Movie Recommender System with Graph Neural Networks

Diploma Thesis
Athanasiou Ioannis

MARCH 2023

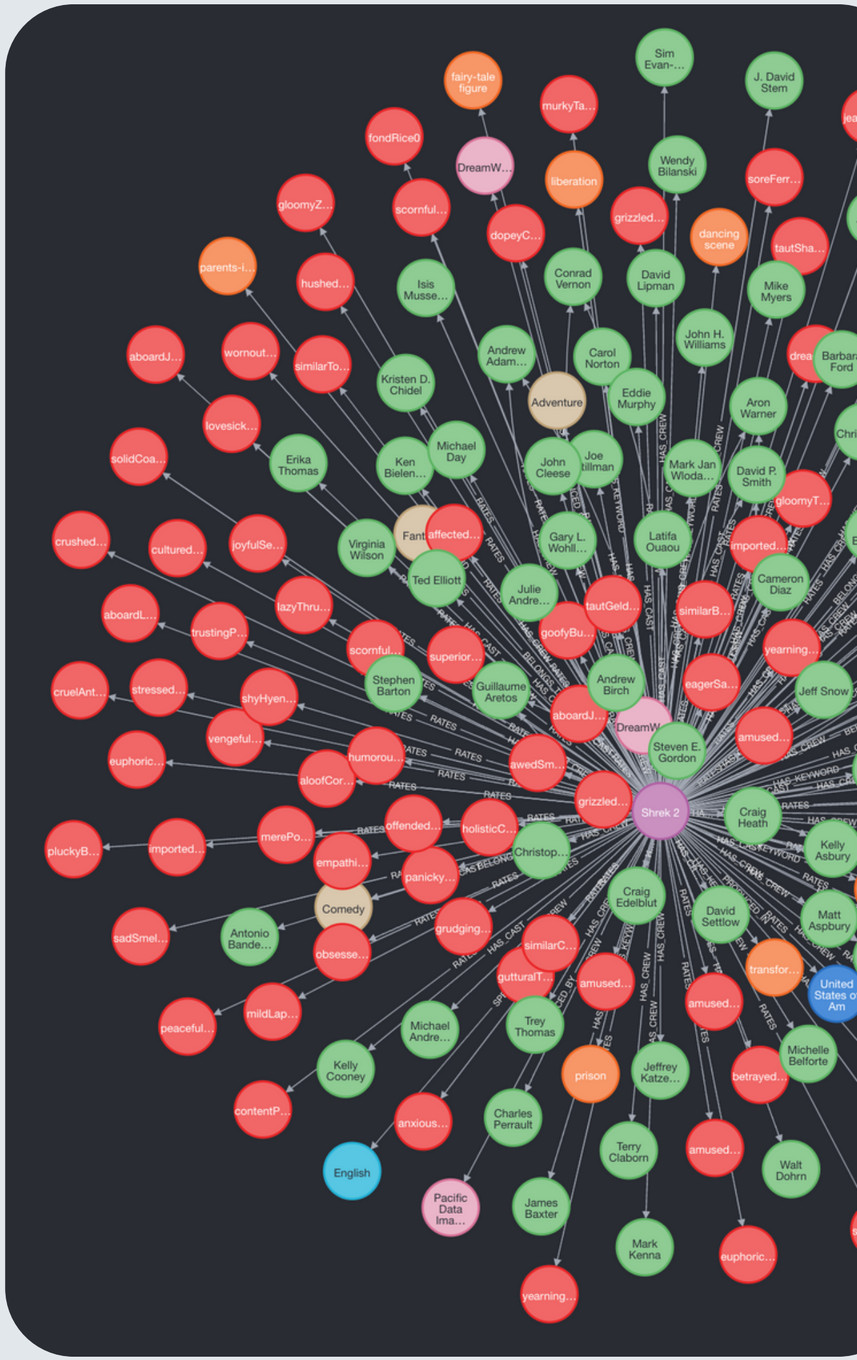


Table of Contents

1. Introduction

1. Introduction

Target

The development of a movie recommender system

Source code

https://github.com/John-Atha/diploma

Approach

Predict the exact rating a user would give to a movie in the range [0, 5]

