Dual Knowledge Multimodal Network for Recommender System

Jiahong Ouyang
Stanford University
jiahongo@stanford.edu

Yuqi Jin
Stanford University
yuqijin@stanford.edu

Xiaoqin Zhou
Stanford University
zhouxq@stanford.edu

Abstract

Yelp is a popular platform providing user reviews and recommendations for various recreation activities based on user-to-business review rated by the users. In this paper, we aim to work on the Yelp open data set, which contains not only the user-item interaction record, but also the reviews, item attributes, and pictures, to design a recommendation system with the help of the underlying knowledge graph. Our idea is that besides the basic user-item interaction properties we will utilize, we can also construct features for each node with the help of textual information from reviews, which increases the richness of features. We also plan to utilize multiple knowledge graphs to enrich the features. Our objective is to give a better recommendation for each user compared to the current existing benchmark.

1. Introduction

With the development of the internet, users are immersed in a vast amount of online content. A challenge with online platforms is how to alleviate the impact of overwhelming information. Thus, advanced recommender systems (RS) are needed to recommend a small set of items to meet users' personalized interests.

Personalized recommendation algorithms can be divided into three categories: content-based, user-item interaction-based, and hybrid method. Content-based methods try to learn the embeddings from the existing items and query items from items' textual [1] and visual information. User-item interaction-based methods model the information interchange between users and items. It can further divide into two subcategories: collaborative filtering based methods [5] and graph based methods [9, 3, 7, 8]. Hybrid methods combine the content of items with the user-item interactions [4, 10].

Our project is based on one of the state-of-the-art methods [8]. Our contributions can be summarized as:
(1) We adapted the baseline model to user-item interaction by rating scores instead of 0/1.
(2) We extracted textual features from reviews and integrated the features to update the user-item embeddings.
(3) We combined the user knowledge graph with a business knowledge graph and adapted the framework to incorporate dual knowledge graphs.

In section 2, we explain three recent works in detail. In section 3, we state the problem definition, the dataset, and the evaluation methods. In section 4, we articulate the construction of graphs and dataset analysis. In section 5, we introduce the preliminary method and our proposed improvements. In section 6, we present the results.

2. Related Works

In this section, we explain two graph-based methods and a hybrid method in detail. Section 1.1 is a method purely based on user-item interaction. Section 1.2 is knowledge-graph enhanced. Section 1.3 is a hybrid method using multi-modal contents.

2.1. Graph-based Recommendation Based on User-item Interaction

The modern high-quality recommendation systems require the recommendations should be both personalized and responsive in real-time, which can be difficult since the graph could be huge. To solve such problem, Pixie [3], a scalable graph-based real-time recommender system was proposed. Recommendations provided by Pixie can lead up to 50% higher user engagement compared to previous benchmarks with 60 ms latency.

Here, we think since the Pixie Random Walk algorithm does not build some model directly for an object, it is more like a collaboration filter algorithm. The clustering analysis is used here to find and return the closet content, i.e., the recommendation to the user. The real-time feature is achieved since there the machine does not need to study every time it makes a recommendation, which makes it very good for a large size recommendation system. However, in the meantime, the heavy-tailed degree will be a disadvantage. Another improvement we think might be feasible is the refresh cycle of the current graph. In the paper, the graph is updated once a day, but if a graph can be updated in a shorter period, even in real-time, then the recommenda-

tion qualities will be much better. Therefore, we may want to invest in GNN techniques to overcome the inherently inductive limitation.

2.2. Knowledge Graph Enhanced Recommendation

Unlike any traditional recommender algorithm, Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems (KGNN-LS) [8] introduces a novel method to capture personalized item embeddings without manual feature engineering. KGNN-LS not only achieved the state-of-the-start in several benchmarks but also introduced a way to train the model end to end. Comparing with traditional methods, it solves some of the key problems in the recommender system. For the cold start problem, since KGNN-LS introduced a knowledge graph and learned a user entity relation, any newly added item can be part of the knowledge graph and, therefore, can also be recommended to a user. The model is also interpretable because from A_u we can identify what entities and relations are essential to the user. Also, we can utilize the embedding to find similar users or items. However, there are several areas KGNN-LS has not yet explored. We could explore other loss functions other than cross-entropy loss that is given by the paper. We could also consider other user interactions other than clicks (ratings in our case). We could even try to inject multiple knowledge graphs if the knowledge bases are in different exclusive domains.

