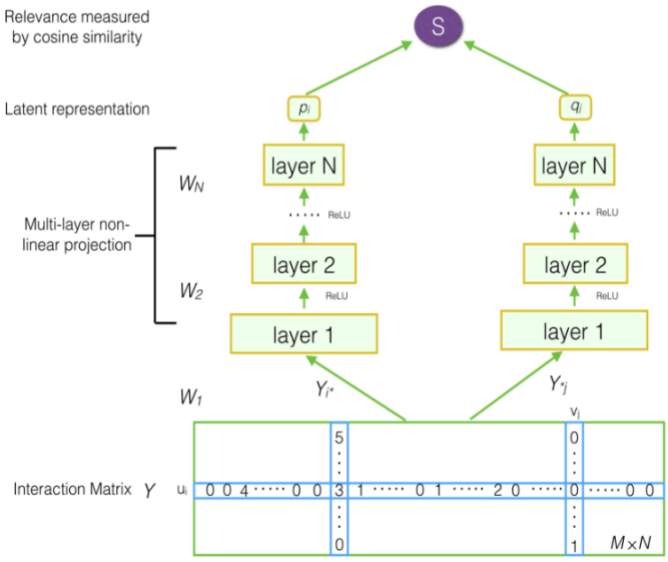
1) Deep Matrix factorization Models

The matrix factorization means decomposing a matrix into the product of two or more matrixes. For user-item rating matrix, we name it $R_{m\times n}$. We can decompose $R_{m\times n}$ into $P_{m\times k}$ and $Q_{k\times n}$. We should make the product of $P_{m\times k}$ and $Q_{k\times n}$ able to return to the original matrix $R_{m\times n}$:

$$R_{m\times n} \approx P_{m\times k} \times Q_{k\times n} = \hat{R}_{m\times n}$$

We can use deep matrix factorization model [8] to map the users and items into a low-dimensional space with non-linear projection. First, we construct the interaction matrix with explicit rating or non-preference implicit feedback. Secondly, we input the matrix to a deep learning architecture of users and items to conduct multi-layer non-linear projection and then

we can obtain the final latent representation. Thirdly, we get the relevance measured by cosine similarity for further algorithm.



From this picture, we can see what the input of user i is the i-th row in the interaction matrix and we mark it as Y_{i*}, what the input of item j is the j-th column in the interaction matrix and we mark it as Y_{*j}. Then these two parts of input respectively get vector p_i and q_j as output that are also the implicit vector representation of user i and item j.

$$p_i = f_{\theta_N^U}(...f_{\theta_3^U}(W_{U2}f_{\theta_2^U}(Y_{i*}W_{U1}))...)$$

$$q_j = f_{\theta_N^I}(...f_{\theta_3^I}(W_{V2}f_{\theta_2^I}(Y_{*j}^T W_{V1}))...)$$

Then, the prediction score is obtained by cosine distance of p_i and q_j:

$$\hat{Y}_{ij} = F^{DMF}(u_i, v_j|\Theta) = cosine(p_i, q_j) = \frac{p_i^T q_j}{\|p_i\| \, \|q_j\|}$$

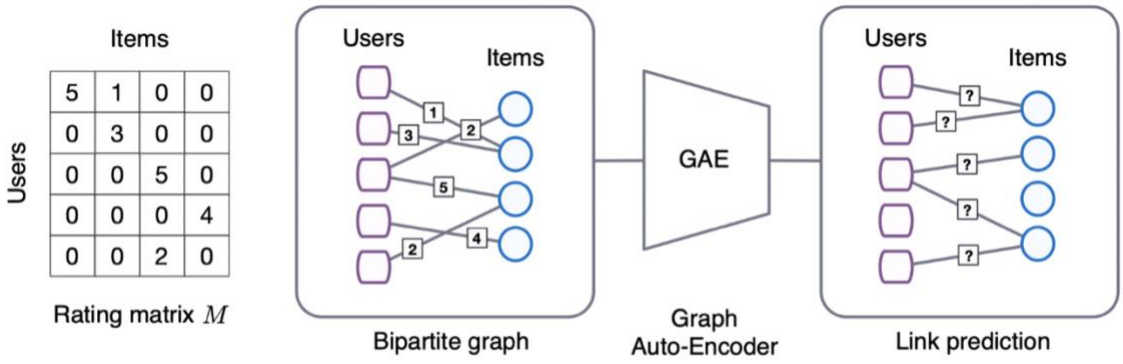
2) Graph Convolutional Matrix Completion

Graph convolutional matrix completion [9] is a graph self-encoder framework proposed to complete the matrix on the basis of research on processing graph structure data with deep learning. This self-encoder generates the implicit features between users and items through information transfer in bipartite interaction graph. The implicit representations between users and items are used to reconstruct scoring links through a bilinear decoder.

matrix completion	**link prediction on graph**
interaction data	bipartite graph (between user and item nodes)
observed ratings/purchases	links
content information	node features
predict ratings	predict labeled links

1. Matrix completion is as a connection prediction in bipartite graphs

- M: Rating matrix, the dimension is $N_u \times N_v$, and N_u is the number of users, N_v is the number of items.
- M_{ij} is the rating of $user_i$ for $item_j$, M_{ij}=0 means a rating that hasn't been observed.



The above picture shows the whole model process. In a binary user-item interaction graph, the task of matrix completion (i.e. prediction of unobserved interactions) can be transformed into the problem of link prediction, and the graph self-encoder with end-to-end training can be used for modelling.

- Interactive data can be represented by undirected graph G: $G = (\mathcal{W}, \mathcal{E}, \mathcal{R})$
- $\mathcal{W} = (\mathcal{W}_u \cup \mathcal{W}_v)$; $\mathcal{W}_u$ is the set of users, its dimension is N_u; $\mathcal{W}_v$ is the set of items, its dimension is N_v.
- Edge $(u_i, r, v_j) \in \mathcal{E}$ is with labels of rating levels, $r \in \{1, \dots, R\} = R$.

2. Revisiting graph auto-encoders

This method adopts the setup introduced in [13], because it can effectively utilize convolution weight sharing and allow the edge information to be included in the form of node features.

Graph self-encoder model: $Z = f(X, A)$

- Input: a feature matrix X: $N \times D$ and a graph adjacency matrix A， D is the number of node features.
- Output: a node embedding matrix $Z = [z_1, \dots, z_N]^T$ of $N \times D$, H is the size of embedding.