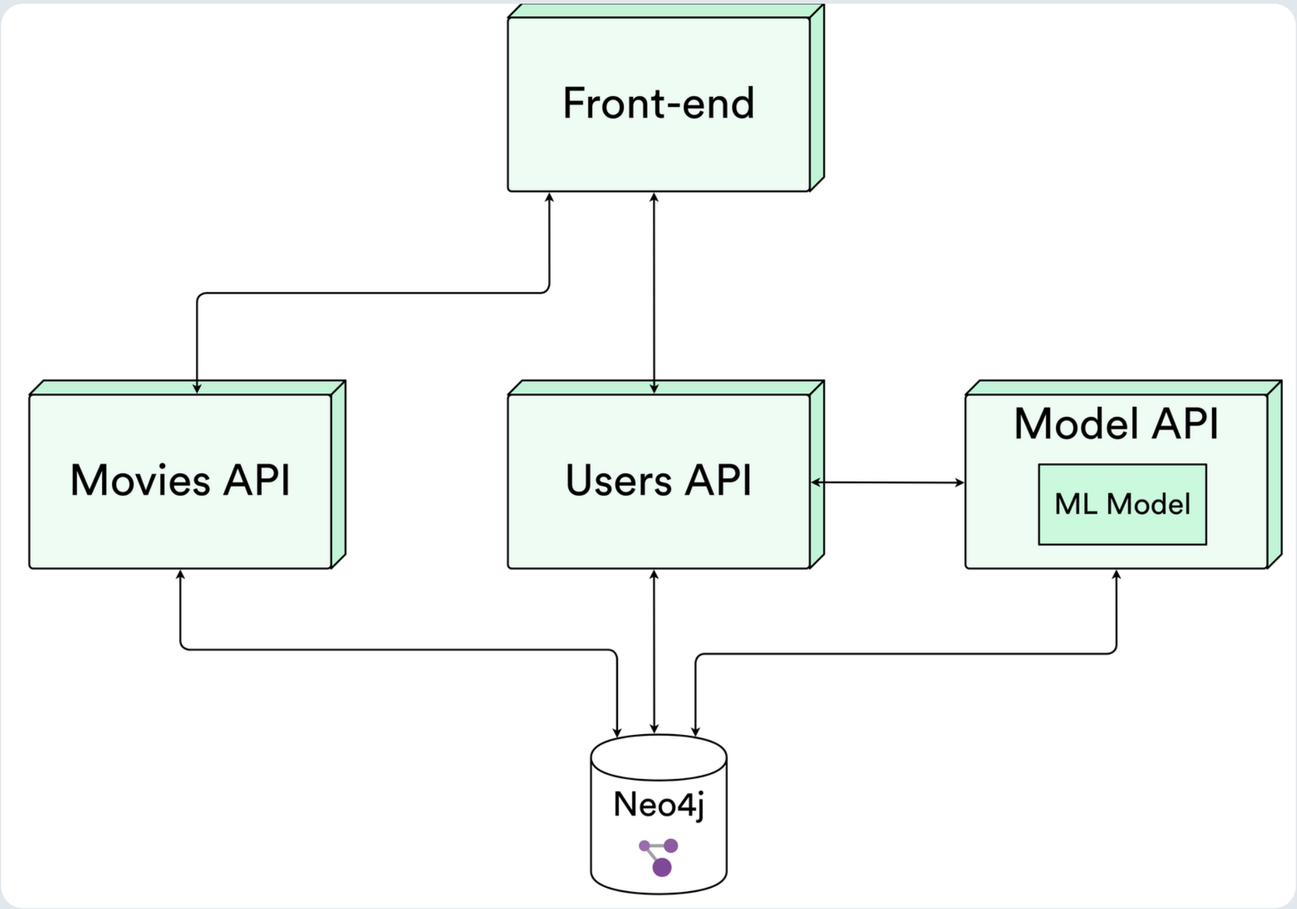
Corresponding ML task

Link weight prediction on the bipartite graph of users and movies, where the edges correspond to ratings

2. Overview

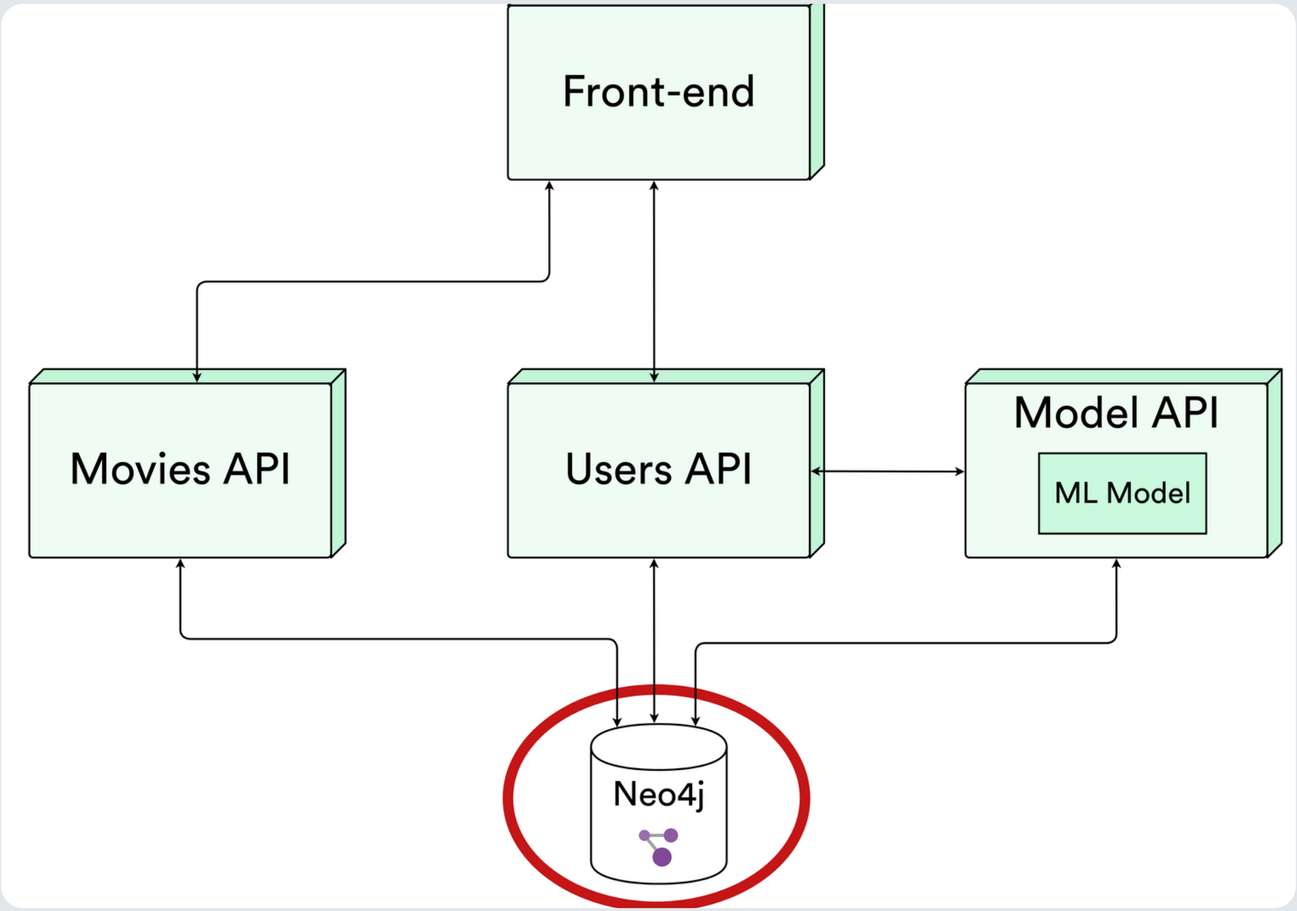
2. Overview

Build a whole web app where users can explore movies and receive personalized recommendations