2.3. Graph and Content Hybrid Recommendation

Wei et al. [10] proposed a graph and content-based method for the micro-video recommendation, exploiting user-item interactions to guide the representation learning in multi-modalities. They designed a Multi-modal Graph Convolution Network (MMGCN) framework on the message-passing idea of graph neural networks to get modal-specific representations of users and items. Specifically, they built a user-item bipartite graph for each modality, and learn the representation of each node by the topological structures and the features of its neighbors. Their framework consists of three components: aggregation layers, combination layers, and prediction layer. In the aggregation layer, they employed mean and max aggregation functions to compute the user structural representation from its neighbors. In combination layer, they first integrated the user's structural representation, its modality representation, and then applied concatenation and element-wise combination function to integrate multiple modalities. In the prediction layer, they obtained the final user and item representation by the mean of the outputs from stacking layers.

We think this paper is a great example of combining structural representation learning from the user-item graph as well as the representation from multi-modality data. We also believe that including a knowledge graph in this framework can better characterize the relation of an item and its attributes.

3. Problem statement

3.1. Problem Definition

In the recommender system, we define the set of users $\mathscr{U}$ and the set of items (businesses) $\mathscr{V}$. The user-item interaction is defined by Y, where $y_u v \in [0, 5]$ denotes the user rating of the business.

We can also build the knowledge graphs $\mathscr{G}_i = (h, r, t)$ from the dataset, where $h \in \mathscr{E}$, $r \in \mathscr{R}$, and $t \in \mathscr{E}$ denotes head, relation, and tail in a knowledge triple respectively. $\mathscr{E}$ represents the set of the entities in the knowledge graph, while $\mathscr{R}$ represents the set of the relations between entities. In our case, for example, the triple can be (Restaurant A, locates, Palo Alto), (Restaurant A, has, 5 star), or (Restaurant A, category, Mexican). We plan to build multiple knowledge graphs, one around business and another around users.

In addition, other auxiliary information of the items can also be helpful in the recommendation. For example, the textual information from the reviews and tips, and also the visual information from the images of the item. We denote the auxiliary information from different modality as $\mathscr{R} = v, t$, where v and t represent images and reviews.

The goal of the project is to build a model to predict the user ratings u on item v, based on the user-item interaction matrix Y, the knowledge graph $\mathscr{G}_i$, and the auxiliary information $\mathscr{R}$. Mathematically, we are attempting to get the prediction:

$$\hat{y_{uv}} = \mathscr{F}(u, v; \Theta, Y, \mathscr{G}_1, , \mathscr{G}_2, \mathscr{R}) \tag{1}$$

Θ denotes all the parameters in function $\mathscr{F}$.

3.2. Data

We propose to build the restaurant recommendation system on Yelp open dataset, with 1,637,138 users and 192,609 businesses. It includes check-in, tips, reviews to characterize the user-item interactions. It also contains the user profile (friendship, fans, etc) and the business profile (location, food category, photos with its classification, etc).

Due to the size of the dataset, we limit the business to Las Vegas and filter out users that have no interaction with the selected business, which gave us 624,056 users, 29,369 businesses (items), and 2,030,798 reviews. To simplify the task, we only use reviews to get the user-item interaction matrix Y. For the knowledge graph, we limit the attributes to rating star level, price range, business properties, and business categories, which is easy to build on our own. We also include textual information from the review comments. Image features are expensive to compute due to limited resources; therefore we excluded them.

Since the dataset does not provide the training and testing set, we plan to split the dataset on our own. We randomly select P percent of the reviews to create the training set and the remaining (100-P) percent to create the test set.