Decoder: $\check{A} = g(Z)$

- Input: node embedding pair (z_i, z_j)
- Output: Prediction of entries in adjacency matrix: $\check{A}_{ij}$

For the bipartite graph $G = (\mathcal{W}, \mathcal{E}, \mathcal{R})$, we can reformulate the encoder:

- $[Z_u, Z_v] = f(X_u, X_V, M_1, \dots, M_R)$, in which $M_r \in \{0,1\}^{N_u \times N_v}$ is represented as a adjacency matrix related to rating level type $r \in R$ (the element is 0 or 1, element =1 means we have observed the rating, element=0 means we don't have observed the rating)
- Z_u is the embedding matrix of user, the dimension is $N_u \times H$
- Z_v is the embedding matrix of item, the dimension is $N_v \times H$
- The embedding of a single $user_i$ is a true valued eigenvector z_i^u

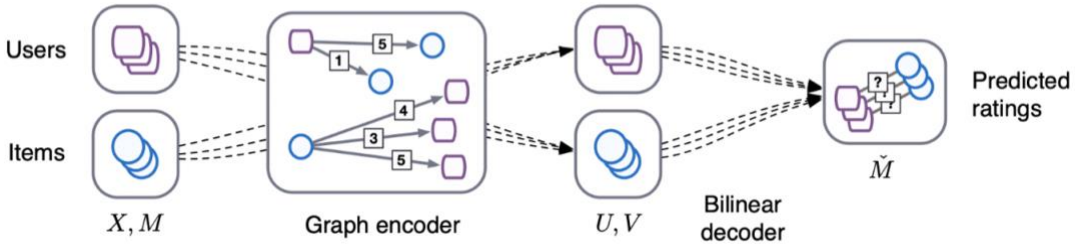
- The embedding of a single $item_j$ is a true valued eigenvector z_j^v

Similarly, we can reformulate the decoder:

- $\breve{M} = g(Z_u, Z_v)$ means a formula acting on the embedding of user and item and returns a reconstructed rating matrix $\breve{M}$, the dimension is $N_u \times N_v$

We can use the reconstruction error in predicted level of minimized $\breve{M}$ and the ground-true level observed in M trains the self-encoder of this graph. Reconstruction error can be:

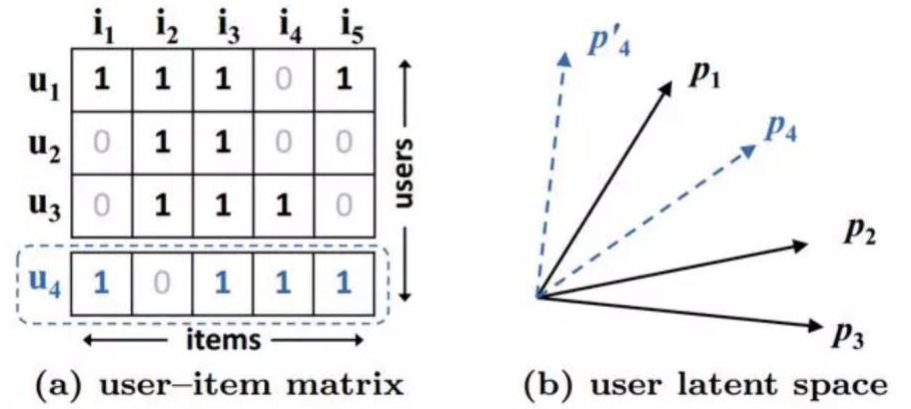
- Root mean square error
- The cross entropy loss can be used when the rating levels are classified into different categories



3) Neural Collaborative Filtering

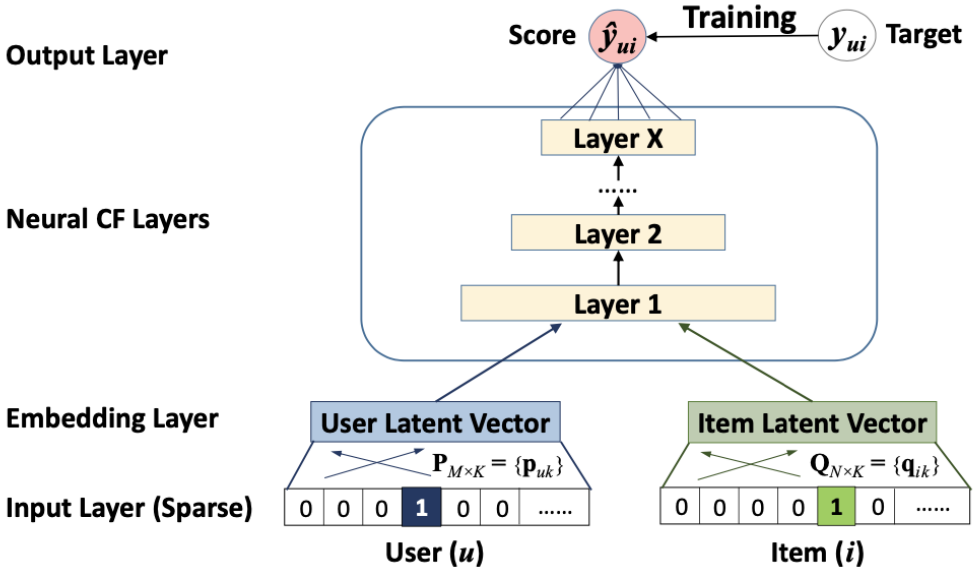
Neural collaborative filtering [10] is proposed in 2017 at first. By replacing inner product of matrix with neural structure, we can learn any function from the data. Based on that, neural network based on collaborative filtering is proposed as general framework, which can express and generalize matrix decomposition. In order to improve the non-linear modelling ability of neural collaborative filtering, multi-layer perceptron is proposed to learn the interaction function between users and projects.

Matrix Factorization uses the inner product of feature vector p_u and q_i to estimate u's preference for i: $\hat{y}_{ui} = \mathrm{f}(u, i|p_u, q_i) = \mathrm{p}_u^T q_i = \sum_{k=1}^{K} p_{uk} q_{ik}$
Because users and items are mapped to the same feature space, then we use the cosine of inner product, which is the cosine of two vectors, to measure the similarity; similarly, we can also use inner product to measure the similarity of two users.