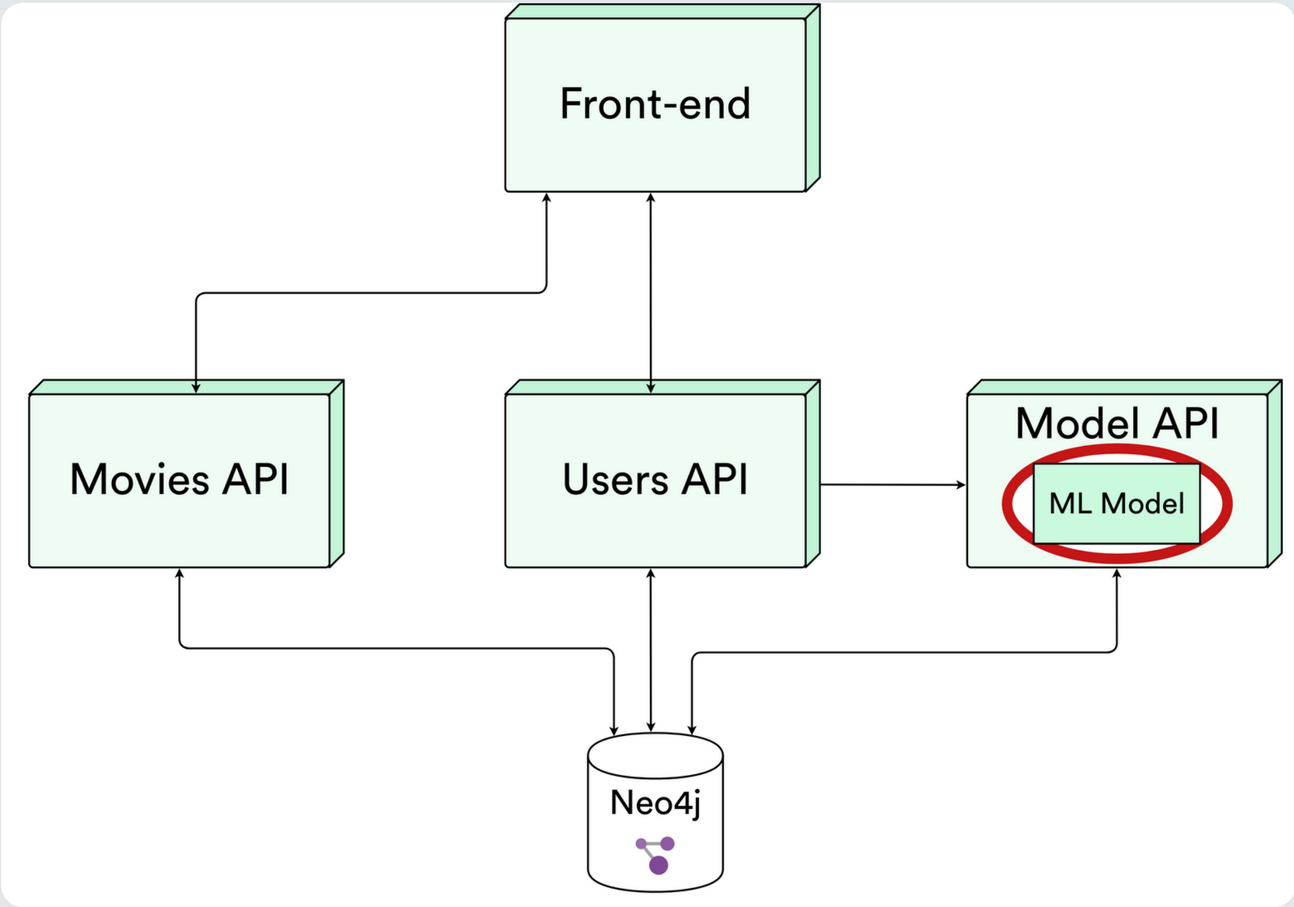
2. Overview

Store The Movies Dataset into a Graph Database System, called Neo4j



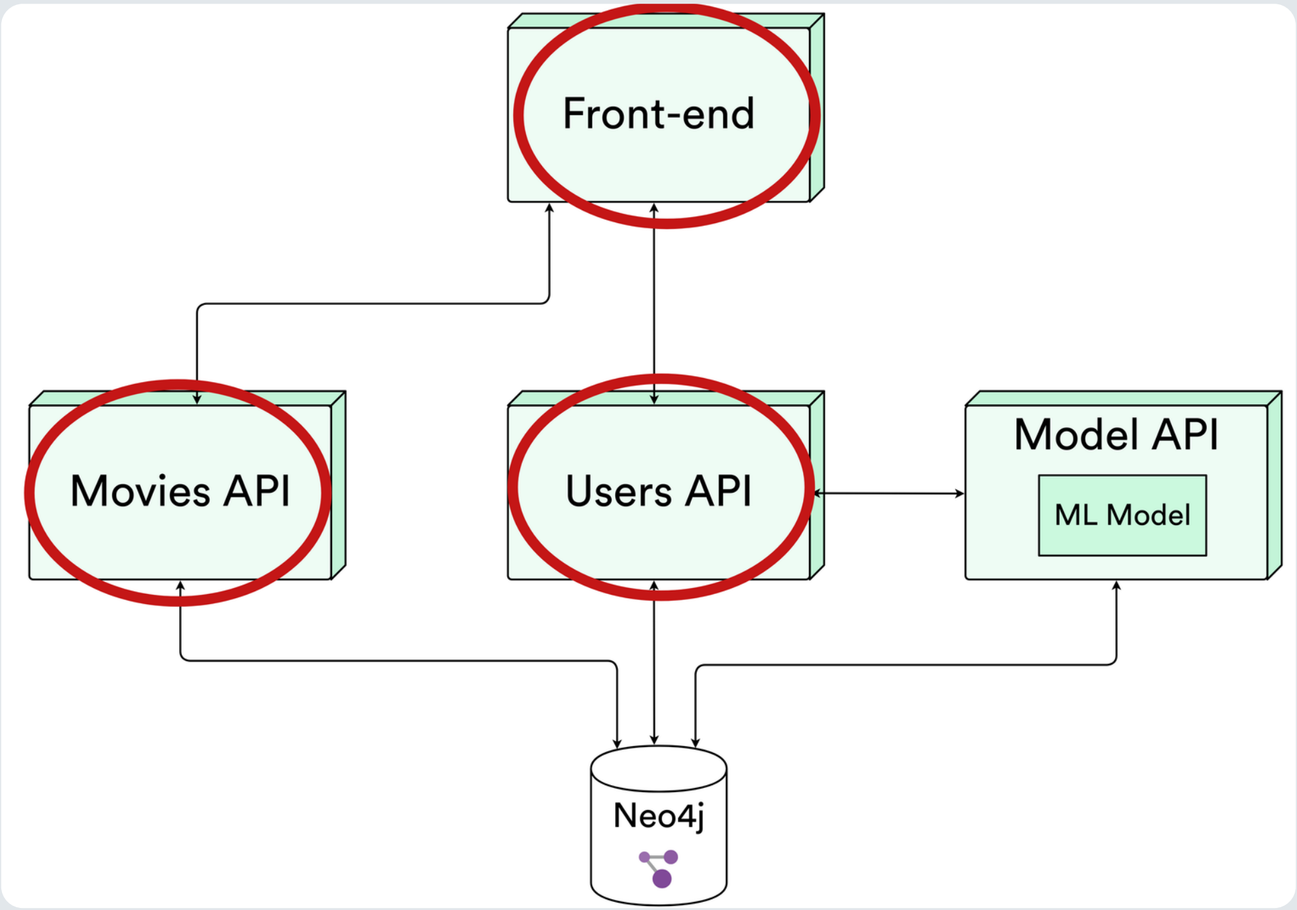
2. Overview

Develop a machine learning model with Graph Neural Networks and perform numerous experiments on solving the task