3.3. Evaluation

We plan to evaluate our method in two parts: graph and recommendation

- Knowledge Graph
  We want evaluate to the properties of knowledge graph by plotting the degree distribution so that we could compare how different are our graph comparing with the graph in KGNN-LS [8]

- Recommendation
  We want to evaluate our model performance using classic recommender system metrics. Specifically, we will be looking at Root Mean Squared Error (RMSE).

  Random splitting can measure the predictive accuracy of a statistical model. But it may not be appropriate for estimating the expected performance of a recommendation model in a real system. Because training set and test set might cross each other, and it does not take temporal effect into considerations.

4. Graph Construction and Analysis

4.1. User-Business Interaction Bipartite Graph

We first analyze the dataset based on the user-business interaction graph. It is a bipartite graph with user nodes and business nodes. An edge between a user node and a business node represents the user gave a review for the business. A user can give multiple reviews for a single business. Each review consists of a paragraph of comments and a star from 1 to 5. The graph consists of 29,370 business nodes and 624,056 user nodes, with 2,030,798 edges.

Figure 1 shows the log-log degree distribution of user and business nodes. Most of the user and business gave/received less than 10 reviews, while the maximum number of reviews the user commented is 2,354 and the business received is 8,570. The analysis of the star distribution is provided in figure 2. Most of the reviews are 4 to 5-stars, holding a positive attitude.

4.2. Business Knowledge Graph

Yelp dataset provides rich information about the businesses. Those businesses includes: restaurants, barber shops, gyms and so on. After preliminary investigation, we came up with four different relations with (*head.relation.tail*) triplet format in table 1 and a visualization is shown in figure 3:

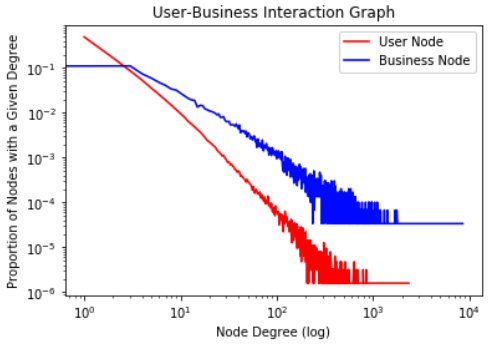

Figure 1. User-Business Interaction Graph

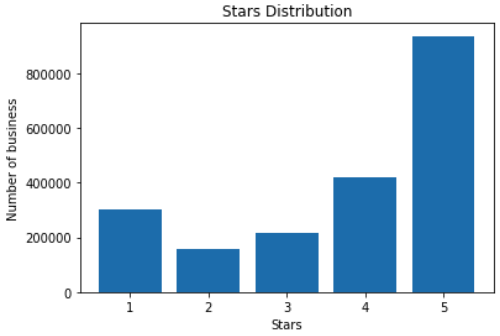

Figure 2. Stars Distribution

head	relation	tail
business	hasNumReviews	numReviews
business	hasStars	stars
business	isCategory	category
business	hasAttribute	attribute

Table 1. Business Knowledge Graph relation tuples.

The business knowledge graph contains 3,0580 entities with 25,6095 relation triplets. We can tell that most businesses contain about less than 12 relations based on the log-log out-degree distribution of the knowledge graph in figure 4. And we can also tell from the in-degree distribution of the knowledge graph that most entities shares 100 to 1000 businesses in figure 5.

4.3. User-User Friendship Graph

In the Yelp dataset, a user's friendship information is also included. We believe the user-user friendship graph can provide useful information for business recommendation. For example, users from the same country tends to be friends and like same type of foods. Also, a user with many friends may have a strong influence on its fans' choices.

