COSE474-2021F: Final Project Report “GNN based Product Recommendation”

T. Wesley Bailey Han Jin Choi Dong Hwan Kim

Abstract

With the original idea of using a Graph Neural Network (GNN) for product recommendations, we wanted to use more feature data for customized recommendation advertising based on individual evaluations. From the Amazon text reviews, we dragged out numerical features data from text using Natural Language Processing (NLP) for pre-processing sequences, based on individual text reviews and ratings. We built our own data-set by merging advertising data with pre-processed features data and user-specific data (id, access-points, view method). This GNN based product recommendation algorithm and accompanying data-set provides an appropriate selection of advertising platforms with relevant merchandise based on user preference. It performs a more accurate selection of recommended products taking into account individual preference. The algorithm’s intended purpose is e-commerce, though it can be used in similar environments where review-based recommendations might be given.

1. Introduction

1.1. Motivation

Recommendation systems have become one of the key technologies in e-commerce to date. Recommendation systems are traditionally branched in two ways: content-based recommendation systems and collaborative filtering systems. The Collaborative filtering system, now the leading method in recommendations, has provided more utility in the recommendation systems after Netflix pioneered them as movie recommendation systems for viewers, especially the latent factor collaborative filtering.

Our key intuition in developing an effective and accurate customized recommendation system is realizing that those who review products invariably evaluate other products. A simple observation but barely implemented so far. Our assumption is that the use of customer reviews can be taken as another factor for the effective recommendation of new products. With this in mind, we decided to use product reviews for predicting what further products a customer might want to purchase.

The method we considered is a matrix completion task. The Graph Convolutional Matrix Completion (GCMC) (van den Berg et al., 2017) viewed matrix completion as a link prediction problem among graphs. For the recommendation system, we have to handle a large amount of features and users who have those features. The GCMC builds a graph based neural network on a matrix with user-item relations and uses it for predictions. So if you want to predict someone’s product preferences, you have to know their information and consist a matrix shape. Preferably, we would not use the user’s private info to intuit their preferences. Improving data-transparency is another goal of this method. Our proposed alternative it to employ the user’s willfully provided text.

1.2. Problem definition

Based on the GCMC, we want to predict the potential rating of someone who might want a product with their given current ratings without invading their private info. To solve this, we looked for the product preference tendency with a review written by each customer. We then infer the user’s preference to items they have not rated. So we build the model based on this GCMC implemented repository (tanimutomo, 2021), with generous modifications, using our own dataset to predict user preferences based on user reviews.

1.3. Concise description of contribution

- **Dong Hwan Kim :** Implementation of NLP preprocessing.
  Dataset feature selection, building, appliance.
  Modifications of experimental design points.

- **Wesley Bailey :** Training on lower gpu machine for improvements, research, report edits for concision,

- **Han Jin Choi :** Setting up the environment.

2. Methods

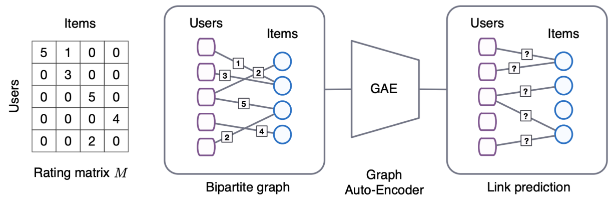

Figure 1. Overall Structure of Computation

2.1. Main figure

Combining two models related to NLP and GNN to make user-specific features to be considered in predicting the product recommendation, which could lead to more accurate prediction.