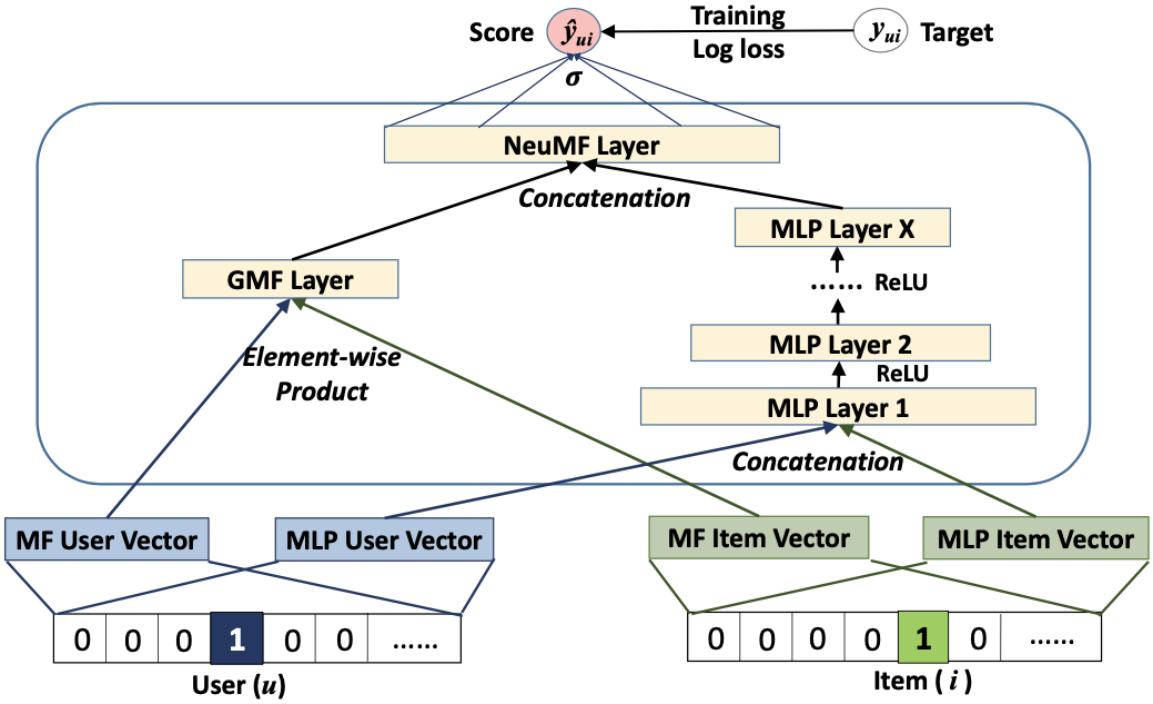


Using Jaccard coefficient (set the ratio of intersection of set A and set B to union of set A and B as the similarity degree of set) as the real similarity degree between users, then in the users from following rating matrix, $S_{23} > S_{12} > S_{13}$, the geometric relationship in latent space is shown in the right figure. When user 4 is added, $S_{41} > S_{43} > S_{42}$, then we make the feature of user 4 close to user 1. However, no matter how you put it, you cannot make user 3 closer to user 4 than user 2. This is the limitation of using inner product to describe similarity. We can increase K to solve this problem, but there is a risk of over fitting.

The following figure shows the NCF framework, the id of user and item is first obtained a feature through embedding layer, then input the feature into multilayer perceptron (MLP) to get the scoring, and train the result with the objective function of pointwise.



In addition, a conventional idea is that two vectors are spliced together as the input of MLP, which is called NeuMF.



Since the model uses implicit feedback (1 / 0), if the square error function is used:

$$L_{sqr} = \sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2$$

Then actually it's equivalent to the assumption that the predicted y_{ui} is a Gaussian distribution with f(u, i|θ) as the mean value, however this assumption is obviously not suitable for binary implicit feedback. Therefore, we can regard the value to be predicted as a classification problem, that is, whether the user has interacts with the item.

By training with logistic regression, we get a binary cross entropy loss, where y^- can be all or part of the negative samples (i.e. the negative sampling method):

$$p(\mathcal{Y}, \mathcal{Y}^-|\mathbf{P}, \mathbf{Q}, \Theta_f) = \prod_{(u,i)\in\mathcal{Y}} \hat{y}_{ui} \prod_{(u,j)\in\mathcal{Y}^-} (1 - \hat{y}_{uj}).$$

$$\begin{aligned} L &= -\sum_{(u,i)\in\mathcal{Y}} \log \hat{y}_{ui} - \sum_{(u,j)\in\mathcal{Y}^-} \log(1 - \hat{y}_{uj}) \\ &= -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui}). \end{aligned}$$

2.2 Comparison between existing solutions

1) Deep Matrix factorization

Deep Matrix factorization maps the high-dimensional user-item scoring matrix to two low-dimensional user-item matrixes and can be used to solve the problem of data sparsity.

Advantages:

- It is easy to program and realize with low time and space. The mapping of one high-dimensional matrix to two low-dimensional matrixes will save storage space, and the training process can be completed offline. Moreover, the parameters obtained by offline training can be directly used for real-time recommendation. So it's very convenient to perform matrix factorization during the process of recommendation.

- The accuracy of prediction is relatively high, and the accuracy of prediction is higher than that of frequently used collaborative filtering methods and content filtering methods.

Disadvantages:

- Model training is time-consuming.

- The recommendation results are not very interpretable as a result of the fact that each dimension of the decomposed user and item matrix is difficult to be interpreted by real-life concepts. It cannot be named after the concept of reality, but can only be understood as the underlying semantic space.

2) Graph Convolutional Matrix Completion

Advantages:

- Previous graph based methods of recommendation system usually use multi-level pipeline, including graph feature extraction model and link prediction model, all of which are trained separately. However, by using the end-to-end learning technology to model the graph structure data, the results can be significantly improved, especially using the graph auto encoder for unsupervised learning and link prediction.