In the user-user friendship graph, each node represents a user and an edge denotes the friendship between the two connected users. It consists of 111,304 nodes and 77,699 edges. Due to the limited size of the dataset, the friendship

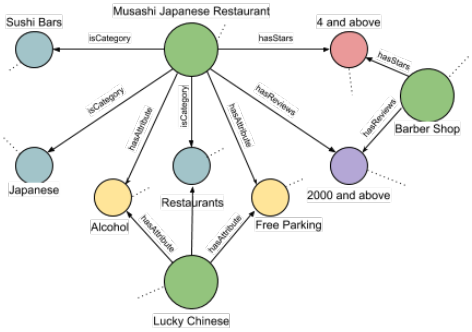

Figure 3. Business Knowledge Graph

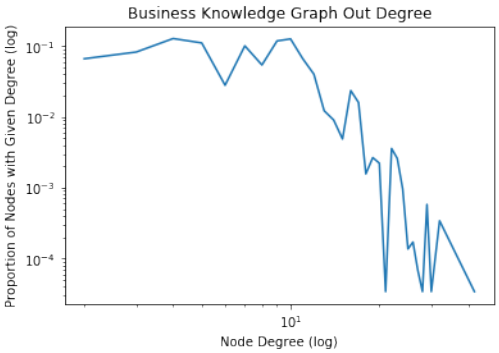

Figure 4. Business Knowledge Graph out Degree

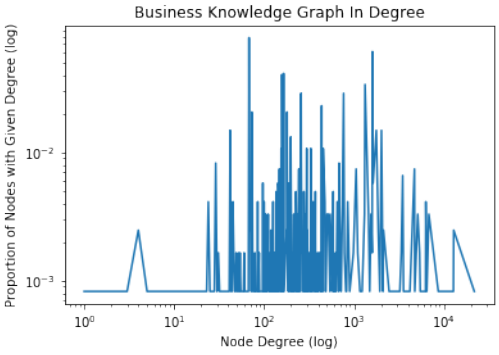

Figure 5. Business Knowledge Graph in Degree

graph is very sparse, with maximum degree 1. Due to the sparsity of the user friendship graph, it may be hard to push forward the idea of combining friendship information.

5. Methods

5.1. KGCN Basline Model

Our GNN baseline model was adopted from KGCN [9]. KGCN was proposed to capture high-order structural proximity among entities in a knowledge graph. Consider a candidate pair of user $\mathbf{u}$ and item (entity) $\mathbf{v}$. We used $N(\mathbf{v})$ to denote the set of entities directly connected to $\mathbf{v}$, and $\mathbf{r}_{e_i,e_j}$ to denote the relation between entities $\mathbf{e}_i$ and $\mathbf{e}_j$. We also used a function $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ (e.g., inner product) to compute the score between a user and a relation:

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

The user-relation score can be normalized by:

$$\tilde{\pi}^u_{r_{v,e}} = \frac{exp(\pi^u_{r_{v,e}})}{\sum_{e \in N(v)} exp(\pi^u_{r_{v,e}})}$$

Then the item vector can be characterize by its neighborhood by:

$$\mathbf{v}^u_{N(v)} = \sum_{e \in N(v)} \tilde{\pi}^u_{r_{v,e}} \mathbf{e}$$

In the end, the entity vector was integrated with its neighborhood vectors $\mathbf{v}^u_{S(v)}$ by aggregators $agg : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$:

$$agg_{sum} = \sigma(\mathbf{W} \cdot (\mathbf{v} + \mathbf{v}^u_{S(v)}) + \mathbf{b})$$

$$agg_{concat} = \sigma(\mathbf{W} \cdot concat(\mathbf{v}, \mathbf{v}^u_{S(v)}) + \mathbf{b})$$

$$agg_{neighbor} = \sigma(\mathbf{W} \cdot \mathbf{v}^u_{S(v)} + \mathbf{b})$$

where $S(v)$ is the sample set from v's neighbors with a fix size of K, σ is ReLU. KGCN can be extended to multiple layers to explore neighbors several hops away and the final H-order entity representation is denoted as $\mathbf{v}^u$. The predicted score can be represented by:

$$\hat{y}_{uv} = 5 \cdot sigmoid(\mathbf{u} \cdot \mathbf{v}^u)$$

The loss function can be represented by:

$$\mathcal{L} = \sum_{u,v} J(y_{uv}, \hat{y}_{uv}) + \lambda R(A) + \gamma \|\mathcal{F}\|_2^2$$

where J is MSE loss, $R(A)$ is label smoothing loss (in detail next section), $\|\mathcal{F}\|_2^2$ us L2-regulizer.

