Transfer and Meta-Learning in Graph Neural Networks: A Recommender System Approach

Hamdy Hamoudi
Computer Science Department
(Stanford University)
hhamoudi@stanford.edu

Rouven Spiess
Computer Science Department
(Stanford University)
rouvens@stanford.edu

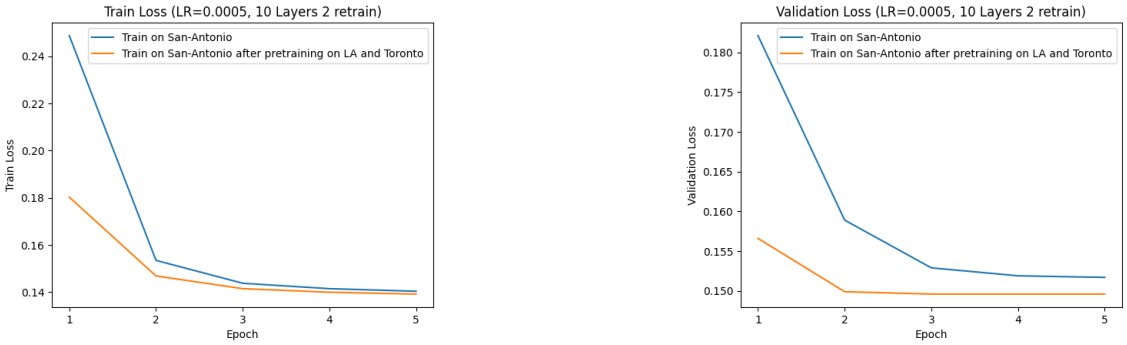
Megan Backus
Electrical Engineering Department
(Stanford University)
backusm@stanford.edu

I. Extended Abstract

In this project, we consider the task of restaurant recommendation, where the goal is to develop effective city-specific recommender systems based on user-restaurant interactions. We utilize a large collection of restaurant reviews scraped from Google Reviews to explore several approaches to training effective graph neural network-based restaurant recommendr systems. Given that user-restaurant interactions are naturally represented as a graph, we construct graphs from our Google Reviews dataset, where restaurants and users are represented as nodes and reviews are represented as links. We use graph neural networks as our recommendation system models to capture higher-order connectivities in the user-restaurant interaction network.

In this research, we explore the capability of transfer and meta-learning approaches to utilize user-restaurant interaction data from other cities in training a more effective model for a specific city, especially cities with sparse data. We experiment with transfer learning approaches with the goal of created an effective pretrained model using large quantites of data from major cities that can be finetuned on review data from any city to create a restaurant recommender for that specific city, including cities with sparse restaurant review data. We also look at meta-learning approaches and explore model architectures that meta-learn how to train a restaurant-recommending graph neural network for any city.

We first show that a LightGCN model trained on two cities with a joint loss function is able to give better predictions than if the model was trained individually. We then use the model in a transfer-learning setting, first by pre-training on one large city, then fine-tuning on another large city and on a smaller city by training only the 2 last GCN-layers. This achieves better results and converges much faster as compared to training on the smaller city directly without pre-training. We find that a 10 layer GCN-model performs better than a 5 layer GCN-model.



Train Loss Validation Loss

Fig. 1: San Antonio, baseline and after pre-training and fine-tuning

We then experiment with different meta-learning architectures, adapted for the graph domain. We implement a MAML architecture, and a modified MAML architecture that includes a graph signature function to effectively meta-learn how to train an effective link prediction variational graph autoencoder for predicting if a user is likely to have visited a restaurant. In this approach we treat the recommendation task for each city as a separate task, and meta-learn over a collection of cities. We show that these meta-learning approaches are more effective than simply training a link prediction model for each city, with the two meta-learning approaches (the MAML and extended MAML models) outperforming our baseline models by 22% and 29%, respectively when using average precision as an evaluation metric.

II. Introduction

While many current recommendation systems use matrix factorization to generate collaborative filtering-based user recommendations, we specifically look at the representation of user-restaurant interactions as a graph, and explore graph neural network-based methods to perform recommendation tasks as a special case of collaborative filtering.

Our goal is to create recommender systems that learn from users' restaurant ratings and predict the best set of restaurants, to recommend to a particular user. This can be represented as a link prediction, or a weighted link prediction task (where the link weights are the restaurant ratings). We utilize graph neural network-based methods to approach this prediction task and experiment with how transfer learning and meta-learning techniques can allow for data from a variety of geographical locations to be utilized in training a model specific to one location.

In this project, we utilize a custom dataset of Google Review data. We collected a total of more than 5 million unique restaurant ratings from 8 cities of varying sizes that we obtained by web scraping Google Maps for restaurant reviews. Each review consists of a user ID, a restaurant ID, a rating (1-5), and review text. Below is a description of the dataset.

City	Number of ratings	Number of Restaurants
Los Angeles	1,295,795	9,521
Toronto	1,285,821	7,991
Chicago	1,071,137	5,580
San Antonio	986,335	3,410
Austin	533,245	2,397
Seattle	236,172	1,689
Atlanta	256,897	1,676
Detroit	138,849	995

TABLE I: description of the dataset

The data is best represented as a bipartite graph with two disjoint sets represented by users and restaurants.

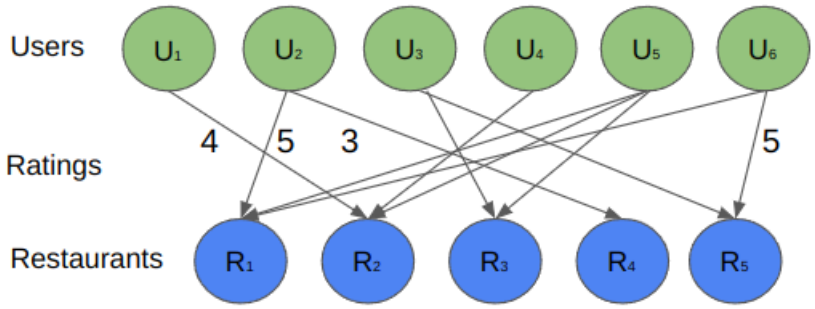


Fig. 2: A graph representation of the dataset.

Given a set of user nodes that correspond to a set of a held-out edges, the goal of a local link or rating prediction model for one city is to predict the a set of edges and/or weights for the given nodes.

The premise of this research is that we view each graph (city) as a separate task due to differences in user behaviours, the existence of discrete geographical user-restaurant graphs (there is no overlap in users or restaurants between cities), and the practical applications of developing city-specific recommender systems.