- When there is structured external information (such as social network) in the recommendation graph, the advantage of using matrix completion as link prediction task on bipartite graph becomes more obvious. Combining these external information with interactive data can alleviate the performance bottleneck related to cold start problems. Experimental results show that the proposed graph self-encoder model can effectively combine interactive data with side information. It is further proved that this method can compete with the latest and most advanced methods in the pure collaborative filtering scenario.

Disadvantages:

- [11] The quality requirement of content is very high, and in the case of poor content quality, it will lead to much lower performance than matrix decomposition.

3) neural collaborative filtering

Advantages:

- Neural collaborative filtering is a technique to replace the inner product with neural network, which can learn any functional relationship between latent features. The algorithm can capture the interactions between users and items, which are the most critical elements of collaborative filtering.

Disadvantages:

- NCF is relatively more complex than matrix factorization, and matrix factorization model can be regarded as special case of NCF.

2.3 Relationship between the existing solutions and the proposed solution

In the research of recommendation systems, collaborative filtering (CF) plays an important role. The direct understanding of collaborative filtering algorithm is: recommend items that other users with the same interest have browsed to users. The popular way to solve this algorithm is to use matrix factorization (MF) to decompose the rating matrix of users and items, so as to get user embeddings and item embeddings. However, MF can only calculate the linear interaction between user and item, ignoring the non-linear interaction between user and item, so with the help of neural network or deep learning model, some neural network frameworks can be used to learn the nonlinear collaborative filtering model, such as neural collaborative filtering (NCF).

The key of recommendation system is to get the embeddings of users and items. The problem in the current collaborative filtering method is that the collaborative signal is not encoded in

the embedding process, which causes that the embedding result is not able to capture the effect of collaborative filtering. Therefore, we put forward neural graph collaborative filtering (NGCF) based on graph neural network, which uses graph structure to express the interaction information of users and items, and models the high-order connectivity of users and items in graph network, so as to inject the collaborative signal into the embedding process explicitly.

3 System modelling and structure

3.1 Proposed solution

The methodology we choose is [12] neural graph collaborative filtering.

The collaborative filtering model extends the MF embedding function by learning the deep representation from the rich side information of the project, and the neural collaborative filtering model replaces the MF interaction function with the nonlinear neural network. All of the above methods are effective, but they cannot provide proper embeddings for collaborative filtering, because there is a lack of a cooperative signal. It is in the interaction between users and items to reveal the behaviour similarity between users or items. More specifically, most of the existing methods only use descriptive features, such as id and attribute, to build embedded functions without considering user-item interactions. These functions are only used to define the target function of model training.

Therefore, we must rely on interaction functions to make up for the deficiency of suboptimal embeddings. Although the direct use of interactive functions can make up for the shortcomings of suboptimal embeddings, the scale of interaction can easily reach millions or even more in the actual development, making it difficult to extract the desired cooperation signals. So in this work, we solve the problems by using the high-level connectivity from the user-item interaction and code the cooperation signals in the interactive graphic structure.

3.2 Justifications of the design choices

The problems GNN solves can be roughly divided into three categories: node classification, link forecast and graph classification. In node classification, the task is to embed each node in the prediction graph. The problem is usually trained in a semi-supervised way, in which only some graphs are marked. Typical applications of node classification include citation networks, reddit posts, YouTube videos and Facebook friends. In link prediction, the task is to understand the relationship between entities in the graph and predict whether there is a connection between two entities. For example, a recommendation system can be regarded as a link prediction problem, in which a model is assigned to a group of users' comments on different products. The task is to predict users' preferences and adjust the recommendation system to push more relevant and interested products according to users. In graph classification, the task is to classify the whole graph into different categories. It is similar to image classification, but the target changes to image domain. There are many industrial problems that can be applied to graph classification. For example, in chemistry, biomedicine

and physics, models are endowed with molecular structures and are required to classify objects into meaningful categories. It speeds up the analysis of atoms, molecules or any other type of structured data.

We can see GNN for hotel recommendation is a link prediction problem. We can get a user-item rating graph based on the hotel data, which facilitate the process of applying deep learning methods into hotel recommendations. To a large extent, we have confidence in improving users' experience by applying GNN to hotel recommendation.

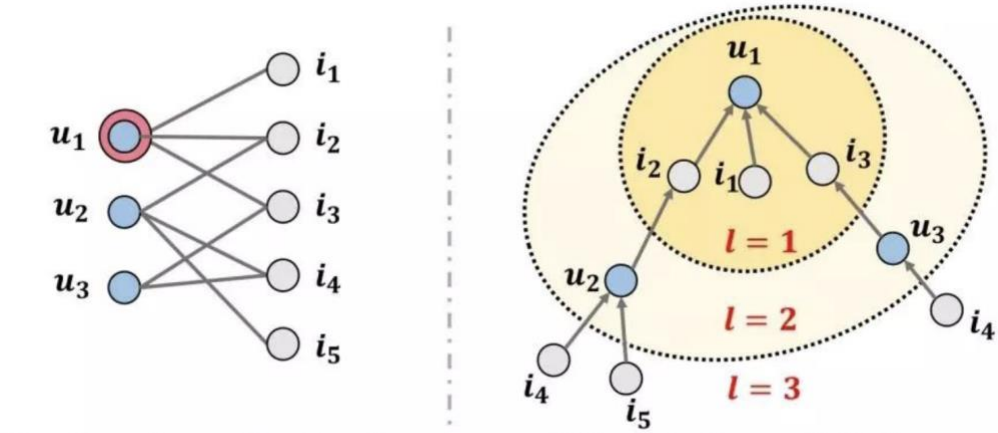
3.3 how they address existing limitations

Existing CF methods (e.g., MF, FISM, AutoRec) don't model high-order connectivity explicitly. So the main limitations are:
- Embedding function only considers descriptive features (e.g., ID, attributes)
- User-item interactions are not considered

GNCF contribution: CF modelling with high-order connectivity via GNN

4 Methodology and algorithms used in the system design

4.1 High-order connectivity



[7]The basic interaction diagram of user-item interaction is shown on the left, the double circle represents the user u_1 to be predicted, the right diagram is the tree structure extended by the user u_1 as the root node in the left diagram.