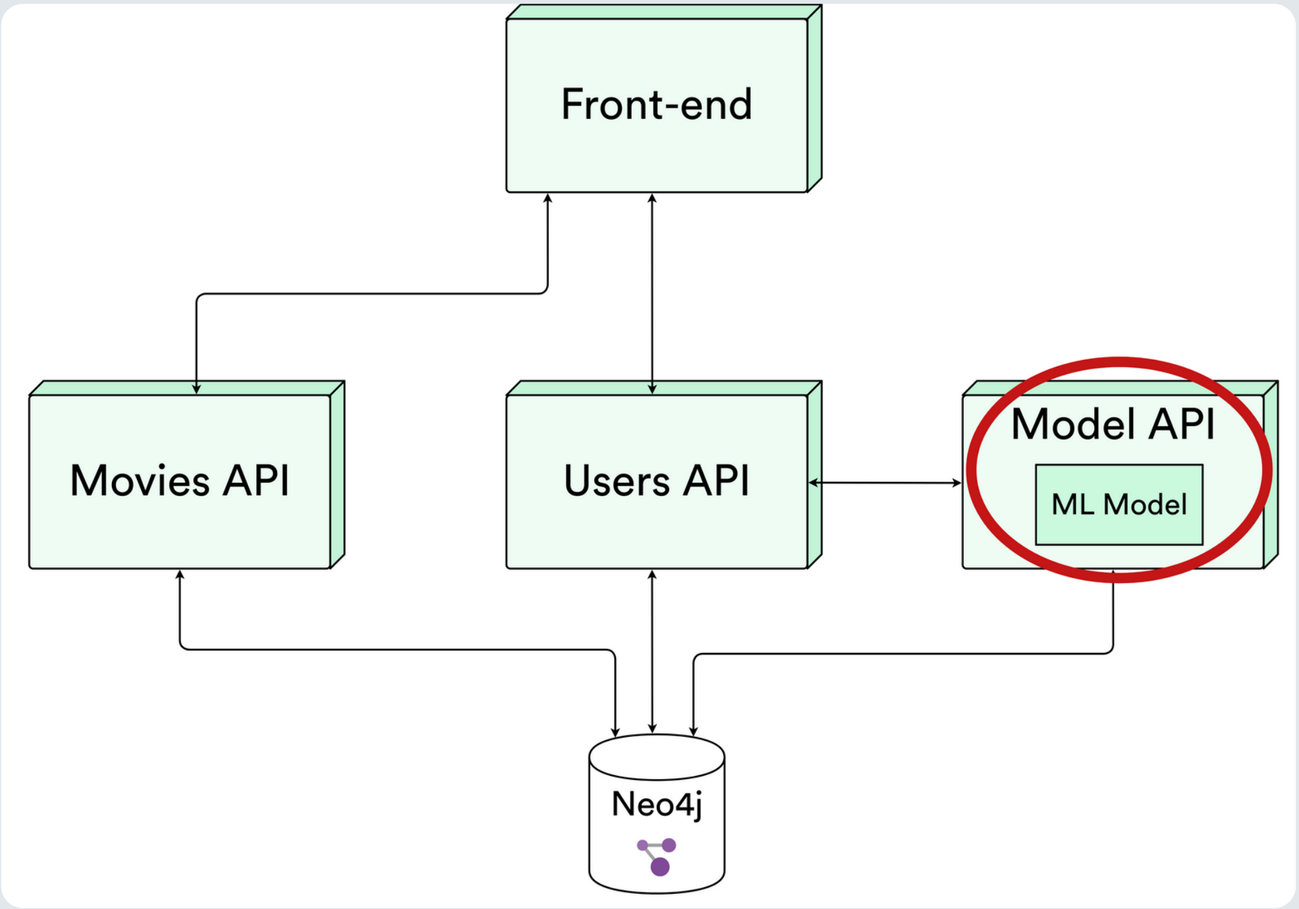
2. Overview

Develop REST APIs and a front-end interface that allows access to the data and the expansion of the dataset with new ratings



2. Overview

Integrate the machine-learning model that performs the predictions into the web app through a dedicated REST API

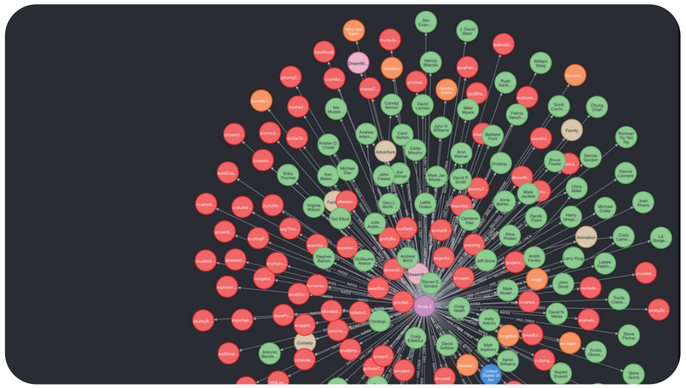


3. Theoretical background

3. Theoretical background > Graphs

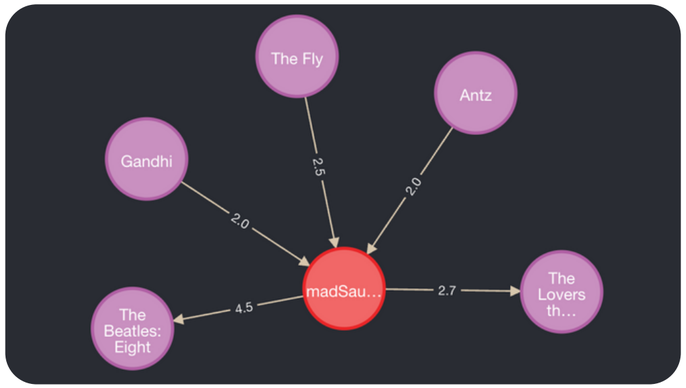
Graphs

Collections of objects, called nodes, and the relationships between them, called edges



