[PMLDL] D1.3 Report

Team

Team name: *16th Data Science Division*
Team members:
- Bulat Sharipov (DS-01); b.sharipov@innopolis.university
- Dinar Yakupov (DS-01); d.yakupov@innopolis.university
- Danil Fathutdinov (DS-01); d.fathutdinov@innopolis.university

Github

Link

Project Topic

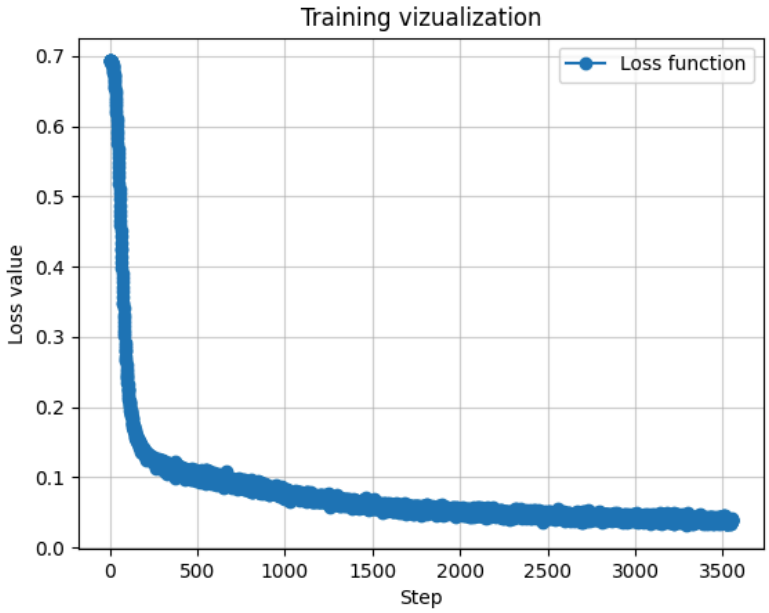
Creating a Recommendation Systems using Graph Neural Networks and Yambda Dataset. We will basically create a web-service that will recommend you next song to listen to, given your previous preferences.

Current progress

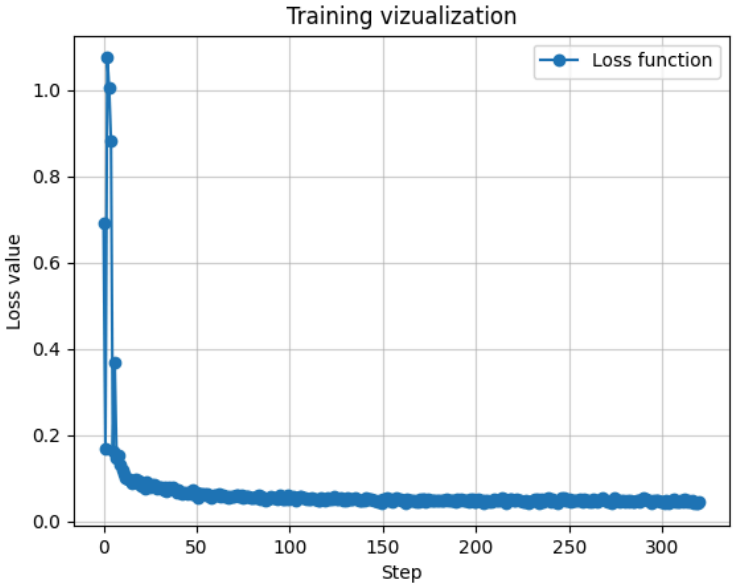
Below we will describe the main tasks that we were solving during the checkpoint

LightGCN x MBGCN experiment

Idea: Train a LightGCN over the training dataset, use learned embeddings from LightGCN as initialization to MBGCN embeddings. We have pre-trained the embeddings using LightGCN architecture, using the "Listens" interaction type. In our setup, we set “Listens” to 1, if the played ratio percentage was >= 50. Then, we have trained the embeddings for 1 epoch. To properly handle a large graph dataset, we have used `LinkNeighborLoader` to avoid Out-Of-Memory issues. Below is the plot, describing the pre-training stage.