From the right diagram, it can be found that item i_1 and i_5 have the same path length of 3, but it is obvious that u_1 has more interests in i_4 than i_5, because there are two paths connected by u_1 and i_4, they are $i_4 \rightarrow u_2 \rightarrow i_2 \rightarrow u_1$ and $i_4 \rightarrow u_3 \rightarrow i_3 \rightarrow u_1$, while there is only one path for u_1 and i_5, it's $i_5 \rightarrow u_2 \rightarrow i_2 \rightarrow u_1$. Through the tree structure, we can see u_1's interest in the items and the connectivity between the item and user. This is the concept of higher-order connectivity.

4.2 NGCF Framework

4.2.1 The embedding layer of user/item

This section will provide initialization information of user and item; It can be either random initialization or initialization using some features of user and item themselves. The embedding layer can provide the required embedding for interaction layers.

$$\mathbf{E} = [\underbrace{\mathbf{e}_{u_1}, \cdots, \mathbf{e}_{u_N}}_{\text{users embeddings}}, \underbrace{\mathbf{e}_{i_1}, \cdots, \mathbf{e}_{i_M}}_{\text{item embeddings}}].$$

Note that the initialized embedding can be considered as a 0-order representation, namely:

$$e_u^{(0)} = e_u \; and \; e_i^{(0)} = e_i$$

4.2.2 Embedding propagation layers

Embedding Propagation, inspired by GNNs:
• Propagate embeddings recursively on the graph
• Construct information flows in the embedding space

1) first-order propagation

The first-order propagation mainly includes message construction and message aggregation.

- **Message construction**

Message construction is generating message from one neighbor. Given (u, i), the message propagated from i to u can be defined as:

$$\mathbf{m}_{u \leftarrow i} = f(\mathbf{e}_i, \mathbf{e}_u, p_{ui})$$

In this study, f (·) can be represented as:

$$m_{u<-i} = p_{ui}(W_1 e_i + W_2(e_i \odot e_u)) = \frac{1}{\sqrt{|N_u||N_i|}}(W_1 e_i + W_2(e_i \odot e_u))$$

- $m_{u<-i}$: message passed from i to u
- $\frac{1}{\sqrt{|N_u||N_i|}}$: discount factor
- $W_2(e_i \odot e_u)$:
  • message dependent on the affinity, distinct from GCN, GraphSage, etc.
  • pass more information to similar nodes
- w_1, w_2 are the trainable weight matrices to distill useful information for propagation, $\odot$ denotes the element-wise product.

- **Message aggregation**

Message aggregation is updating ego node's representation by aggregating message from all neighbors. In this part, we aggregate the messages propagated from the neighborhood of u to refine the representation. specifically, we define the aggregation function as:

$$\mathbf{e}_u^{(1)} = \text{LeakyReLU}\left(\mathbf{m}_{u \leftarrow u} + \sum_{i \in \mathcal{N}_u} \mathbf{m}_{u \leftarrow i}\right)$$

- $m_{u\leftarrow \mathrm{u}}$: self-connections
- $\sum_{i\in N_u} m_{u\leftarrow \mathrm{i}}$: the connections from all neighbors of u

When it comes to high-order propagation, the user u can be represented as:

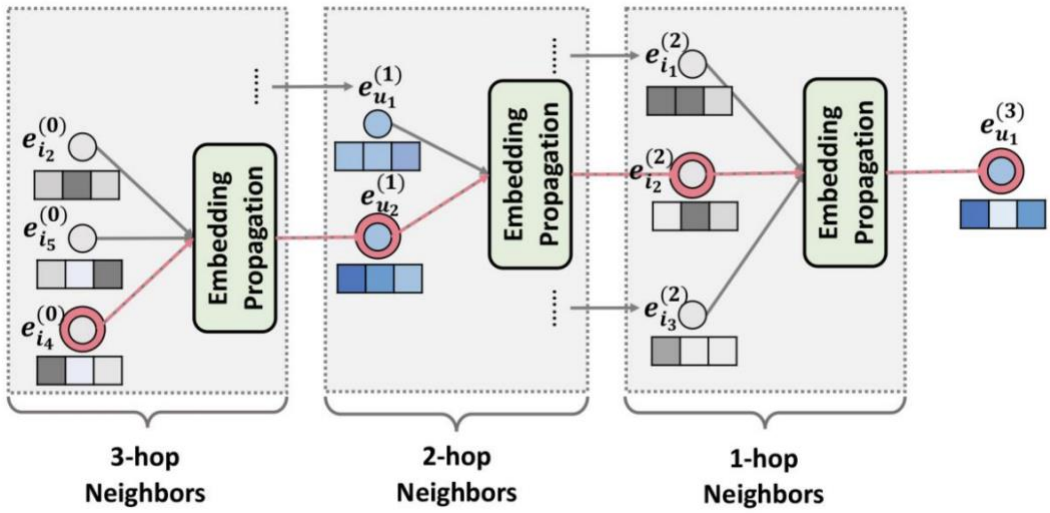
$$\mathbf{e}_u^{(l)} = \mathrm{LeakyReLU}\left(\mathbf{m}_{u\leftarrow u}^{(l)} + \sum_{i\in\mathcal{N}_u} \mathbf{m}_{u\leftarrow i}^{(l)}\right)$$

- $e_u^{(l)}$: representation of u at the l-th layer

And the messages being propagated can be defined as:

$$\begin{cases} \mathbf{m}_{u\leftarrow i}^{(l)} = p_{ui}\left(\mathbf{W}_1^{(l)}\mathbf{e}_i^{(l-1)} + \mathbf{W}_2^{(l)}(\mathbf{e}_i^{(l-1)} \odot \mathbf{e}_u^{(l-1)})\right) \\ \mathbf{m}_{u\leftarrow u}^{(l)} = \mathbf{W}_1^{(l)}\mathbf{e}_u^{(l-1)}, \end{cases}$$

• The collaborative signal like u1 ← i2 ← u2 ← i4 can be captured in the embedding propagation process.
• Collaborative signal can be injected into the representation learning process.



4.2.3 Model prediction

This part will gather embeddings of different propagation layers, then predict the matching degree between users and items, so as to complete the prediction. We join the L-order node representations together respectively, then we conduct the prediction by inner product. The multiple represents for u is $e_u^{(0)} || \ldots ||e_u^{(L)}$, the multiple represents for i is $e_i^{(0)} || \ldots ||e_i^{(L)}$.