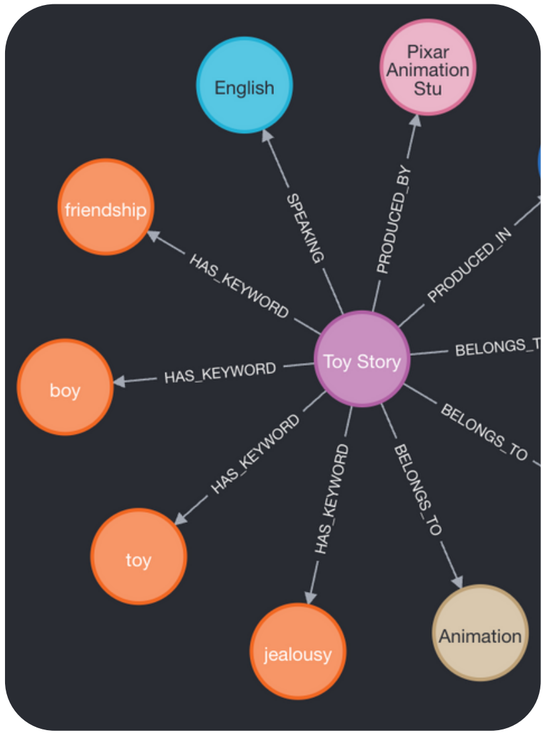
Weighted Graphs

A weight is assigned to each edge



Heterogeneous Graphs

Nodes and edges are imbued with types



3. Theoretical background > ML on Graphs

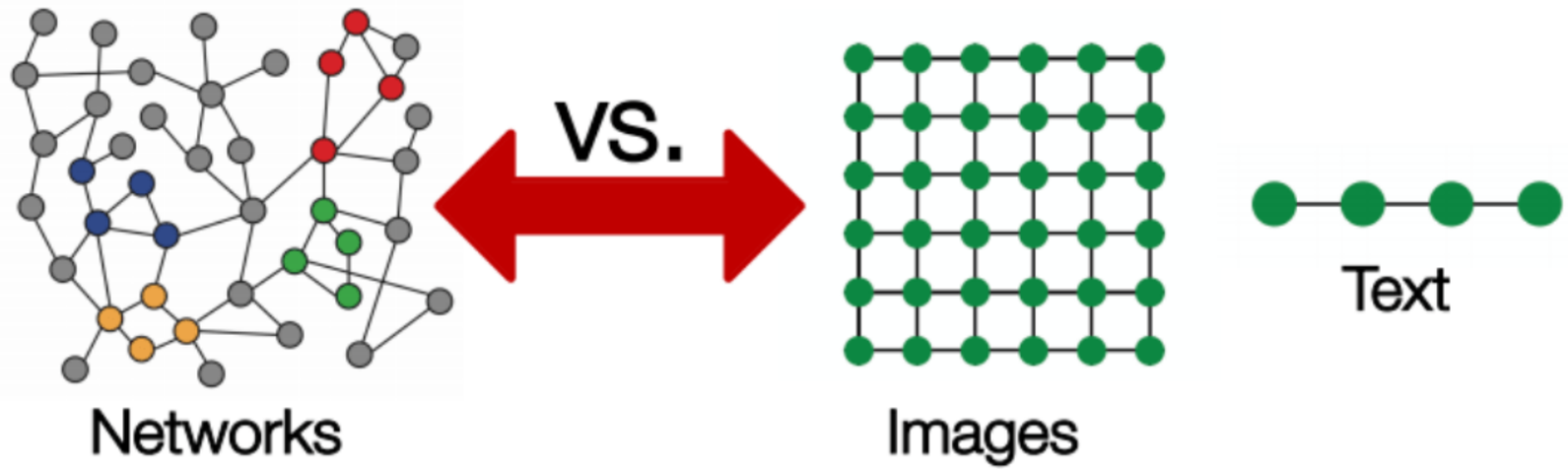
Introduction & Challenges

Motivation

- Rich structural information
- Various real-world applications

Main challenge

Graphs are irregular structures
How to represent them?



Source: http://web.stanford.edu/class/cs224w/

3. Theoretical background > ML on Graphs

Traditional Approaches

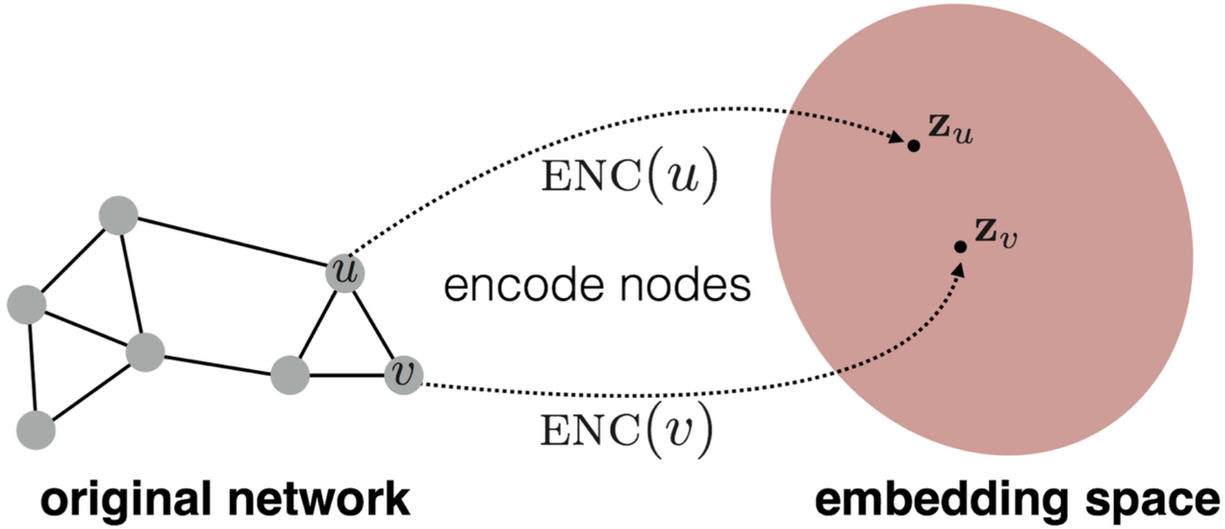
Hand-crafted features

Node-level features, such as the node degree

Used as input to traditional machine learning algorithms, such as SVMs

Node embeddings

Low-dimensional, real-valued vectors that encode the node's role in the graph



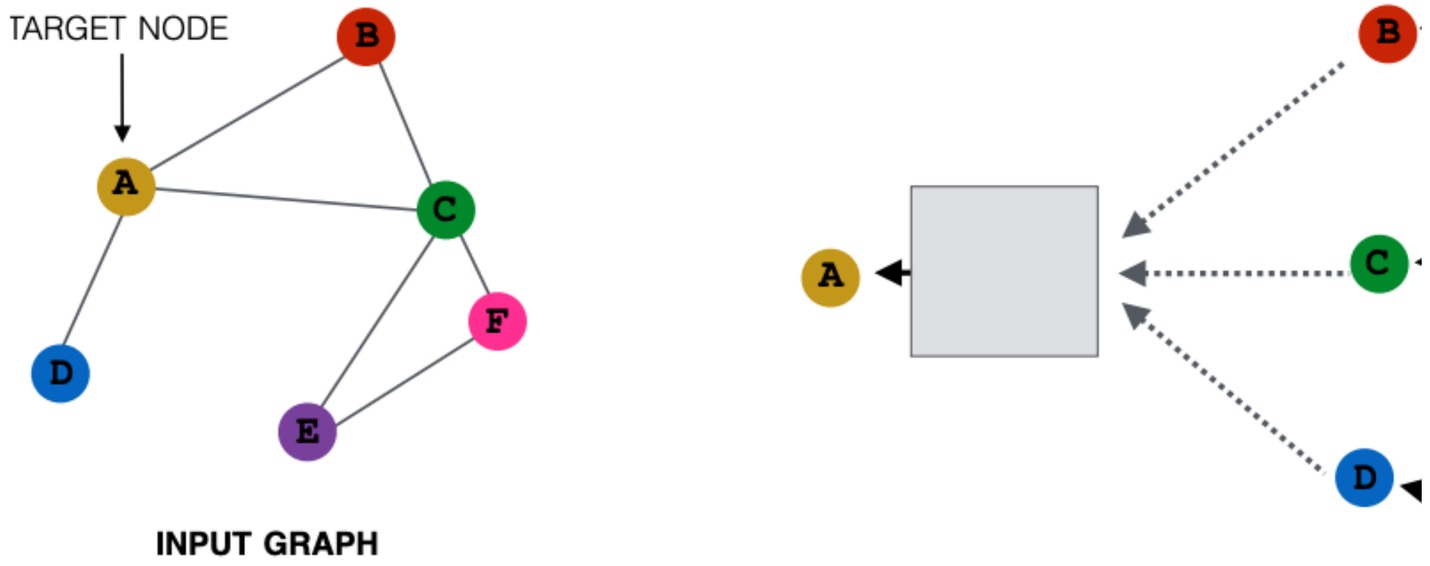
Source: http://web.stanford.edu/class/cs224w/