Our overarching goal in this project is to experiment with different architectures to see if we can effectively utilize the restaurant-user interaction patterns of other cities in developing and training an effective restaurant recommendation model that is specific to one city.

We first use a multi-task learning approach on two graphs, each representing one city, to develop a base 'pre-trained' model that will then be applied to transfer learn to new cities that have a more limited number of ratings, in order to speed up the learning process and significantly improve prediction accuracy.

We then experiment with different meta-learning architectures to develop a model that effectively learns to train a recommender system for any city.

III. Related work

Recent progresses from transfer learning in Natural Language Processing could be adapted for making recommendations [6]. The fine-tuning of transformer based models as BERT was first trained in text classification, and then used as basis for user vector embeddings. The users are first grouped given their rated items, and then embeddings of the users are learned by their rating levels using gradient descent. Other approaches than the embeddings method were also proposed, including matrix factorization, binary recommendation and multi-level recommendation.

There has also been some previous work done in researching the effectiveness of applying GNNs to recommendation tasks. In Graph Neural Networks in Recommender Systems: A Survey [1], the authors provide a comprehensive review of GNN based recommender systems and conclude that the use of GNNs has become widespread given that "information in recommender systems essentially has graph structure and GNN has superiority in graph representation learning".

Previous work has been done to leverage meta-learning techniques in graph problems by applying them to GNNs, as outlined in [7]. In particular, previous works have applied meta-learning approaches to GNNs, and have used several such approaches to solve various node classification, link prediction, and graph classifcation tasks.

While there has been some previous exploration of applying meta-learning approaches to various tasks in the graph domain, our work differs in attempting to take a similar high-level idea and applying it to a recommendation system. The task of utilizing meta-learning for a graph-based recommender system is not widely explored, and we hope to build off previous related works in focusing on this specific task. We also apply these GNN-based approaches in combination with transfer and meta-learning on a unique dataset we curated ourselves from scraping Google Reviews data.

IV. Method in detail

A. Transfer learning Approach

In our first approach we experiment with the use of transfer learning. We aim to train a base model on a large amount of data that can then be leveraged to learn new tasks with far more limited or sparse data while maintaining a high level of accuracy.

We first implement a LightGCN; a simplified graph neural network that achieves feature smoothing on neighbourhood aggregation for collaborative filtering as described in [1]:

$$e_u^{(k+1)} = AGG(e_u^{(k)}, \{e_i^{(k+1)}, i \in \mathbb{N}_u\})$$

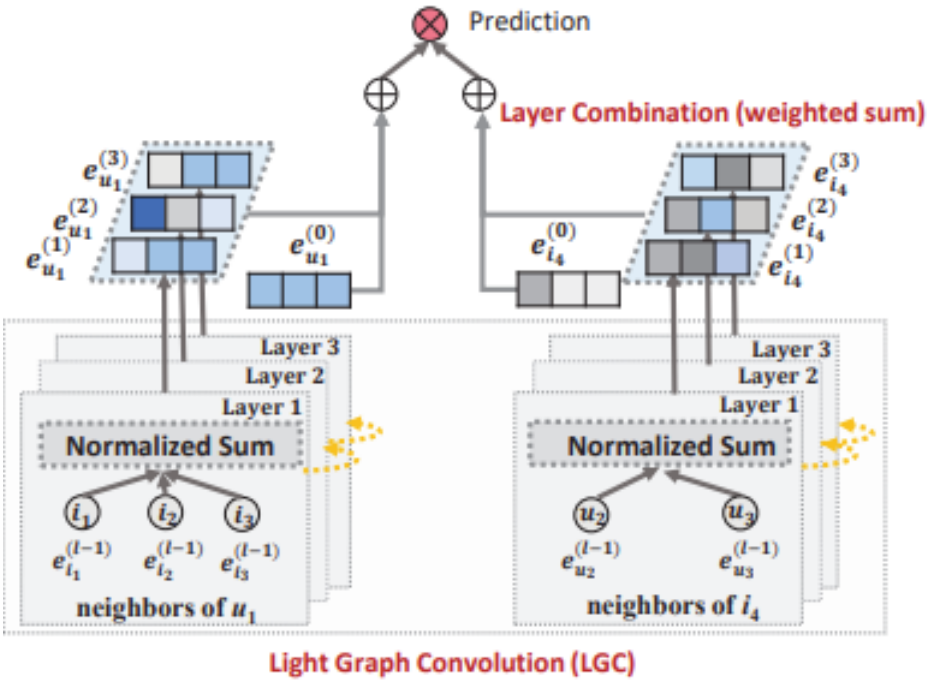


Fig. 3: LGC

In this approach, we are solving a restaurant rating prediction task. We score the accuracy of the model using the Root of the Mean Square Error (RMSE) between the predicted and the actual rating. We use RMSE, as apposed to Mean Absolute Error, as it penalizes for larger errors disproportionately, which is a design choice that we make to improve the quality of the recommendations.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(rating_i - \hat{rating_i}\right)^2}$$

The model then predicts user ratings by calculating the inner product of the final user and restaurant (item) embeddings, computing the similarity between them:

$$\hat{y}_{ui} = {e_u}^T e_i$$

In order to create our 'base-model', we first train a model with a LightGCN architecture over a superset of data that contains restaurant ratings from the two largest city datasets (LA and Toronto). In this model architecture the high-order connectivity between nodes is equivalent to the number of convolutional layers we set in the model parameters. We train our model with 10 convolutional layers.

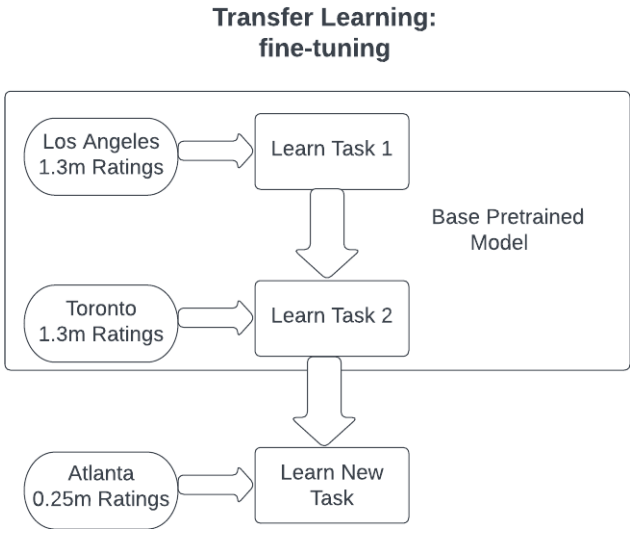


Fig. 4: Our Transfer Learning methodology

After we obtain our pre-trained base-mode, we then transfer learn from the base model to new tasks of cities with smaller datasets. We experiment with parameter freezing and retraining the different layers and compare the results to find the model that performs better in terms of both converging quicker (lower number of epochs) as well as achieving a lower train and validation loss.