Then, we have used the MBGCN architecture. Here, we have initialized the initial embeddings with these ones learned by LightGCN. We also have used optimizer loaders, and trained the model for 3 epochs. Below is the learning curve for the MBGCN

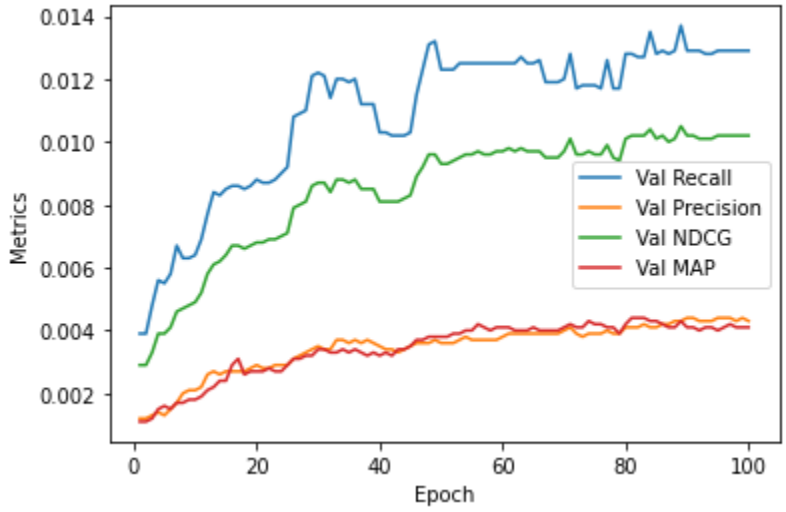


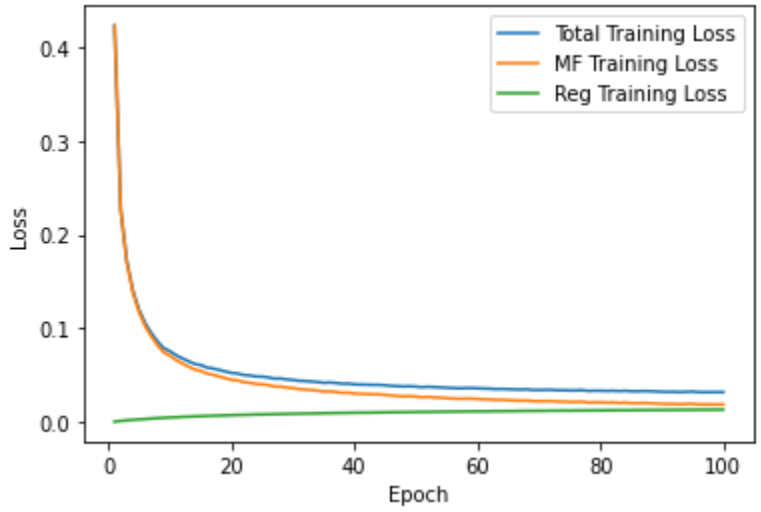
The resulting metrics from this experiment: `Recall@20=0.0008 NDCG@20=0.0003`

LightGCN training

We enhanced our end-to-end LightGCN recommendation pipeline. The “Yambda” dataset was optimized by removing unnecessary features, followed by a proper global temporal split into training, validation, and test sets. We incorporated normalized external item embeddings, kept them frozen during training, and used “like” events as positive interactions. The training process is now CUDA-enabled and memory-safe. As a result, we observe consistent improvements in