5.2. Apply Label Smoothing

Inspired by the fact that Knowledge Graphs(KG) capture structured information and relations among entities, which is helpful for building a recommender system, the Knowledge-aware Graph Neural Networks with Label Smoothness regularization (KGNN-LS) [8], which extends GNNs architecture to knowledge graphs has been applied to our data set.

Since we used the star review prediction, which ranges from 0 to 5, to evaluate the results, we need to enable the network to work on the regression task instead of the original binary classification problem. The main modification we made here is regarding the loss function. For the original network, the loss function is composed of three parts: base loss function, L2 loss function and the label smoothness loss function.

We applied RMES as the base loss function to evaluate the performance of the model. We also maintained L2-regularizer as a part of the loss function. For the label smoothness feature mechanism introduced in KGNN-LS [8], it can solve the problem of over-fitting since previously, the user-item interactions are the only source of supervised signal. We observed that by adding the label smoothness feature, the final performance can be enhanced. Moreover, regarding the label smoothness mechanism, we applied the leave-one-out method introduced in KGNN-LS [8] to perform the label propagation, and set the default value for the label as the media of the review range, which equals to 3.0. The label smoothness loss function can be expressed as

$$\mathcal{R}(A) = \sum_u \sum_v J(y_{uv}, \hat{l}_u(v))$$

where J is MSE loss.

5.3. Incorporate Textual Review Features

In the reviews, the text comments can possibly convey more information than a single star level, such as the reason that the user gave this star or what the user most care about. Here, we extracted textual features from the reviews by BERT [2].

For each paragraph of text, we first transformed it into tokens using WordPiece embeddings and then translated tokens into a vector as input to the model. We limit the maximum sequence length of tokens as 128. We then utilized a pretrained cased 12-layers BERT model to extract 768-dimensional features from the last layer.

To show the representative of the extracted features, we divided the features into two clusters by K-Means and showed it in 2d t-SNE plot. We vaguely split the reviews into positive (4-5 stars), negative (1-2 stars), and medium attitude (3 stars). As shown in figure 6, dark green and dark red represents true positive and true negative data points, while light green and light red denotes the misclassified data points. The distinct two clusters and the majority of the correct classified points proved the potential of the textual features.

The extracted textual features $\mathbf{t}_{uv}$ (given by user u to business v) were then integrated to the framework. We designed an "attention" module to reweight each dimension of user vector $\mathbf{u}$ and item vector $\mathbf{v}^u$ given by each review between them. We also normalized the updated user/item vector to prevent them from vanishing along iterations.

$$\mathbf{a}_v^u = \sigma(\mathbf{W}_u \cdot \mathbf{t}_{uv} + \mathbf{b}_u)$$
$$\mathbf{a}_u^v = \sigma(\mathbf{W}_v \cdot \mathbf{t}_{uv} + \mathbf{b}_v)$$
$$\mathbf{u} = \frac{\mathbf{u} \circ \mathbf{a}_v^u}{\|\mathbf{u} \circ \mathbf{a}_v^u\|}, \mathbf{v}^u = \frac{\mathbf{v}^u \circ \mathbf{a}_u^v}{\|\mathbf{v}^u \circ \mathbf{a}_u^v\|}$$

where $\circ$ is element-wise multiplication. $\mathbf{a}_v^u$ and $\mathbf{a}_u^v$ are attention for user and vector got from a single review.

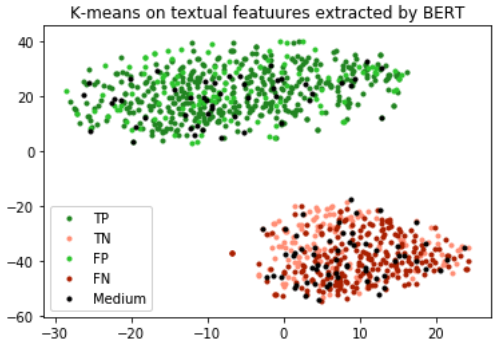

Figure 6. Texual features analysis.