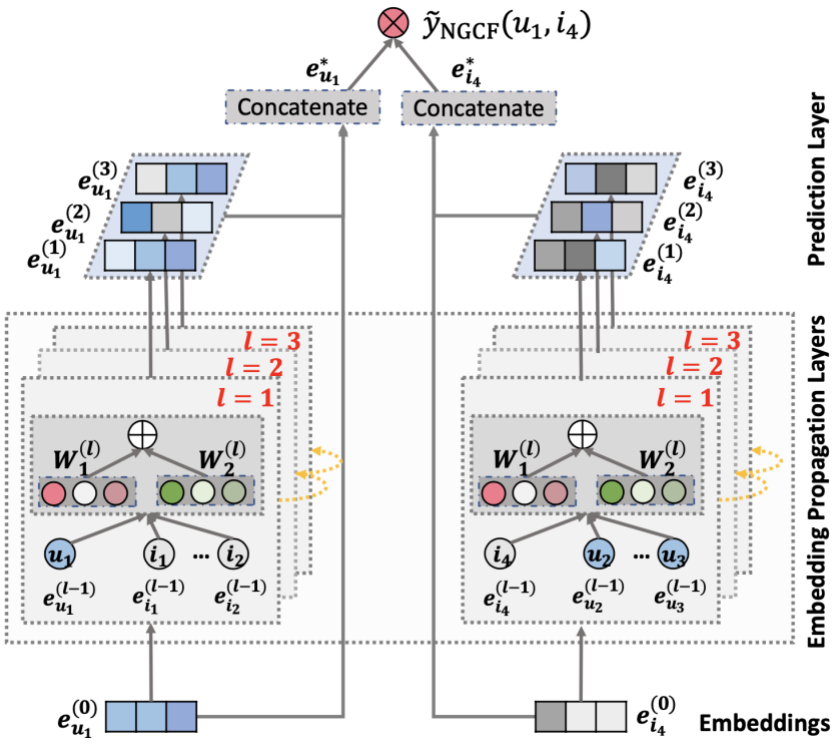
$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)}\| \cdots \|\mathbf{e}_u^{(L)}, \quad \mathbf{e}_i^* = \mathbf{e}_i^{(0)}\| \cdots \|\mathbf{e}_i^{(L)}$$

Finally, the user's interests in item can be represented as:

$$\hat{y}_{\mathrm{NGCF}}(u,i) = \mathbf{e}_u^{*\top}\mathbf{e}_i^*$$

- $e_u^{(0)} || \ldots ||e_u^{(L)}$: the representations at different layers
  • emphasize the messages passed over different connections
  • have different contributions in reflecting user preference

This layer is mainly used to iteratively calculate the relationships between users and items through the links between users and items. The following diagram shows how the multiple embedding propagation layers refine user u_1 and item i_4.



5 Preliminary performance analysis of algorithm/system

5.1 Dataset

[14] Expedia provides a dataset on Kaggle website so that we can use them for recommendation system research. This dataset contains customer behavior data, including which hotel the customer clicked on, whether the hotel was booked, and the details of hotels and users. Our objective is to analyze user behavior from historical data to predict which "hotel clusters" users are most likely to be interested in and book.

The Expedia hotel dataset is divided into training set and test set. The data in 2013 and 2014 is the test set, and the data in 2015 is the training set. The 2013 and 2014 data include all users' data, as well as click events and book events. However, the 2015 data only contains the book event data.

The following is the dataset overview with the chosen columns:

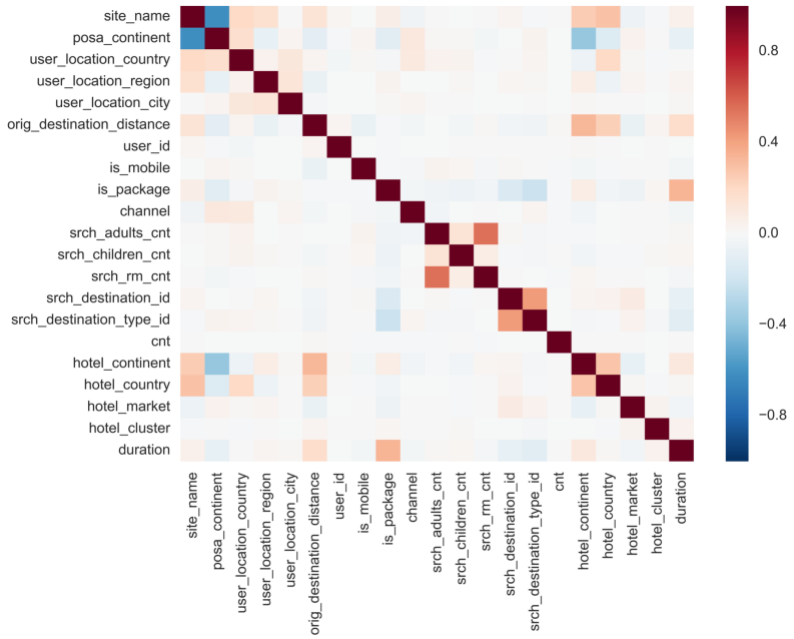
user_id	is_mobile	srch_destination_id	srch_destination_type_id	is_booking	hotel_continent	hotel_country	hotel_market	hotel_cluster
12	0	8250	1	0	2	50	628	1
12	0	8250	1	1	2	50	628	1
12	0	8250	1	0	2	50	628	1
93	0	14984	1	0	2	50	1457	80
93	0	14984	1	0	2	50	1457	21
93	0	14984	1	0	2	50	1457	92
501	0	8267	1	0	2	50	675	41
501	0	8267	1	0	2	50	675	41
501	0	8267	1	0	2	50	675	69
501	0	8267	1	0	2	50	675	70