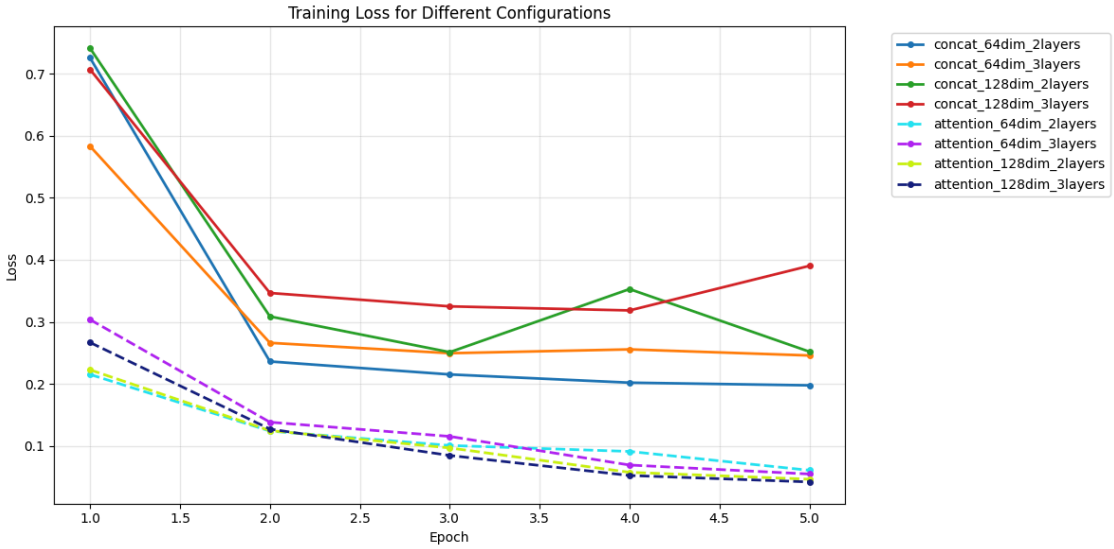
Recall@K, Precision@K, NDCG@K, and MAP@K across epochs, accompanied by a smooth and steady decrease in training loss.