3. Theoretical background > Graph Neural Networks

The message passing framework of Convolutional Graph Neural Networks

Message Passing Framework - Intuition

The embedding of each node is mainly affected by the embeddings of its closest neighbours



Source: http://web.stanford.edu/class/cs224w/

Formalization

At each iteration k:

- Each node u has a hidden state
- Collect the hidden states of the node's neighbors, via an AGGREGATE function
- Use them to update the node's hidden state via an UPDATE function

After K steps, the hidden state of each node is considered its new embedding

The AGGREGATE and UPDATE are learned differentiable functions

4. The platform

4. The platform > Graph Database

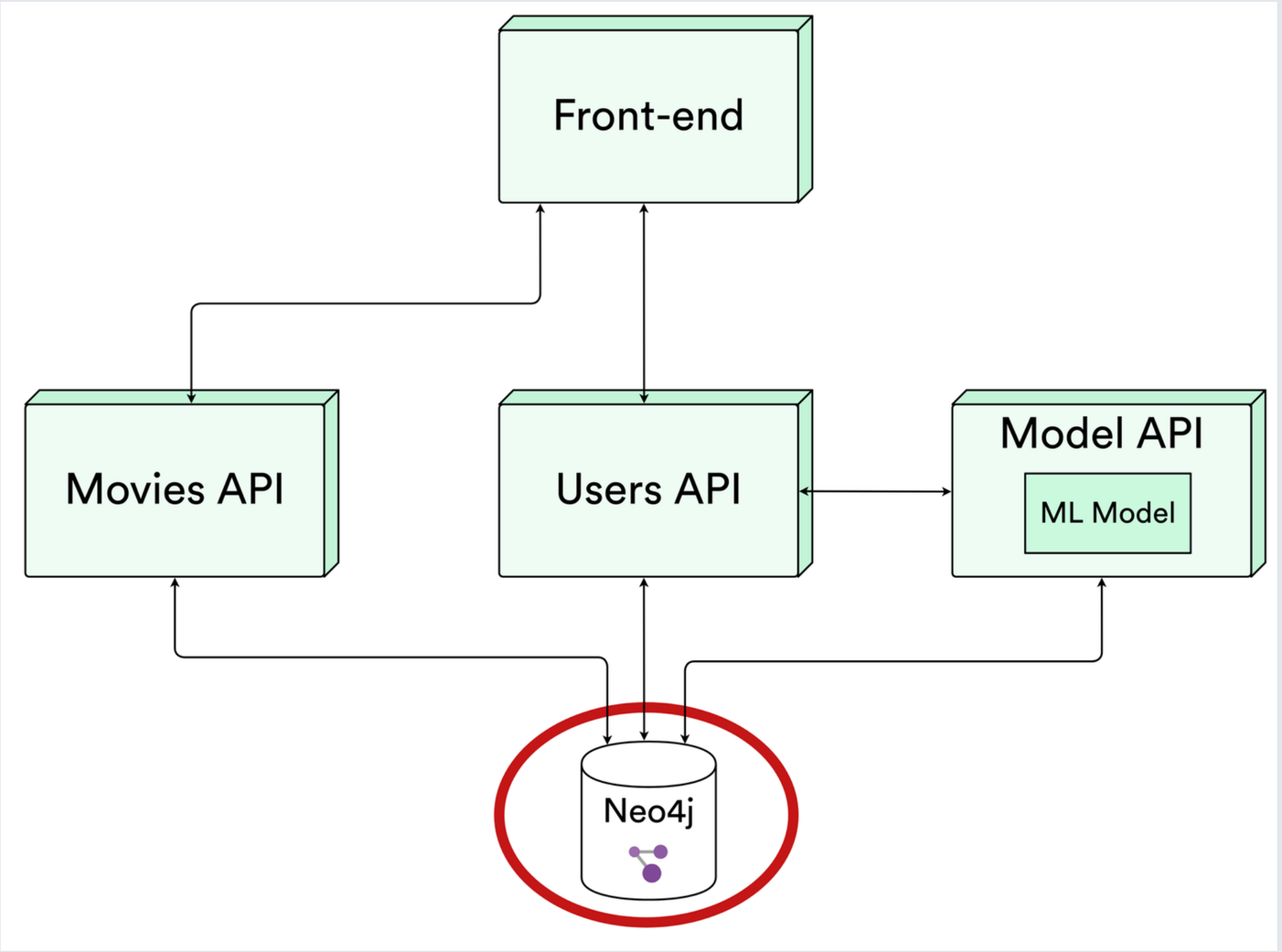
Introduction

Graph Database System

Data as nodes and edges instead of tables/documents

Neo4j

- Graph-optimized query language, called Cypher
- Neo4j Graph Data Science Library to apply common graph algorithms