5.2 Preliminary Analysis

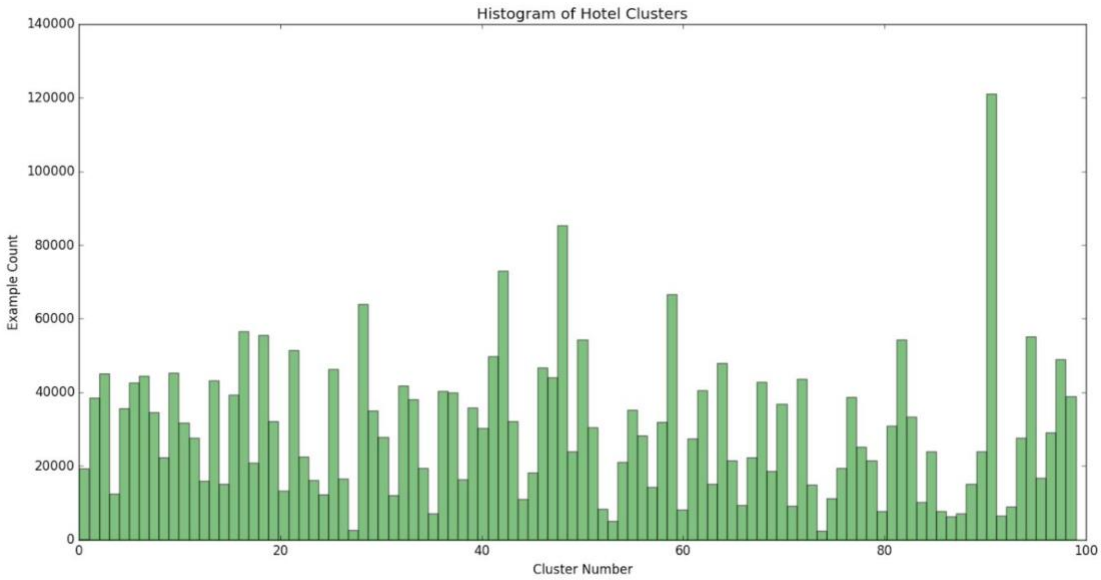
Though each transaction in the dataset has more than 20 features, Hotel cluster don’t have a strong correlation (positive or negative) with other features according to the diagram below. Therefore, methods which model linear relationships between features will not be very successful in this project. It’s a great signal that we may need to dig more non-linear relationships in the dataset.

It’s because the hotel cluster doesn’t have remarkable liner relationships with other attributes in this database that we can use NGCF algorithm to validate the effects of GNN for hotel recommendations.

The Correlation matrix of features are as follow:



The hotel clusters graph is as follow:



5.3 Data process and program design

5.3.1 Data process

By data preprocessing, we only retain [‘user_id’, ‘is_booking’, ‘hotel_cluster’] columns. We can find the interactions between users and items as follow:

```
train = pd.read_csv('train.csv',
                    dtype={'is_booking':int,'srch_destination_id':np.int32, 'hotel_cluster':np.int32},
                    usecols=['user_id','is_booking','hotel_cluster'], nrows = 1000000)
train=train[train['is_booking']==1]
```

User_id	Hotel_cluster
1048	36,58
893024	21, 41, 81, 44, 82, 89, 28, 96, 69, 31, 97, 95, 58, 42, 51
893032	13, 83, 17, 9, 50, 10, 98, 49
…	… …

	user_id	is_booking	hotel_cluster
1	12	1	1
20	756	1	2
27	1048	1	58
72	1048	1	36
79	1482	1	95
...	...	...	...
999977	893032	1	9
999979	893032	1	50
999983	893032	1	10
999985	893032	1	98
999996	893032	1	49

82096 rows × 3 columns

```
train.pivot_table(index=['user_id'],columns=['hotel_cluster'],values=['is_booking'])
```

	is_booking																				
hotel_cluster	0	1	2	3	4	5	6	7	8	9	...	90	91	92	93	94	95	96	97	98	99
user_id																					
12	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
114	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
190	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
192	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1198374	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1198524	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1198634	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1198650	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1198784	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

21905 rows × 100 columns

By data preprocessing, we only retain [‘user_id’, ‘is_booking’, ‘hotel_cluster’] columns. We can find the interactions between users and items as follow:

User_id	Hotel_cluster
1048	36,58
893024	21, 41, 81, 44, 82, 89, 28, 96, 69, 31, 97, 95, 58, 42, 51
893032	13, 83, 17, 9, 50, 10, 98, 49
…	… …

Train file-train.txt: Each line is a user with her/his positive interactions with Hotel_cluster (userID and a list of Hotel_cluster).
Test file-test.txt: Each line is a user with her/his positive interactions with Hotel_cluster (userID and a list of itemID).
Note that here we treat all unobserved interactions as the negative instances.

5.3.2 Program design

Environment Requirement:
The code has been tested running under Python 3. The required packages are as follows:
tensorflow == 1.8.0
numpy == 1.14.3
scipy == 1.1.0
sklearn == 0.19.1

Process:

In class NGCF(object):

1. Create Placeholder for Input Data & Dropout.
2. Compute Graph-based Representations of all users & items via Message-Passing Mechanism of Graph Neural Networks.
3. Establish the final representations for user-item pairs in batch.
4. Inference for the testing phase.
5. Generate Predictions

In '__main__':

1. Generate the Laplacian matrix, where each entry defines the decay factor (e.g., p_ui) between two connected nodes.
2. Save the model parameters.
3. Reload the pretrained model parameters.
4. Train the model.
5. Save the results.

5.4 Outcome

For each user in the dataset, all items that the user does not interact with are regarded as negative items. In order to evaluate the effectiveness of Top-k recommendation and optimization, we the widely used evaluation methods: recall rate. [15]Recall is the proportion of Real Positive cases that are correctly Predicted Positive. It can be used to represent how effectively our recommender system works.

The results are as follows:
n_users=21905, n_items=100

n_train= 82096, n_test=20987
recall=0.168

MAP@5 score from other methods, we refer to [16]:

Model Used	Motivation	Results
XG Boost	Performs very well (but difficult to tune)	0.25289
Naive Bayes	Simple and relatively fast model	0.02690
KNN	Simple and relatively fast model	0.12458
Factorization Machine/SVM	Works well with large datasets and categorical values	N/A
Random Forest (multiclass)	Deals well with uncorrelated or non-linear correlations	0.15624
Random Forest (binary)	This is the best ensembling method, easy to tune	0.28141
Logistic	To confirm that this method does not work well with uncorrelated data	0.10200
Baseline	Based on the 5 most probable classes	0.07213

From this result, we can find NGCF algorithm is relatively effective when processing the interaction data, but the recall rate is not high and we speculate the ignored attributes such as users' location and hotels' location take an important part in hotel recommendation system.

6 Conclusion and Future Work

6.1 Conclusion

This is the summary for this paper.

