# Research Project Proposal: Learning-to-Hash with GNN for Efficient and Effective Recommender Systems

JIAYING LI, The University of California, USA
WENHE ZHANG, The University of California, USA
YU HOU, The University of California, USA
YUHAN SHAO, The University of California, USA

## 1 INTRODUCTION

Nowadays, recommender systems are becoming more and more popular in the real world. A successful recommender system should not only generate effective recommendations, but also generate recommendations with high throughput. Most of the research on recommender systems nowadays focus more on generating accurate recommendations with high performance rather than high speed. However, high efficiency is also important in designing a successful recommendation system since recommendation systems sometimes need to fulfill a great deal of requests from users at the same time. In our project, we hope to design a recommendation system that could give effective recommendations by ensuring high throughput at the same time.

Graph neural network techniques have been widely utilized in recommendation systems because most of the information in recommender systems essentially has graph structure. In our project, we propose to use GNN in the recommendation system but we will seek to find a more simple and light-weight GNN design such as LightGCN[3] to achieve the efficiency requirement.

Existing recommender systems typically generate high-dimensional, continuous embeddings for users and products and use their inner products to estimate their relevance and do the recommendations. However, finding top-K maximum inner products between two continuous high-dimension embeddings is quite challenging. To make the recommendations more efficient, we also propose to use learning-to-hash functions in our project, which aims to learn a hash function to transform continuous vectors into binary vectors to accelerate the inner product search.

In our work, by experimenting with different GNN and learning-to-hash functions, we hope that we could design a recommendation system that could give highly effective recommendations with high efficiency.

## 2 RELATED WORK

The major tasks of recommender systems are predicting whether users would interact with an item, and returning recommendations based on predicted user behavior. User-item interaction graphs are extensively used in this process. Traditional approaches represent the objects in user-item interaction graphs as continuous embedding vectors, but suffer from the computational costs of searching for similar continuous vectors. In recent years, more and more researchers are working on the hashing-based representation of graphs for recommender systems. For example, GHashing[8], a graphical neural network(GNN) based semantic hashing method, could produce hash indexes that enable graph lookup in constant time. HashGNN[9] proposed an effective framework for jointly learning continuous vectors and hash codes for graphs.

Besides the explorations on hashing-based representations, some researchers also delved into the utilization of subgraph structures of users to improve the embedding learning. Inspired by Graph Convolution Network(GCN), some researchers proposed Neural Graph Collaborative Filtering(NGCF) model[10], which involves feature transformation, neighborhood aggregation, and nonlinear activation to refine graph embeddings. LightGCN[3] further simplifies the structure of NGCF and only incorporates neighborhood aggregation, the essential component in GCN, to learn the embeddings on user-item interaction graphs. It is more light-weight, but achieves better empirical performance than NGCF.

Recommender systems not only need fit-for-purpose graph representations, but also involve similarity search for the top recommendations. Faiss[5], a library for efficient similarity search and clustering of dense vectors, was created to facilitate this process. Accelerating similarity search also helps the efficiency of recommender systems.

## 3 DATA PLAN

The datasets used in related work include the recommender system datasets from Netflix, Yahoo, Amazon, MovieLens, Gowalla, Pinterest, and Alibaba. Most of the datasets are in the form of user ratings associated with items. Considering the data availability and our computational resources, we would like to conduct our experiments on the Netflix[4] and MovieLens[1] dataset, which respectively include 100480507 and 1000209 user-item interactions in total.

## 4 SOLUTION PLAN

### 4.1 Sketch of the proposed method

Basically, we separate the main architecture of our project into 2 parts:

(1) LightGCN[3] for recommendation system

In this part, there are many loss functions that can be chosen from, we would do some research on typical loss functions for recommendation systems, e.g.: cross-entropy loss and ranking loss. Specifically, we may consider the loss function suggested in the LightGCN paper: Bayesian Personalized Ranking (BPR) loss.