B. Meta-Learning Approach

We experiment with the use of meta-learning architectures to perform the task of link prediction over restaurant-user graphs for multiple cities. In this approach, we somewhat simplify our target task from rating prediction (as in our transfer learning approach) to link prediction, predicting the probability of a link between a user and a restaurant (i.e., how likely it is that a particular user would visit a particular restaurant).

The goal of our meta-learning methodology is to learn a meta link prediction model from a set of training graphs $G_i \sim p(G)$ (where each graph in the set of training graphs is a user-restaurant interaction graph for one city) such that this meta model can be used to learn an link prediction model for a new graph that is more effective and requires less training time and training data. This model has the goal of learning an effective parameter initialization for a local link prediction model.

In our setup, we generate a set of graphs G from our Google Reviews restaurant rating data. Each distinct user and each distinct restaurant are represented by one node in the graph, and the set of edges in the graph represents which users have reviewed which restaurants. Here we make the assumption that reviewing a restaurant is equivalent to having visited that restaurant and vice versa. The total set of edges in each graph is divided into a train and test set. The meta-train set is then comprised of the entire set of graphs (each one corresponding to one city) with the exception of one held-out graph, which is used as the meta-test set.

In our approach, we utilize model architectures that adapt classical gradient-based meta-learning to the graph domain. In the context of data that can be represented as graphs, we consider the distribution over graphs as the distribution over tasks from which our meta-learning model learns a set of global initialization parameters.

In our meta-learning approach, we implement two model architectures:

1.) Gradient-based MAML
2.) Meta-Graph [4]

1) Local Link Prediction Model: In both architectures, a variational graph autoencoder (VGAE) is used as the local link prediction model, chosen due to its strong performance on local link prediction benchmarks in other work [3]. The VGAE link prediction model is defined as follows:

Given a graph $G_i = (V_i, A_i, X_i)$ where A_i is an adjacency matrix for edges in the graph G_i. The VGAE learns an inference model $q_\phi(Z|A, X)$ that defines the distribution over node embeddings. Specifically, each row of Z is a node embedding that is used to score the likelihood of a edge between node pairs. The inference model is defined as:

$$q_\phi(z_v|A, X) = \mathcal{N}(z_v, \mu_v, diag(\sigma_v^2))$$

, where $z_v \in R^d$ is one row of Z. The parameters of this model are learned via GNNs:

$$\mu = GNN_\mu(A, X)$$
$$log(\sigma) = GNN_\sigma(A, X)$$

The likelihood of an edge existing between two nodes is then defined as:

$$p(A|Z) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(A_{u,v}|z_u, z_v)$$

where

$$p(A_{u,v}|z_u, z_v) = \sigma(z_u^T z_v)$$

The inference GNNs are trained to minimize the variational lower bound on the training data:

$$\mathcal{L}_\mathcal{G} = E_{q_\phi}[\log p(A^{train}|Z)] - KL[q_\phi Z|X, A^{train})||p(z)]$$

2) MAML Model: We utilize an adaptation of MAML [2] to the link prediction setting as one meta-learning approach. With this approach, we optimize a shared parameter initialization θ that is used to initialize all the parameters for the VGAE inference GNNs. The global parameters θ are optimized using second-order gradient descent.

3) Meta-Graph Model: We also experiment with a second meta-learning architecture, Meta-Graph, proposed by [4] for a variety of other link prediction tasks. The Meta-Graph model can be viewed as an extended version of the adapted MAML model where in addition to learning a shared parameter initialization θ, a graph signature $\psi(\mathcal{G}_i)$ is also learned.

The graph signature $\psi(\mathcal{G}_i)$ has the purpose of capturing graph information from the history of seen training graphs, and using this learned signature to modulate the parameters of the VGAE inference model. That is, the inference model $q_{\phi i}$ for each graph can be conditioned on the graph signature $q_{\phi i}(Z|A, X, \psi(\mathcal{G}_i))$. The graph signature function used to compute the graph signature is implemented as a 2-layer GCN with sum pooling:

$$\psi(\mathcal{G}) = MLP(\sum_{v \in \mathcal{V}} z_v)$$
$$Z = GCN(A, X)$$

In addition to the shared initialization parameter θ the graph signature $\psi(\mathcal{G}_i)$ is also learned via second-order gradient descent. The full architecture of the meta-graph model is visualized in Figure 6.

The Meta-Graph algorithm is run in the following way:

1) a batch of training graphs is sampled
2) the local link prediction VGAEs are initialized for the training graph batch using θ and $\psi(\mathcal{G}_i)$
3) K steps of gradient descent are run to optimize each of these VGAE models
4) second-order gradient descent is used to update θ and $\psi(\mathcal{G}_i)$ based on validation edges

The full algorithm as proposed by [4] is shown below:

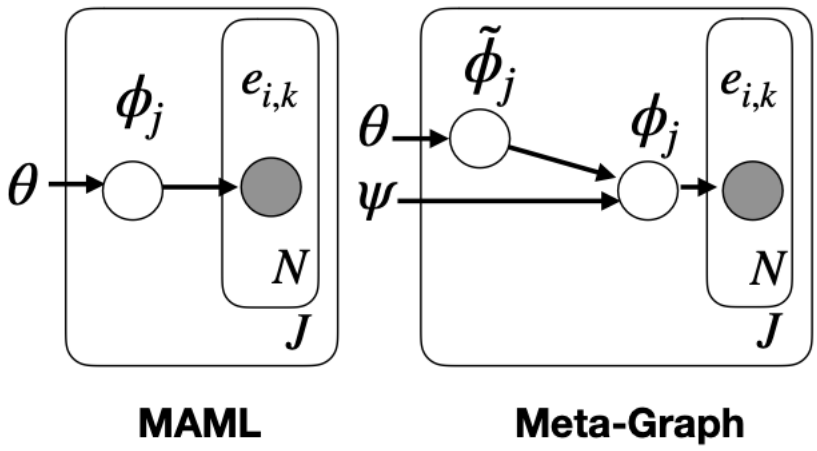


Fig. 5: A high level comparison of the adapted MAML and Meta-Graph architectures.