$\mathbf{W}_u, \mathbf{b}_u$ and $\mathbf{W}_u, \mathbf{b}_u$ are transformation weight and bias for user and item vector respectively. σ is sigmoid function to normalize the attention between 0 and 1.

5.4. Integrate Dual Knowledge Graphs

Incorporating business KG in the GNN has given us a good performance improvement. It allows us to capture user-entity relationships and also considered the neighbors of the entities. What if we can also do something similar by incorporating the user-user relationship? Although we are not able to construct a user-user friendship graph due to the limitation of the Yelp dataset, we can use a similar method to build a user knowledge graph (user KG). In the Yelp dataset, the user profile data has many attributes that can be used to link different users together. The table 2 below is a list of relation tuples we used to construct the user knowledge graph. Figure 7 demonstrates an example of how the user knowledge graph looks. In analogy to the classical matrix factorization method, you can think of the business KG captures the item-item similarity, and the user KG captures the user-user similarity. The user KG contains 653,458 entities with 4,835,355 relation triplets. We can tell that most users contain about less than 13 relations based on the log-log out-degree distribution of the knowledge graph in figure 8. Notice that some user has more than 1,000 relations. This aligns with some user has reviewed more than 1,000 businesses. We can also tell from the in-degree distribution of the knowledge graph that most entities shares 500 users in figure 9.

Similarly from 5.1, we can also define $N(\mathbf{u})$ to be the set of entities directly connected to $\mathbf{u}$, and $\mathbf{r'}_{e'_i,e'_j}$ to denote the relation between entities $\mathbf{e}'_i$ and $\mathbf{e}'_j$ in the user KG. We can use a similar scoring function such that $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ to compute the score between a user and a relation:

$$\pi_{r'}^u = g(\mathbf{u}, \mathbf{r}')$$

head	relation	tail
user	isFriendwith	user
user	hasFans	fanCount
user	hasReviewedBusiness	business
user	isElite	eliteYear
user	hasReviewCount	reviewCount
user	hasAvergaRating	averageRating
user	hasCompliment	Compliment

Table 2. User Knowledge Graph relation tuples.

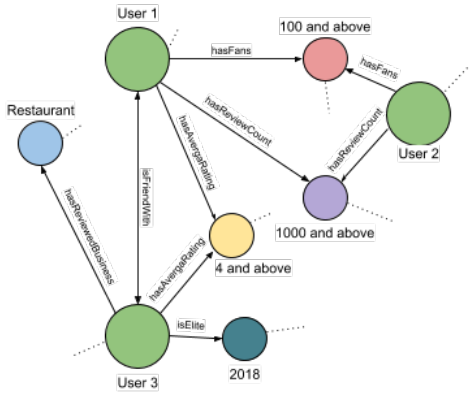

Figure 7. User Knowledge Graph

Therefore the user vector can be characterize by:

$$\mathbf{u}^u_{N(u)} = \sum_{e \in N(u)} \tilde{\pi}^u_{r_{u,e'}} \mathbf{e}'$$

In addition the aggregation functions can be modified into:

$$agg_{sum} = \sigma(\mathbf{W}_1 \cdot (\mathbf{v} + \mathbf{v}^u_{S(v)}) + \mathbf{W}_2 \cdot (\mathbf{u} + \mathbf{u}^u_{S(u)}) + \mathbf{b})$$

$$agg_{concat} = \sigma(\mathbf{W} \cdot concat(\mathbf{v}, \mathbf{v}^u_{S(v)}, \mathbf{u}, \mathbf{u}^u_{S(u)}) + \mathbf{b})$$

$$agg_{neighbor} = \sigma(\mathbf{W}_1 \cdot \mathbf{v}^u_{S(v)} + \mathbf{W}_2 \cdot \mathbf{u}^u_{S(u)}\mathbf{b})$$

and finally our predicted store is:

$$\hat{y}_{uv} = 5 \cdot sigmoid(\mathbf{u} \cdot \mathbf{v}^u + \mathbf{u} \cdot \mathbf{u}^u)$$