(2) Learning-based Hashing[7] (Refer to the un-publish paper: Learning-based Hashing for Threshold Inner Product Search)

Authors' addresses: Jiaying Li, The University of California, Los Angeles, USA, jl64@ucla.edu; Wenhe Zhang, The University of California, Los Angeles, USA, zwh990119@g.ucla.edu; Yu Hou, The University of California, Los Angeles, USA, yuhou316@g.ucla.edu; Yuhan Shao, The University of California, Los Angeles, USA, yuhan17@g.ucla.edu.

In this part, we would consider how to train the binary code since the hash codes are 0s and 1s which are discrete, not continuous. Therefore, rather than use the gradient descent directly, we would first use some relaxation methods to transform the binary code into something that can use gradient descent. One possible relaxation method would be the function like: tanh(x) + regularization term which maps discrete 0s and 1s into the continuous interval [-1, 1].

### 4.2 Major contributions of the proposed method

Our model considers both efficiency and effectiveness for our recommendation system. We suppose our model could achieve the same or higher effectiveness by also considering the efficiency.

## 5 EVALUATION PLAN

### 5.1 Baseline

The main competing methods are Faiss[5] and HashGNN[9]. Faiss is a popular library for efficient similarity search and clustering of dense vectors. There are several methods used by this library. For example, graph-based methods HNSW[6] and NSG[2]. So one of our baseline is learning the continuous embedding and using the method from Faiss to find the top k item vectors which has the largest inner product with the user vector. Another baseline is HashGNN[9], which is a novel end-to-end learning framework for hashing graph data. It consists of two components, a GNN encoder for learning node representations, and a hash layer for encoding representations to hash codes.

### 5.2 Evaluation

We evaluate efficiency and effectiveness of our methods. In addition to measuring the accuracy and speed of each method individually, we also compare the accuracy and speed of the trade-offs simultaneously. To make the comparison fair, we will measure our method on the Netflix[4] and MovieLens[1] datasets at a fixed query throughput and compare the effectiveness of our methods with the effectiveness of baseline methods.

## 6 TIMELINE

Here is the schedule of our project:

- Week 4: Proposal
- Week 5: Get the dataset and run the baseline(faiss, HashGNN)
- Week 6 and 7: LightGCN + Hash + Evaluation
- Week 8 and 9: Evaluation
- Week 10: Report and presentation

## REFERENCES

[1] Harper F. Maxwell and Konstan Joseph A. 2015. The MovieLens Datasets: History and Context. (2015). https://grouplens.org/datasets/movielens/1m/

[2] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proceedings of the VLDB Endowment* 12, 5 (2019), 461–474. https://doi.org/10.14778/3303753.3303754

[3] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation.* Association for Computing Machinery, New York, NY, USA, 639–648. https://doi.org/10.1145/3397271.3401063

[4] Bennett James and Lanning Stan. 2007. The Netflix Prize. (2007). https://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a

[5] Jeff Johnson, Matthijs Douze, and Herve Jegou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data* 7, 3 (2021), 535–547. https://doi.org/10.1109/tbdata.2019.2921572

[6] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 824–836. https://doi.org/10.1109/tpami.2018.2889473

[7] Zongyue Qin. 2022. Learning-based Hashing for Threshold Inner Product Search. (2022).

[8] Zongyue Qin, Yunsheng Bai, and Yizhou Sun. 2020. GHashing: Semantic Graph Hashing for Approximate Similarity Search in Graph Databases. 2062–2072. https://doi.org/10.1145/3394486.3403257

[9] Qiaoyu Tan, Ninghao Liu, Xing Zhao, Hongxia Yang, Jingren Zhou, and Xia Hu. 2020. Learning to hash with graph neural networks for recommender systems. *Proceedings of The Web Conference 2020* (2020). https://doi.org/10.1145/3366423.3380266

[10] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. https://doi.org/10.1145/3331184.3331267