2.2. Text analyzer

The Natural Language Processing (NLP) was performed with sentiment analysis pretrained model (huggingface.co), distilled version of the BERT base model and fine-tuned on SST-2. DistilBERT (Sanh et al., 2019) is a smaller and faster version of original BERT. We used the pipeline function which made easier to use this pre-trained model. Through this model, we converted text reviews of Amazon product(Ni et al., 2018) into numerical process-able shape, under condition of $H_{text} \times x_{polarity}$ with u_{rate} over positive evaluations. $S_{score} = H_{text} \times x_{polarity}\ \ \&\ \ u_{rate} \geq Threshold$

2.3. Graph neural network

This Graph neural network (GNN) part is the same as what GCMC did. We build a matrix M of shape $N_u \times N_v$. Entities M_{ij} is observed rating (user i rated item j). This matrix can be viewed as graph $G = (\mathcal{W}, \mathcal{E}, \mathcal{R})$. Graph auto-encoder predict the graph G's link. Graph auto-encoder is consist of two parts which are graph convolutional encoder and bilinear decoder. Graph convolutional encoder extracts user and item embeddings $\mathcal{U}$ and $\mathcal{V}$. This can be formulated as $[\mathcal{U}, \mathcal{V}] = f(X, M_1, ..., M_R)$ where M_i is adjacency matrix that user rates i to item. We can also reformulate bilinear decoder as $\tilde{M} = g(U, V)$.

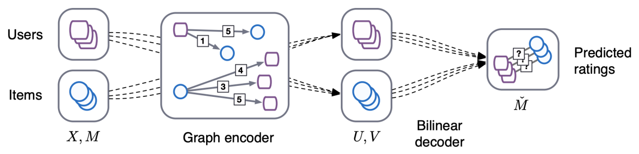

Figure 2. Graph Convolutional Encoder Flow

2.4. Formulation

2.4.1. Relation of Undirected Graph

$$G = (\mathcal{W}, \mathcal{E}, \mathcal{R}),$$
$$\mathcal{W} = \mathcal{U} \cup \mathcal{V},$$
$$(u_i, r, v_j) \in \mathcal{E},$$
$$r \in [1, \ldots, R] = \mathcal{R}$$

* $\mathcal{W}$ = User/Item
* $\mathcal{E}$ = Edge
* $\mathcal{R}$ = Ratings

2.4.2. Implementation of Effective Vector Computing

$$[U, V] = f(X, M_1, \ldots, M_{R-1}, M_R) = \sigma([H_u, H_v]W^T)$$
$$\text{with}[H_u, H_v] = \sigma(\sum_{r=1}^{R} D^{-1} M_r X W_r^T)$$
$$M_r = \begin{pmatrix} 0 & M_r \\ M_r^t & 0 \end{pmatrix}$$

* U, V : Node Embedding Matrix
* H : Hidden Matrix
* W : Weight Matrix
* D : Normalization Matrix
* X : Feature Matrix
* W_r^T : Edge-type Specific Weight Matrix

2.4.3. Weight Sharing

$$W_r = \sum_{s=1}^{r} T_s$$
$$Q_r = \sum_{s=1}^{n_b} a_{rs} P_s$$

Spec: Ratings for each node is considered as inference factor, passed through user and item nodes with the Message Passing. Since all the user and item has each ratings, it uses **Weight Sharing** for equally optimized weight.

* n_b : Number of Basis Weight
* a_{rs} : Learn-able Parameter
* Q_r : Decoder Weight Matrix

3. Experiments

3.1. Dataset

- Amazon Product Reviews(Ni et al., 2018)
- Product rate and access point (PODSYP, 2019)

3.2. Computing resource

- CPU
- Ubuntu 18.04.5 LTS
- PyTorch 1.0.0
- PyTorch Geometric 1.0.1

3.3. Experimental design

3.3.1. Activation Function

As this model has densely connected layers and through 1000 epochs of learning, using **ReLU** could lead the model to the **Dead ReLU** problem.
The **LeakyReLU** was a good alternative, but we assume that this model might need more learnable parameters for extracting features.
The **PReLU** uses the learnable parameter slope on the negative side of x, which satisfies our need.

3.3.2. Weight Initialization Function

The model was originally using **Xavier Uniform** function for weight initializing, but since we were using **ReLU** variations, we thought it would be more effective using **He Uniform** function.