Algorithm 1: Meta-Graph for Few Shot Link Prediction

Result: Global parameters θ, Graph signature function ψ
Initialize learning rates: α, ϵ
Sample a mini-batch of graphs, $\mathcal{G}_{batch}$ from $p(\mathcal{G})$;
for *each* $\mathcal{G} \in \mathcal{G}_{batch}$ **do**
  $\mathcal{E} = \mathcal{E}^{\text{train}} \cup \mathcal{E}^{\text{val}} \cup \mathcal{E}^{\text{test}}$ // Split edges into train, val, and test
  $s_{\mathcal{G}} = \psi(\mathcal{G}, \mathcal{E}^{\text{train}})$ // Compute graph signature
  Initialize: $\phi^{(0)} \leftarrow \theta$ // Initialize local parameters via global parameters
  for $k\ in\ [1:K]$ **do**
    $s_{\mathcal{G}} = \text{stopgrad}(s_{\mathcal{G}})$ // Stop Gradients to Graph Signature
    $\mathcal{L}_{train} = \mathbb{E}_q[\log p(A^{\text{train}}|Z)] - KL[q_\phi(Z|\mathcal{E}^{\text{train}}, s_{\mathcal{G}})||p(z)]$
    Update $\phi^{(k)} \leftarrow \phi^{(k-1)} - \alpha \nabla_\phi \mathcal{L}_{train}$
  end
  Initialize: $\theta \leftarrow \phi_K$
  $s_{\mathcal{G}} = \psi(\mathcal{G}, \mathcal{E}^{\text{val}} \cup \mathcal{E}^{\text{train}})$ // Compute graph signature with validation edges
  $\mathcal{L}_{val} = \mathbb{E}_q[\log p(A^{\text{val}}|Z)] - KL[q(Z|\mathcal{E}^{\text{val}} \cup \mathcal{E}^{\text{train}}, s_{\mathcal{G}})||p(z)]$
  Update $\theta \leftarrow \theta - \epsilon \nabla_\theta \mathcal{L}_{val}$
  Update $\psi \leftarrow \psi - \epsilon \nabla_\psi \mathcal{L}_{val}$
end

V. Experiments

Dataset recap: We have collected restaurant reviews for 8 cities that we obtained by web scraping Google Maps. 3 of those cities represented a much larger population base, and therefore have a much larger data sample for training. We have also collected data from 5 smaller cities in order to show that transfer learning from the model trained on the larger dataset can then be applied with minimal retraining with strong results on the new unseen data.

A. Joint Loss and Transfer-Learning Experiments on LightGCN architecture

The following experiments were conducted on the modified LightGCN architecture:

- Hyperparameter optimization on larger city (learning rate, number of GCN layers)
- Joint training individual models for Los Angeles and Toronto by optimizing over a joint loss function
- Transfer learning: Pre-training on a large city and transferring to smaller cities, fine-tuning by freezing the first layers
- Optimization of the number of GCN layers

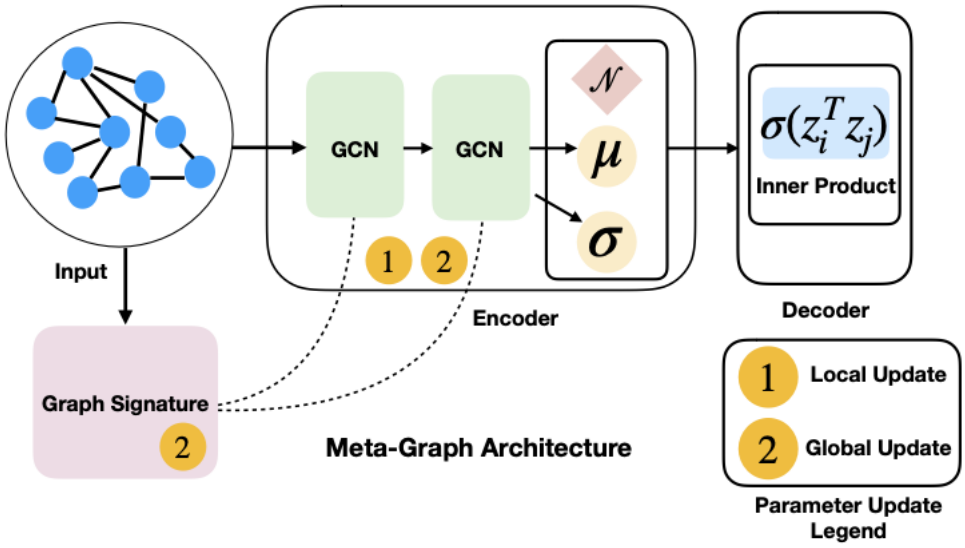


Fig. 6: A high level diagram of the Meta-Graph architecture.

For training the base model, we have only kept users that have given 10 or more ratings, and restaurants that have received 20 or more ratings.

B. Meta-Learning Experiments

1) Baseline Model: We implemented a baseline for comparison with our two meta-learning models. Our baseline models consist of the same VGAE architecture used for local link prediction in the meta-learning models. A separate VGAE model was used to evaluate each city. Each model was trained over the data for all but one city sequentially, then fine-tuned and evaluated on the held-out city.

2) Adding Node Attributes: To further make use of our web-scraped Google Reviews data in producing an effective recommender system, we experimented with augmenting our meta-learning models by adding review text information as node features to the graphs in our meta-dataset. To create a feature embeddings from the restaurant review text, we used a pretrained BERT sentence encoder to produce a 50-element embedding from each review's text. We then use a matrix of review embeddings, where each row is one review embedding, as a node feature for each node in a city graph. With this method, there are repeat embedding rows between nodes since the embedding of a review written by a user about a restaurant is included in the node feature matrix of both the user and the restaurant.