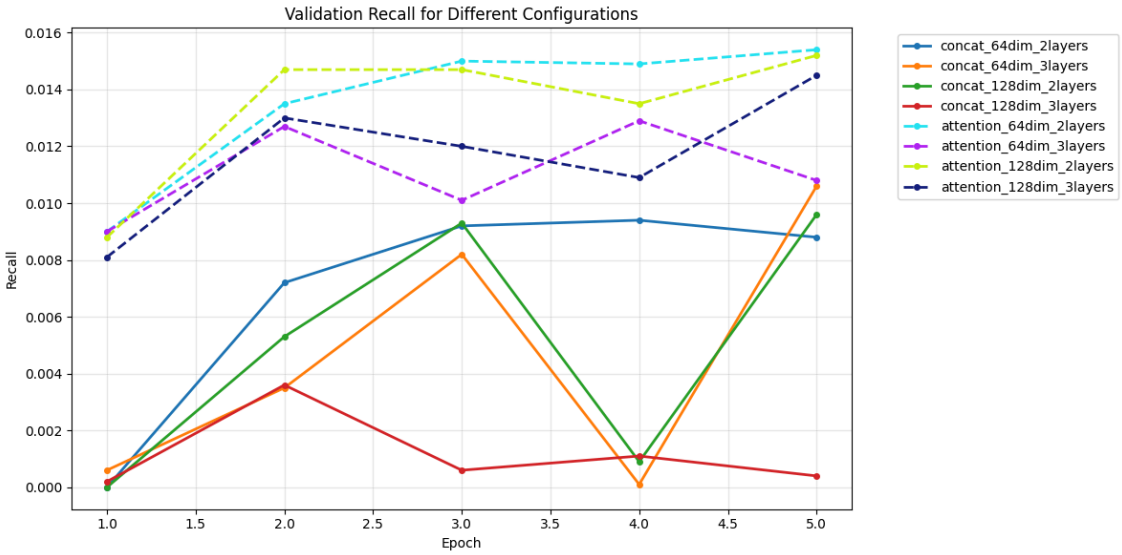


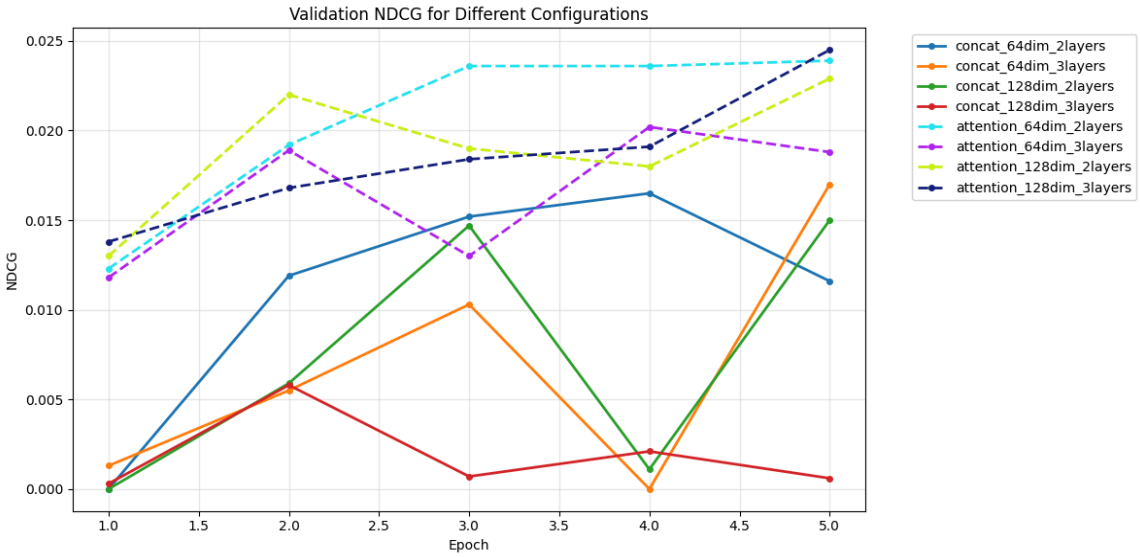


MBGCN Experiments

The experiments with such configurations, as behaviour blocks' fusion type, user embeddings' dimensionality, and number of layers were conducted in MBGCN architecture. The sampling of interactions by reducing the number of items to 100K out of 1M was performed due to huge time and memory consumption of model training. Only two types of relationships were used without differentiating by 'is_organic' flag: likes and listens. Also, the item embeddings were initialized randomly and did not participate in backpropagation. These actions can significantly impact the quality of the trained model. Nevertheless, the results of the training can be used to estimate correlation between the model quality and different configurations. The Recall@10 and NDCG@10 were used as validation metrics. The results of model evaluation on different configurations are shown below.


Training Loss for Different Configurations
Loss
Epoch
concat_64dim_2layers
concat_64dim_3layers
concat_128dim_2layers
concat_128dim_3layers
attention_64dim_2layers
attention_64dim_3layers
attention_128dim_2layers
attention_128dim_3layers


Validation Recall for Different Configurations
Recall
Epoch
concat_64dim_2layers
concat_64dim_3layers
concat_128dim_2layers
concat_128dim_3layers
attention_64dim_2layers
attention_64dim_3layers
attention_128dim_2layers
attention_128dim_3layers


Validation NDCG for Different Configurations
NDCG
Epoch
concat_64dim_2layers
concat_64dim_3layers
concat_128dim_2layers
concat_128dim_3layers
attention_64dim_2layers
attention_64dim_3layers
attention_128dim_2layers
attention_128dim_3layers

The attention fusion between behaviour blocks definitely outperform concatenation fusion. Also, the better performance is observed, when the user embedding dimension is higher.

Work distribution

- Bulat Sharipov: LightGCN x MBGCN experiment
- Danil Fathutdinov: MBGCN Experiments
- Dinar Yakupov: LightGCN training

Plan for next week

- Generate new hypothesis for experiments with given results
- Create a uniform evaluation script
- Integrating new models into recommendation web-service