4. The platform > Graph Database

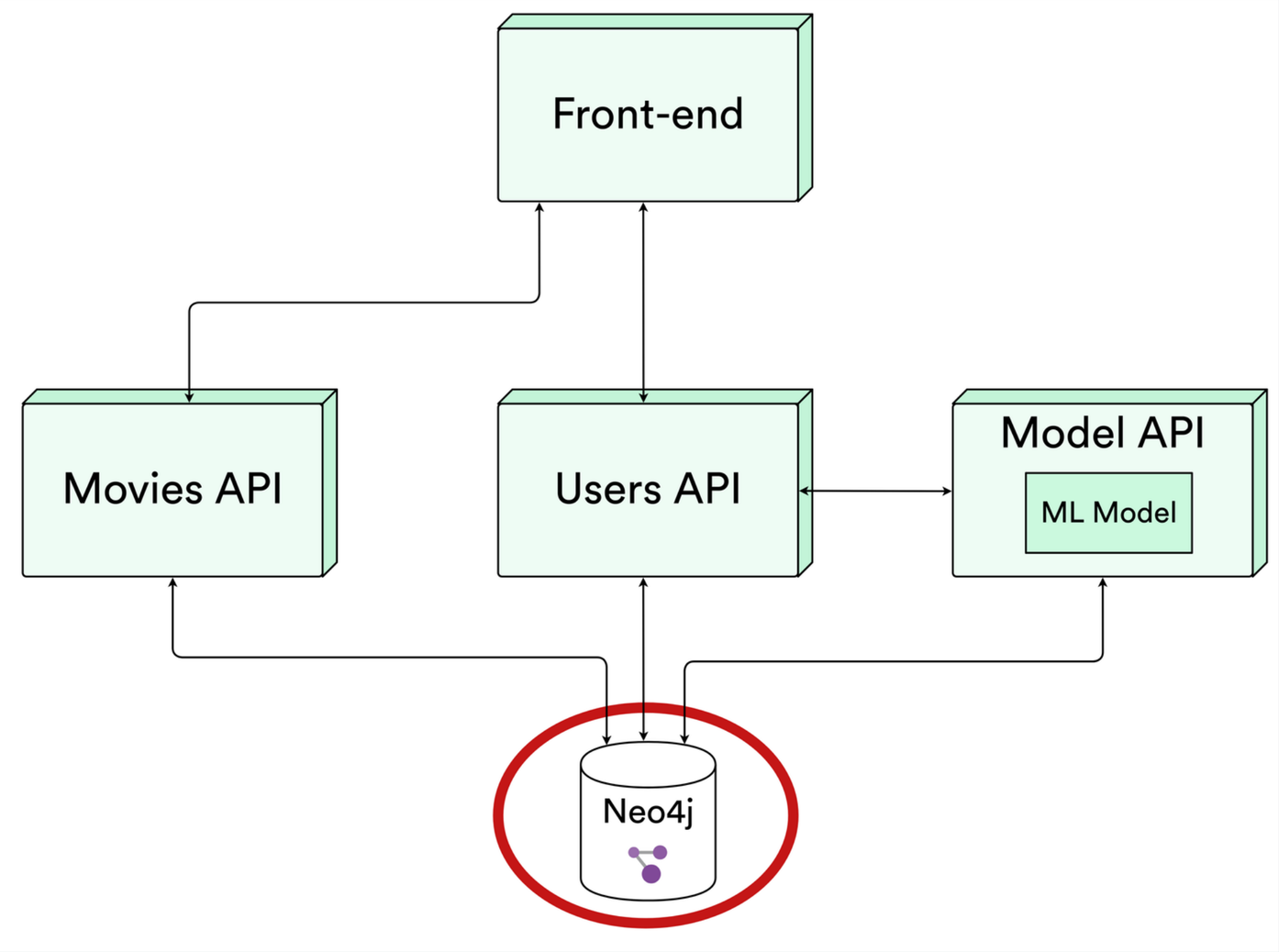
Introduction

The Movies Dataset

- Movies, users, and ratings of the MovieLens datasets
- Additional rich movies metadata, scrapped from IMDB and TMDB

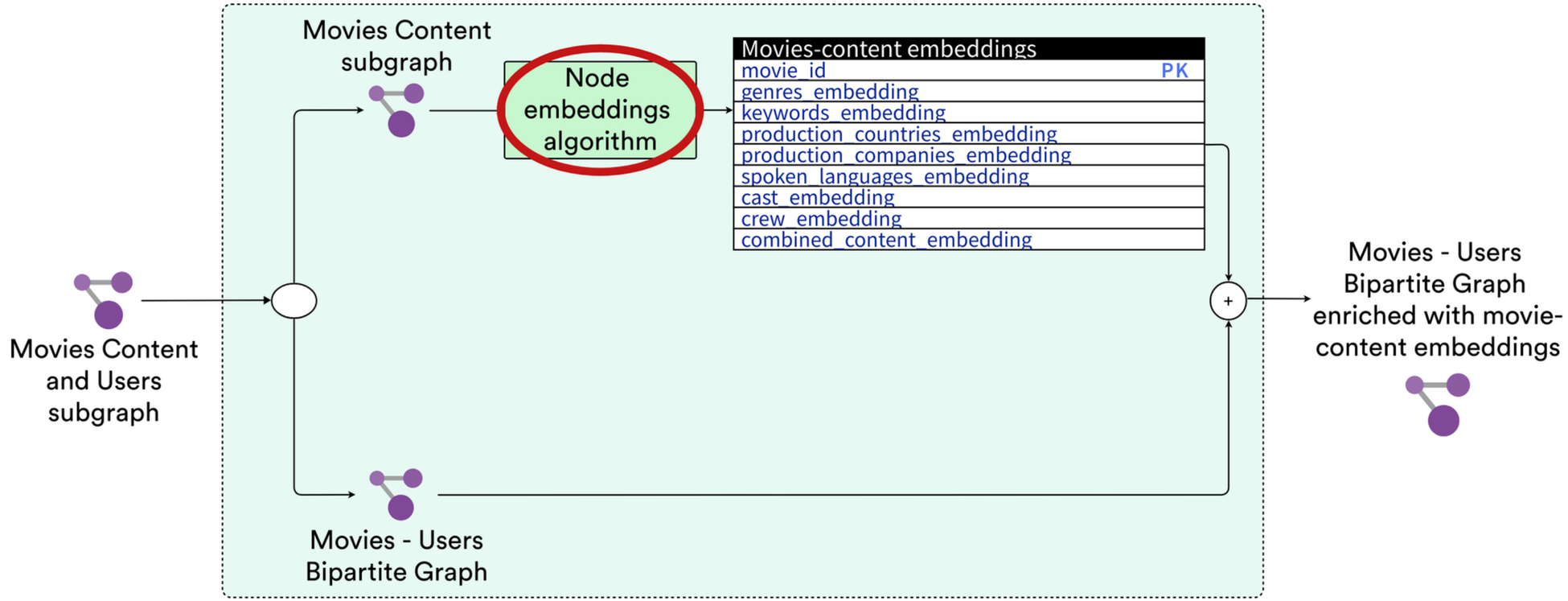
Modeling metadata as a graph

Entities related to movies, such as genres, and keywords are modeled as separate nodes



4. The platform > Graph Database

Encoding the movies metadata with node embeddings



Node embedding algorithms

- Node2Vec
- FastRP
- GraphSAGE

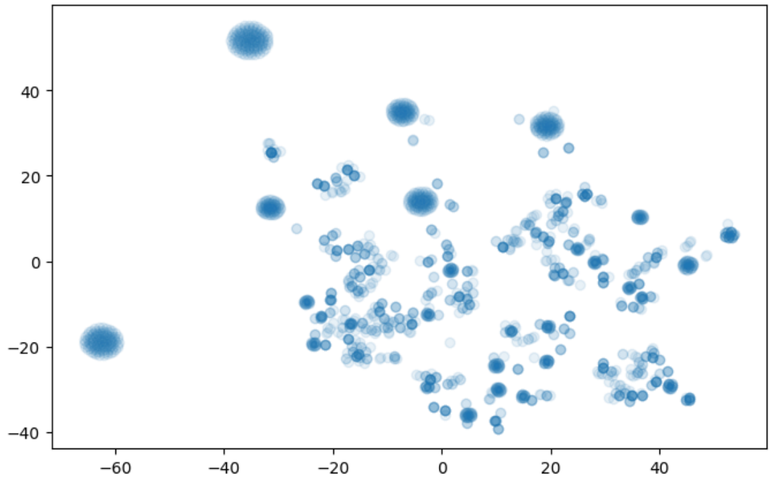
Movies-content Subgraph

- Whole movies-content subgraph
- Multiple separate bipartite subgraphs, each one containing movies, and another type of content-related nodes, such as genres