The following experiments were conducted to evaluate our meta-learning approach:

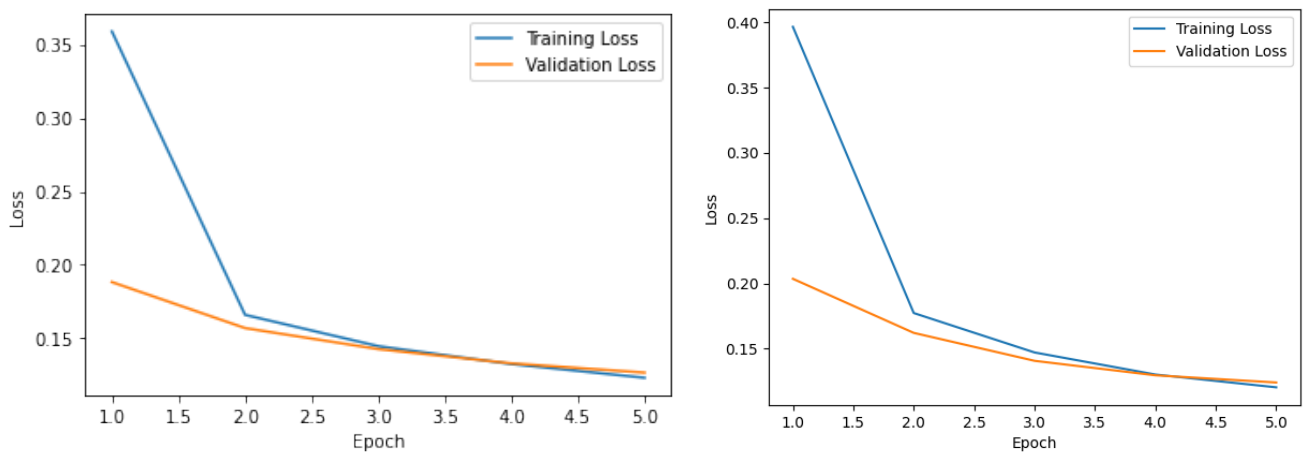
- Training and testing a fine-tuned VGAE baseline model for each city
- Training a MAML model over all cities but one, and testing on the held-out city. This process was repeated for each city in the dataset.
- Training a Meta-Graph model over all cities but one, and testing on the held-out city. This process was repeated for each city in the dataset.
- Repeating the MAML and Meta-Graph experiments using review text embeddings as node features.

VI. Results

A. Transfer Learning Approach

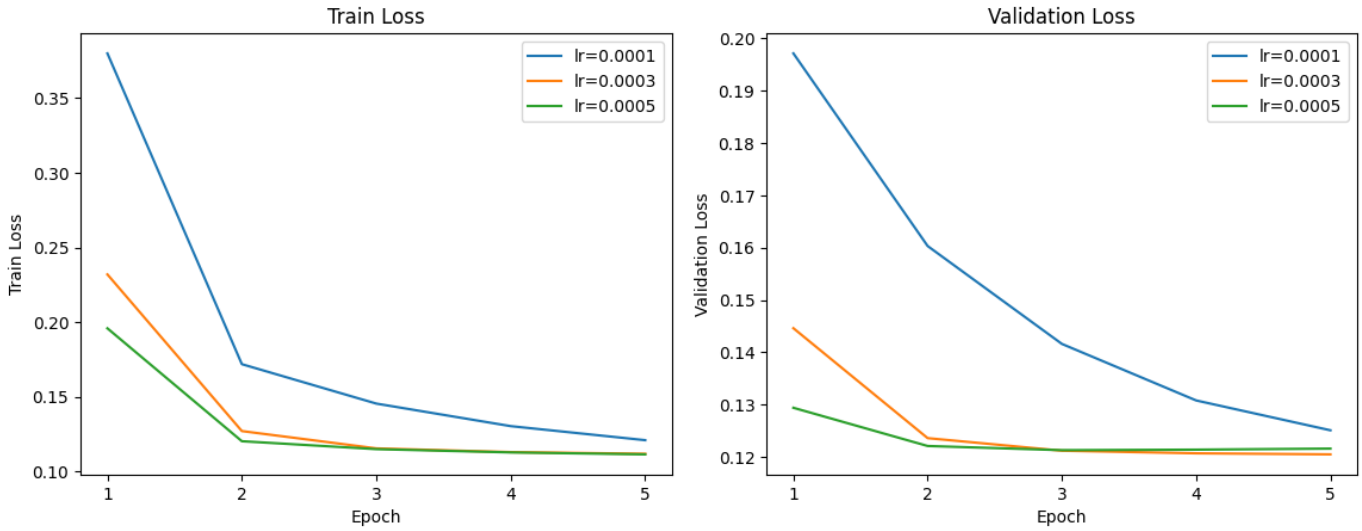
1) Baseline: We trained this LightGCN model separately on each cities datasets. Below, plots of the RMSE values for the train/validation split for the two largest cities are shown.

2) Joint Training: The joint training on individual models for Los Angeles and Toronto datasets was done by optimizing over a joint loss function. With this method, we reach a plateau in validation loss at about 0.12 which is better than our baseline. Furthermore, experimenting with the learning rate show that performance can be improved that way. We fix the learning rate value to 0.0005 for all subsequent experiments.



Toronto Los Angeles

Fig. 7: Baseline: Toronto and Los Angeles (3-layer GCN)



Train Loss Validation Loss

Fig. 8: Joint Loss training on Toronto and Los Angeles datasets and learning rate parameter tuning (3-layer GCN)

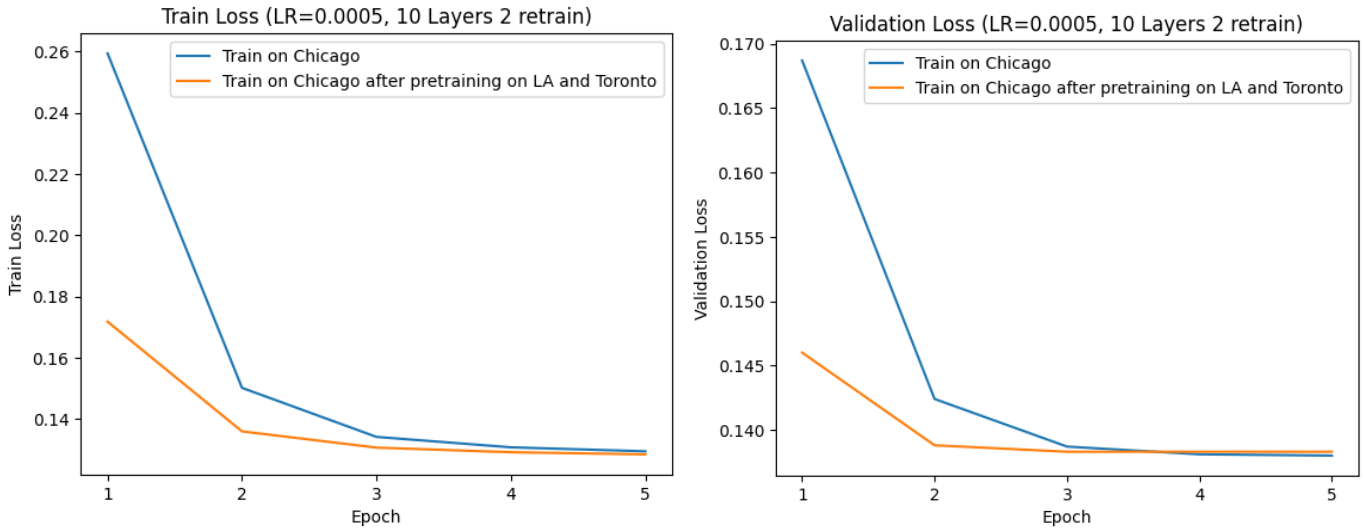
3) Transfer Learning: After pre-training a LightGCN model on the Los Angeles dataset, we retrain it on Toronto and then on a smaller city while fine-tuning the 2 last layers only. This achieves better results and converges faster in comparison to training on the smaller city directly, without pre-training. It was shown that a 10 GCN-layer model performs this transfer learning quite well, as opposed to a 5 GCN-layer model where the difference is vanishingly small. This is possibly due to the fact that we integrate up to 10-hop node information, which results in high-order connectivity in the user-restaurants bipartite graph:

$$u_1 \leftarrow r_2 \leftarrow u_5 \leftarrow r_3 \leftarrow u_3 \leftarrow r_5 \leftarrow u_6 \leftarrow r_1 \leftarrow u_2 \leftarrow \dots$$

We show that the pre-trained and fine-tuned models result in faster convergence for all cities and lower RMSEs for San Antonio, Seattle, Atlanta and Detroit after 5 epochs. We achieve validation losses in the range 0.05 to 0.21. We find no clear correlation between the city's size and validation loss after convergence, as shown in Table II.

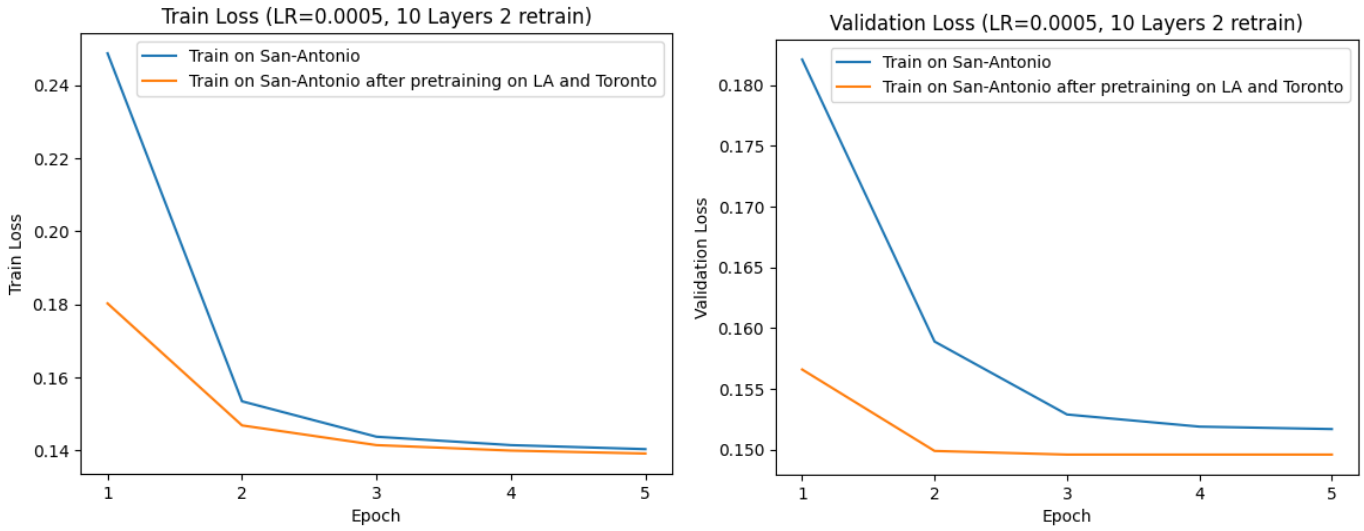
B. Meta-Learning

The results from our meta-learning experiments can be seen in Table III.



Train Loss Validation Loss

Fig. 9: Chicago, baseline and after pre-training and fine-tuning



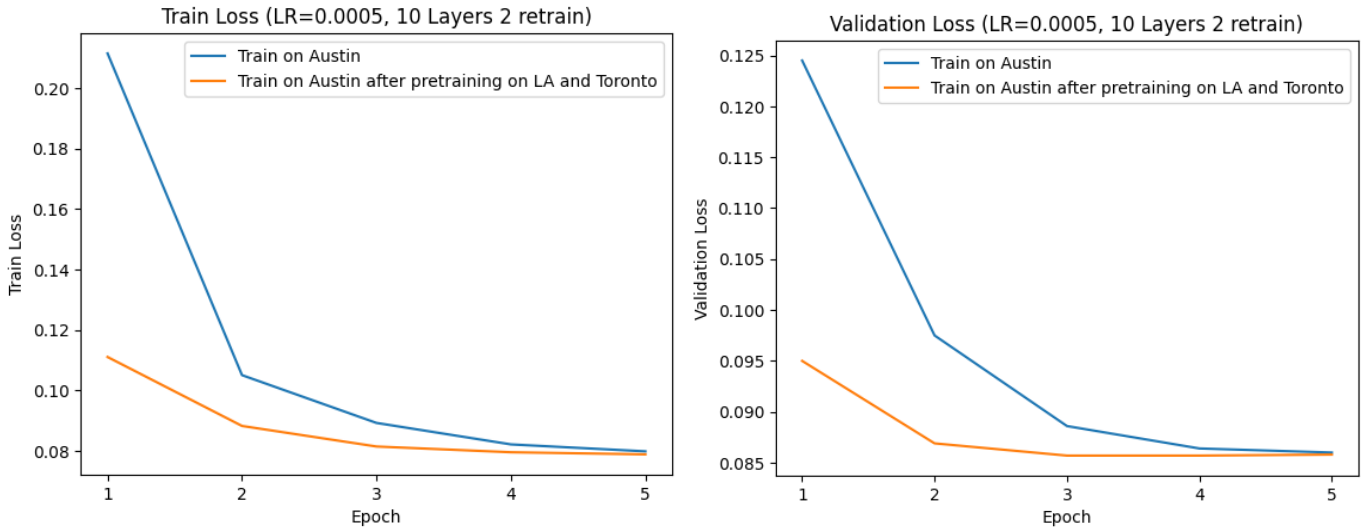
Train Loss Validation Loss

Fig. 10: San Antonio, baseline and after pre-training and fine-tuning

City	# ratings	# Restaurants	validation loss
Chicago	1,071,137	5,580	**0.14**
San Antonio	986,335	3,410	**0.15**
Austin	533,245	2,397	**0.09**
Seattle	236,172	1,689	**0.05**
Atlanta	256,897	1,676	**0.11**
Detroit	138,849	995	**0.21**