6. Results

6.1. Experimental Setting

We did a grid search for some hyperparameters. We chose the parameters setting: dimension of user/item embedding vectors as 14, number of neighbor samples as 8, number of iterations computing entity representation as 2, weight for label smoothing loss as 0.5, batch size as 6553. If using dual knowledge graphs, we set the number of epochs as 20, learning rate as 5e-4. If not, we set the number of epochs as 10, learning rate as 5e-3. We also experimented with different numbers of hops in the neighbor aggregations. We trained the model with 1, 2, and 3 hops but didn't notice any performance difference.

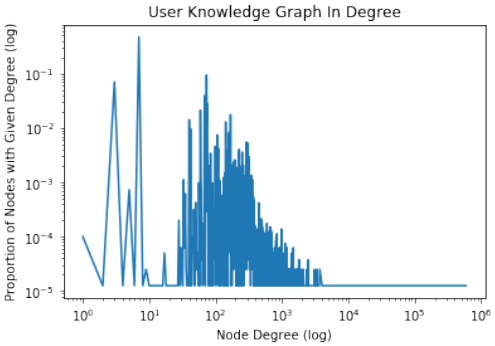

Figure 8. User Knowledge Graph out Degree

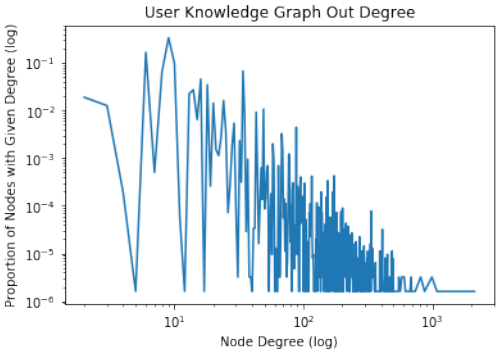

Figure 9. User Knowledge Graph in Degree

6.1.1 Impact of dimension of embedding

We used the grid search to find the optimal hyperparameters. Among the different parameters, the dimension of embedding generates the biggest impact. The result is shown in Table 3 is derived in a simple test scenario that the text review features have not been added, and only the business knowledge graph is used. But the result is intuitive that at first, we get better rmse as dimension size (d) increases since more information is encoded, while then rmse will become worse because of the over-fitting problem.

d	rmse	d	rmse
3	1.6167	13	1.5301
5	1.5739	14	1.5298
7	1.5540	15	1.5299
10	1.5375	20	1.5337
11	1.5347	25	1.5397
12	1.5314	30	1.5503

Table 3. RMSE with different dimension of embedding

Methods	BL-MF	BL-GNN	I1	I2	I3	I1+I2+I3
Rating Score RMSE	3.875	1.591	1.529	1.522	1.517	1.531

Table 4. Test rating score RMSE for baseline methods and proposed methods.

6.2. Quantitative Analysis

The figure 10 shows the train and test RMSE losses for each methods along epochs.