3.3.3. Threshold in the NLP Unit

The pre-trained NLP model has a probability of misunderstanding, among user reviews. To get solid features of text, we used user ratings at a product for decision making whether the review was a positive or negative factor. This feature selection helped decrease loss.

3.3.4. Learning Rate Scheduler

Learning Rate scheduler applied to converge in Global Minima. We observed that our model has too big learning rates in presumed minima point. Weight Decay function **Reduce-LR-on-Plateau** was used since our learning rate was not shrinking and do not converge on Global Minima.

3.3.5. Data-set based built relation defined

Redefined the relation based on our own data-set(Dong-Hwan) to use more features to Node connections.

3.4. Quantitative results

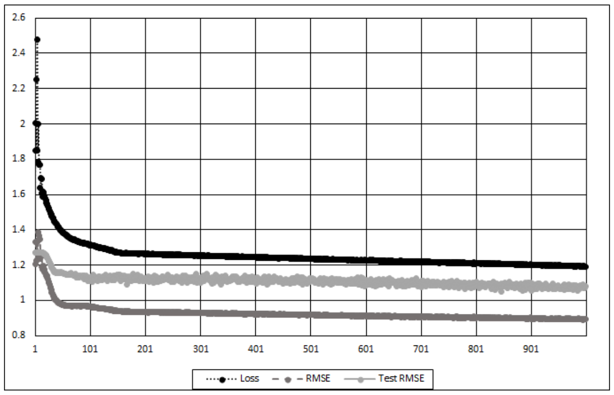

Figure 3. Loss and RMSE of train and test

	Loss	Train_RMSE	Test_RMSE
Base_GCMC	1.350716	1.066707	1.186720
Mode_GCMC	1.327854	1.038250	1.136517
Mode w/ data set	1.190801	0.891637	1.079486

Table 1. Baseline and Modified Comparison

Note: We compared the baseline GCMC model with the modified GCMC to observe improved performances among the same dataset.

Note: w/data set means, the modified GCMC with our own dataset.

3.5. Qualitative results

3.5.1. Performances

The original model did not show enough performances predicting un-seen data. We presumed that the reason of this problem was about underfitting and **Dead ReLU** problem. Our modification improved the **RMSE** score mainly the Test data, slightly on Train data and Loss

3.5.2. Dataset

The assumption of user-specified feature selection will help predict un-seen user preferences, was successful. It showed significant improvement in prediction. We assumed that a bigger dataset and more trainable parameters, with specified features, had credits.

4. Future direction

In our work, we made static graph nodes that can only predict the node's link under a single fixed graph. However, in the real world, the graph nodes relations are not static. To predict the unseen data, we need a model that expands flexibly depending on node connections. To compensate for this defect, the combination with the idea of the GraphSAGE (Hamilton et al., 2017) may resolve this problem. The GraphSAGE is a general inductive framework that leverages node feature information to generate node embeddings for previously unseen data. By using this idea, maybe we could implement a real-time graph, with fast and effective computing if optimized. Furthermore, In terms of affective computing, there are lots of alternatives instead of using text data for feature extracting and node relation predictions. Since the Natural Language Processing(NLP) model costs a lot of computation powers and time, selecting other types of customized features can make an even more effective model.

References

Dong-Hwan, K. URL https://drive.google.com/file/d/18lZkSoCgX3XjI3Ob60izqmUBG1nQxdSP/view?usp=sharing.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs, 2017.

huggingface.co. distilbert-base-uncased-finetuned-sst-2-english.

Ni, J., Li, J., and McAuley, J. Amazon review data, 2018. URL https://nijianmo.github.io/amazon/index.html.

PODSYP. Product analysis, 2019. URL https://www.kaggle.com/podsyp/how-to-do-product-analytics.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL http://arxiv.org/abs/1910.01108.

tanimutomo. gcmc. https://github.com/tanimutomo/gcmc, 2021.

van den Berg, R., Kipf, T. N., and Welling, M. Graph convolutional matrix completion, 2017.