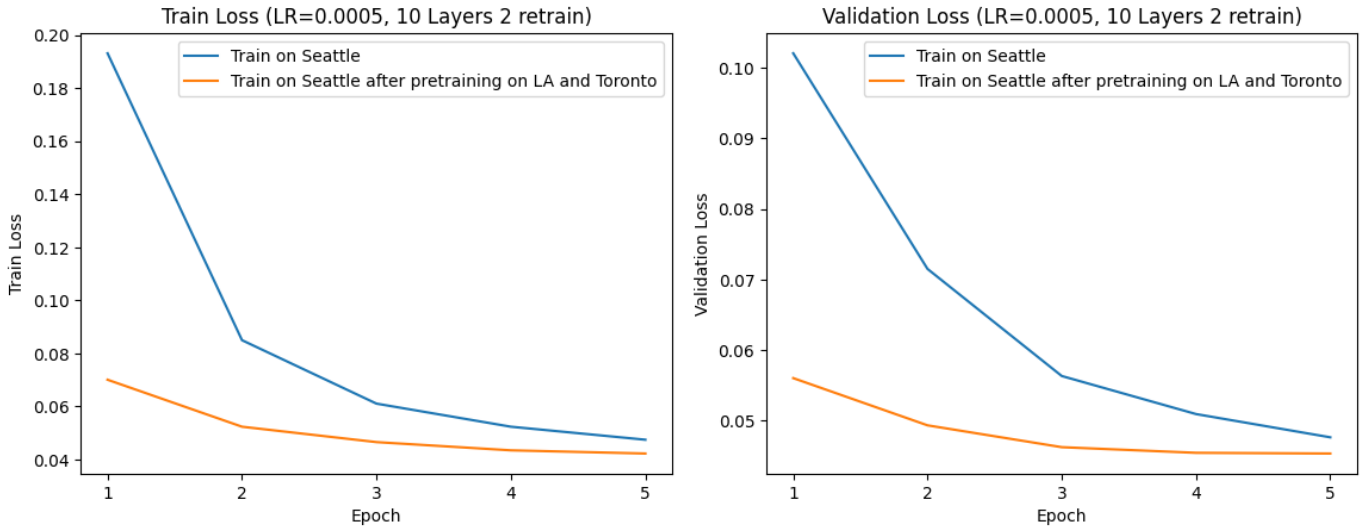
TABLE II: Validation loss after fine-tuning during 5 epochs

In our meta-learning approach, we develop models for the link prediction task, so the final output of our recommender models given an input of nodes is a set of edges to be compared with a held-out set of edges. In other words, the models predict the existence, or non-existence, of a link between pairs of nodes. Because of this, we use AUC (area under the receiving operating curve) and AP (average precision) as evaluation metrics. The ROC used in the computation of AUC is the plot of



Train Loss Validation Loss

Fig. 11: Austin, baseline and after pre-training and fine-tuning



Train Loss Validation Loss

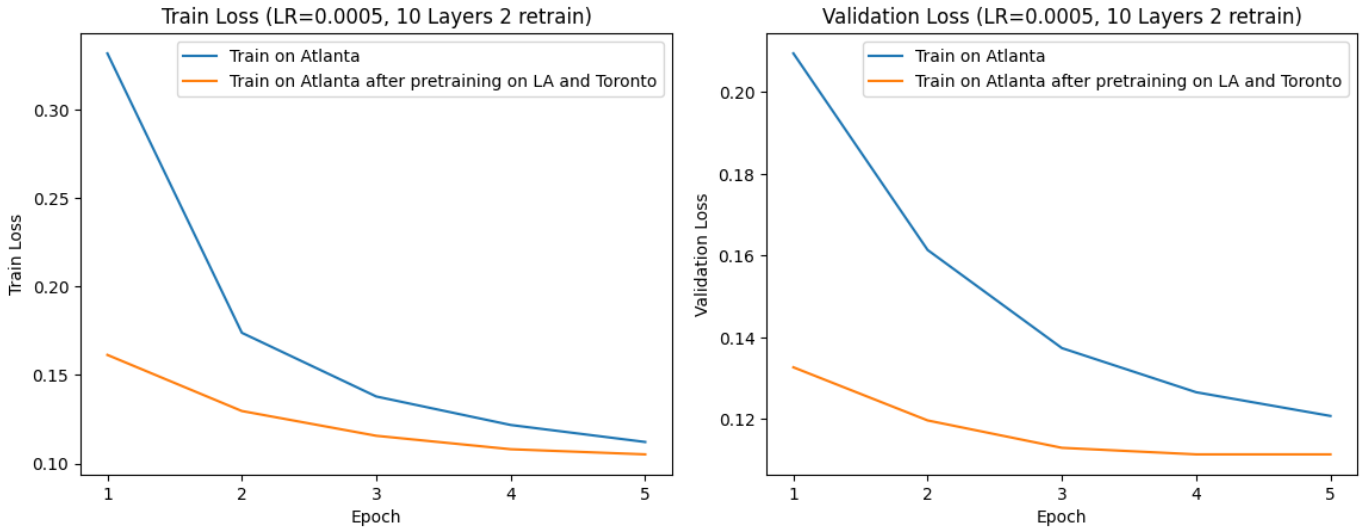
Fig. 12: Seattle, baseline and after pre-training and fine-tuning