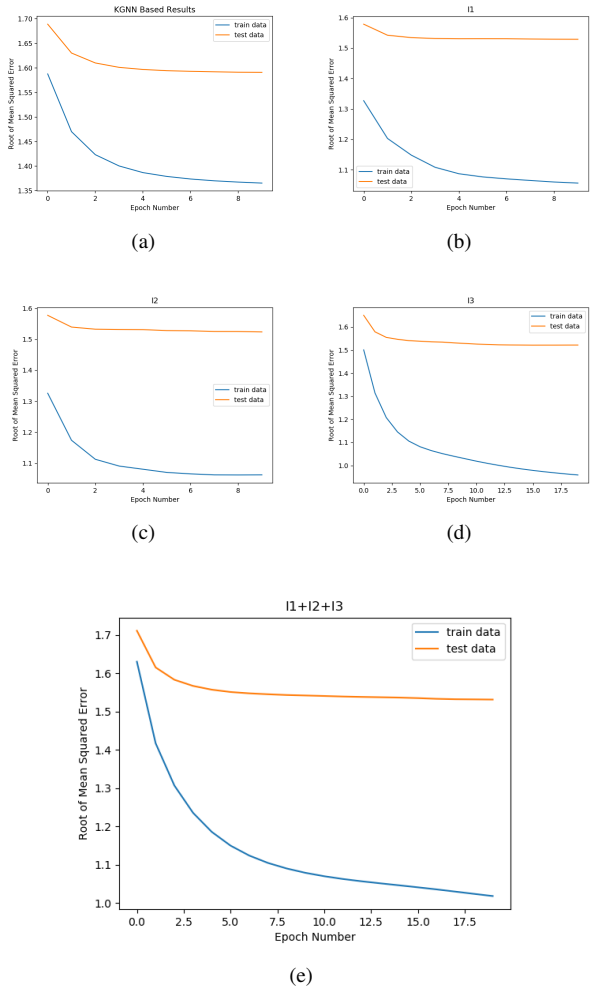

Figure 10. (a)BL-GNN: GNN-based baseline. (b) I1: GNN-based method with label smoothing improvement. (c) I2: GNN-based method with textual features. (d) I3: GNN-based method with dual knowledge graph. (e) I1+I2+I3: Proposed.

The table 4 shows the best test rating score RMSE. Here, we also implemented a classic version of the matrix factorization algorithm [6] as a baseline to compare against. We set the latent embedding dimension to 20, and we got a Root Mean Squared Error (RMSE) of 3.875.

Comparing to the traditional method, GCN based baseline method outperformed by 59%. Comparing with the GCN baseline, the proposed improvements all slightly improved the results by 3.9%, 4.3%, and 4.7%, respectively. Although the results are improved, we saw some over-fitting in the training set comparing to the baseline GCN. The results from the integrated model with all three improvements only improved the GCN baseline by 3.8%. We think it might because of the multiple updates of user and item vectors by improvement 2 and 3. It also might be caused by over-fitting as we were using a more complex model. Besides, due to the large number of hyperparameters, we think the integrated model might be able to achieve better results under other settings.

7. Conclusion

In this project, we presented a novel method of building a recommender system using multiple knowledge graphs and text features. We started by formulating the problem into a recommender system settings. Then we modified KGNN-LS[8] method to be our baseline method on the Yelp dataset. By building a GNN on top of a business KG and user KG with user reviews, we were able to achieve better results than the baseline GNN.

In the future, the project might also be improved in the following ways. Choosing better hyperparameters is significant, especially in a complex model. Also, as we mentioned in result analysis, a better way to update the user/item embeddings when using dual knowledge graph and textual features should be helpful. Finally, we could also explore having different hops for user KG and business KG when aggregating the neighboring features.

8. Work Split-Up

Yuqi Jin: Setting-up GNN baseline model, Improvement 1: label smoothing, hyperparameter grid search, writing report.
Jiahong Ouyang: User-business interaction bipartite graph analysis, Improvement 2: textual review features, integrating three improvements, writing report.
Xiaoqin Zhou: Building knowledge graphs, Setting-up matrix factorization baseline, Improvement 3: dual knowledge graph, writing report.

References

[1] T. Chen, L. Hong, Y. Shi, and Y. Sun. Joint text embedding for personalized content-based recommendation. *arXiv preprint arXiv:1706.01084*, 2017. 1

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 5

[3] C. Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich, and J. Leskovec. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 World Wide Web Conference*, pages 1775–1784. International World Wide Web Conferences Steering Committee, 2018. 1

[4] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017. 1

[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017. 1

[6] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009. 7

[7] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 417–426. ACM, 2018. 1

[8] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. *arXiv:1905.04413*, 2019. 1, 2, 3, 4, 5, 7

[9] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo. Knowledge graph convolutional networks for recommender systems. In *The World Wide Web Conference*, pages 3307–3313. ACM, 2019. 1, 4

[10] Y. Wei, X. Wang, L. Nie, X. He, R. Hong, and T.-S. Chua. Mmgcn: Multi-modal graph convolution network for personalized recommendation of micro-video. 2019. 1, 2