In the first part, I introduced the application and framework of recommender system, gave a brief introduction to GNN and the application of GNN for recommender system, the reason why I'd like to choose this topic, as well as a potential outcome.

In the second part, I introduced some related work about collaborative filtering algorithms which help us in recommender system field, such as deep matrix factorization model, graph convolutional matrix completion, neural collaborative filter. Then I compared the advantages and disadvantages of these three methods. Last, I analysed the relationship between the existing solutions and the proposed solution.

In the third part, system modelling and structure, I mainly introduced the proposed solution and justifications of the design choices and how the proposed method can address the

existing limitations. The main existing limitations are two, one is that embedding function only considers descriptive features, and the other one is that user-item interactions are not considered. That is the reason why I want to apply NGCF to hotel recommender system.

In the fourth part, I introduced the methodology and algorithm theories we used in the project, first I gave a brief introduction for the high-order connectivity, and then I introduced the NGCF framework. NGCF is a new recommendation framework based on graph neural network, explicitly encoding the collaborative signal in the form of high-order connectivity in user-item bipartite graph by performing embedding propagation.

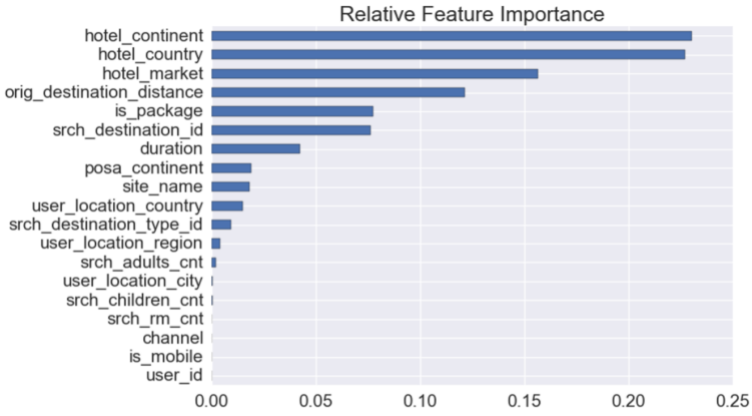
In the fifth part, I introduced the dataset we used in this project and preliminary analysis, to a large extent this dataset is suitable for us to apply NGCF and we want to dig out more user-item interaction information from the data. Then I introduce some steps when I do the data preliminary process and algorithm design. From the outcome, we may see the result is not as good as expected, but I still think I make a great progress in applying graph algorithm to hotel data to see how it comes with deep learning algorithms.

The seventh part in the paper is conclusion and future work. Of course, I still have more to do for this topic.

6.2 Future Work

1. Improve the recall rate

[17]There has been a study adopting these factors - service, price, facility ratings combined with the location of the customers to make recommendations so that the recommendations can be better to meet the customers' needs. It gives us inspiration if NGCF is more powerful when combined with location factors. In our project, we find the hotel_continent, hotel_country and hotel_market are three main features related to the users' choices. So maybe we can dig more information from hotels' locations to improve the recall rate.



2. Whether we can do more creative work

Except NGCF, whether we can apply other algorithms of graph neural network to hotel recommendations.

7 References

[1] M., Venkatesan & Thangadurai, K.. (2017). History and Overview of the Recommender Systems. 10.4018/978-1-5225-0489-4.ch004.

[2] Scarselli, Franco & Gori, Marco & Tsoi, Ah & Hagenbuchner, Markus & Monfardini, Gabriele. (2009). The Graph Neural Network Model. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council. 20. 61-80. 10.1109/TNN.2008.2005605.

[3] Xu, Keyulu & Hu, Weihua & Leskovec, Jure & Jegelka, Stefanie. (2018). How Powerful are Graph Neural Networks?.

[4] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun, "Graph Neural Networks: A Review of Methods and Applications", arXiv:1812.08434, vol. Dec, 2018.

[5] Snehal Reddy Koukuntla. (2019). *Evolution of Graph Neural Networks for Recommender Systems.* Towards Data Science. Retrieved from https://towardsdatascience.com/evolution-of-graph-neural-networks-for-recommender-systems-51e24d3b5bd4

[6] Cui, Zeyu & Li, Zekun & Wu, Shu & Zhang, Xiaoyu & Wang, Liang. (2019). Dressing as a Whole: Outfit Compatibility Learning Based on Node-wise Graph Neural Networks. 10.1145/3308558.3313444.

[7]Mavalankar, Aditi & Gupta, Ajitesh & Gandotra, Chetan & Misra, Rishabh. (2019). Hotel Recommendation System. 10.13140/RG.2.2.27394.22728/1.

[8] Xue, Hong-Jian & Dai, Xinyu & Zhang, Jianbing & Huang, Shujian & Chen, Jiajun. (2017). Deep Matrix Factorization Models for Recommender Systems. 3203-3209. 10.24963/ijcai.2017/447.

[9] Berg, Rianne & Kipf, Thomas & Welling, Max. (2017). Graph Convolutional Matrix Completion. arXiv:1706.02263v2

[10] He, Xiangnan & Liao, Lizi & Zhang, Hanwang. (2017). Neural Collaborative Filtering. Proceedings of the 26th International Conference on World Wide Web.

[11] Zhong, Kai & Song, Zhao & Jain, Prateek & Dhillon, Inderjit. (2018). Nonlinear Inductive Matrix Completion based on One-layer Neural Networks.

[12] Wang, Xiang et al. "Neural Graph Collaborative Filtering." Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR'19 (2019): n. pag. Crossref. Web.

[13] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. NIPS Bayesian Deep Learning Work- shop, 2016.

[14] Shenoy, Gourav & Wagle, Mangirish & Shaikh, Anwar. (2017). Kaggle Competition: Expedia Hotel Recommendations.

[15] Powers, David & Ailab,. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. J. Mach. Learn. Technol. 2. 2229-3981. 10.9735/2229-3981.

[16] Wesley Klock,.(Dec 16, 2018). *Expedia Hotel Recommendations*. Medum. Retrieved from https://medium.com/@wesleyklock/expedia-hotel-recommendations-ea6a9d5fbaa7

[17] Mavalankar, Aditi & Gupta, Ajitesh & Gandotra, Chetan & Misra, Rishabh. (2019). Hotel Recommendation System.