	VGAE Baseline		MAML		Meta-Graph		MAML w/ Review Embeddings		Meta-Graph w/ Review Embeddings	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
LA	0.5004	0.5002	0.7773	0.7880	0.7766	0.7912	0.7843	0.7921	0.8102	0.8001
Chicago	0.5040	0.5020	0.772103	0.786686	0.776630	0.791274	0.7751	0.7890	0.7882	0.7992
Toronto	0.6019	0.6609	0.773651	0.789384	0.776631	0.791274	0.8192	0.8392	0.8902	0.8102
Atlanta	0.8865	0.8453	0.773965	0.789047	0.776649	0.791288	0.7621	0.7644	0.7701	0.7789
San Antonio	0.5081	0.5084	0.7822	0.7902	0.7991	0.7982	0.7862	0.7961	0.8001	0.8032
Austin	0.6001	0.6004	0.7823	0.7901	0.7926	0.7943	0.7902	0.7912	0.7944	0.7992
Seattle	0.6023	0.6028	0.6991	0.6982	0.6901	0.6871	0.6994	0.6992	0.6922	0.6901

TABLE III: Meta-Learning Results

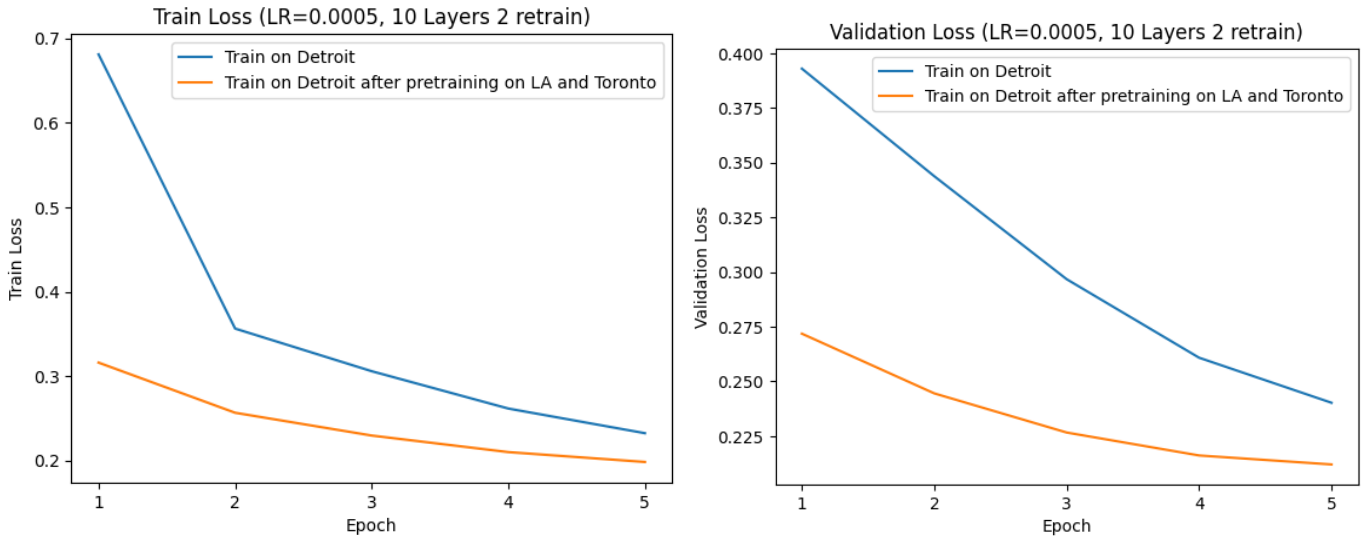
TPR (true positive rate) vs. FPR (false positive rate), where:

$$TPR = \frac{TP}{TP + FN}$$



Train Loss Validation Loss

Fig. 13: Atlanta, baseline and after pre-training and fine-tuning



Train Loss Validation Loss

Fig. 14: Detroit, baseline and after pre-training and fine-tuning

$$FPR = \frac{FP}{TN + FP}$$

, as a function of true positives (TP), false negatives (FN), falspe postives (FP), and true negatives (TN). The AP values are computed as:

$$\frac{TP}{TP + FP}$$

We can see from these results that for all cities except for Atlanta, both the MAML and Meta-Graph models outperformed the baseline models, showing that for these cities meta-training over several cities produced a more effective link prediction model. On average, MAML had a 22.3% higher AP than the baseline models and Meta-graph had a 29.0% higher AP than the baseline models. While using this metric, Meta-Graph performs slightly better, there is no clear outperforming model between MAML and Meta-Graph from these experiments, as on average, the models had comparable performance, with MAML slightly outperforming Meta-Graph for some cities and Meta-Graph slightly outperforming MAML for other cities. We see that for

all experiments on all cities, the addition of Google Review text embeddings as node features to the graphs increased the performance of both meta-learning models slightly. Adding review embeddings increased the performance of MAML by an average of 0.7% and increased the performance of Meta-Graph by an average of 0.6%.

VII. Conclusions and Future Work

The goal of this project was to explore the effectiveness of several approaches in utilizing restaurant-user patterns of other geographical locations to create a better recommender model with less data. In both the transfer learning and meta-learning approaches we experimented with, we found that these approaches were able to perform noticeably better than the baseline models that do not utilize data from other geographical locations. Given this, it seems the approaches we have tried are effective in creating higher-performing city-specific restaurant recommendation models by utilizing data from other cities.

One of the future steps we hope to take to continue this work is to experiment with testing the effectiveness of these approaches in being able to train on cities with very small amounts of data. We hope to extend the experimentation to more scraped data from even smaller cities to get a better understanding of how these approaches mitigate the limited data problem.

We also look to combine the two approaches, by first narrowing down the list of all restaurants in a city that could be recommended to the user by applying the MAML-trained model, then ranking the candidates by scoring them using the LightGCN model to create a stronger recommendation system.

VIII. Team Contributions

Hamdy gathered and sourced the required data sets and developed the initial implementation of a LightGCN model that was used to extended to multiple tasks. Hamdy and Rouven implemented and ran the experiments for fine-tuning the GCN model. Megan implemented and experimented with the MAML and Meta-Graph models, the VGAE baseline, and the modified meta-learning models using review text encodings.

References

[1] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, Meng Wang. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks.

[3] Thomas N Kipf and Max Welling. Variational graph auto-encoders.

[4] Avishek Joey Bose, Ankit Jain, Piero Molino, William L. Hamilton, Link Prediction from Sparse Data Using Meta Learning.

[5] Chelsea Finn, CS330 Deep Multi-Task and Meta-Learning, Fall 2022. https://cs330.stanford.edu/lecture_slides/cs330_transfer_meta_learning.pdf

[6] Xing Fang, Making recommendations using transfer learning

[7] Debmalya Mandal, Sourav Medya, Brian Uzzi, Charu Aggarwal, Meta-Learning with Graph Neural Network: Methods and Applications