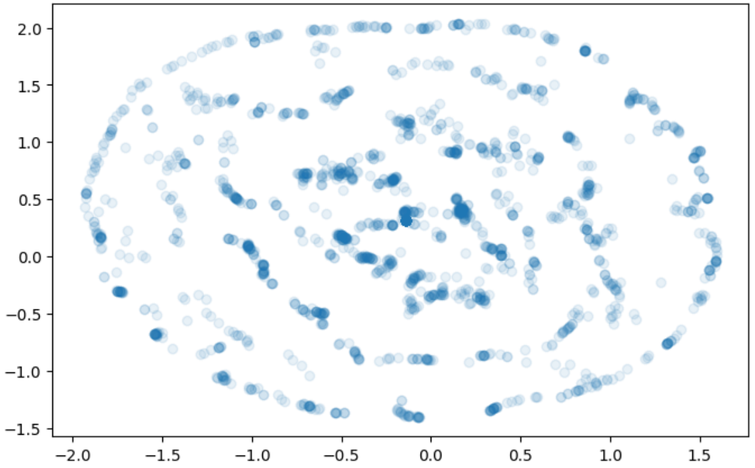
4. The platform > Graph Database

Encoding the movies metadata with node embeddings

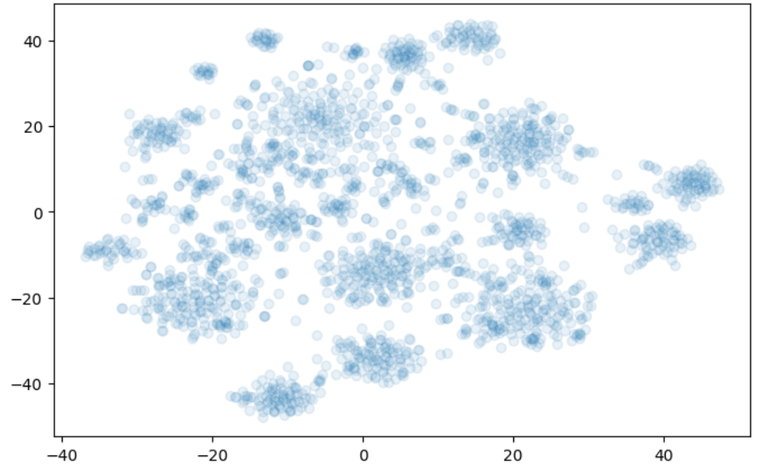
FastRP movie embeddings based on the **movies-keywords** bipartite graph



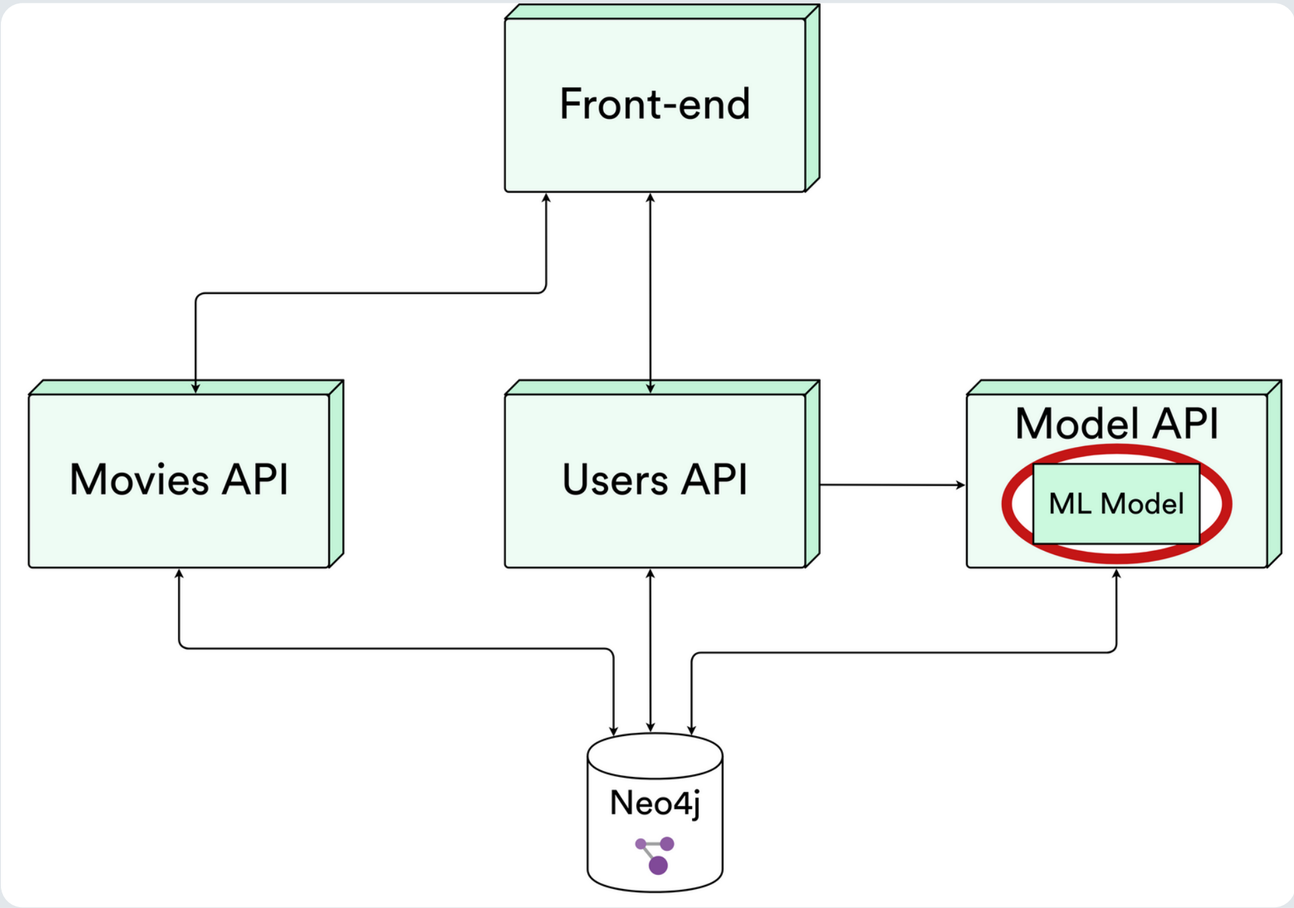
GraphSAGE movie embeddings based on the **whole movies-content** subgraph



Node2Vec movie embeddings based on the **production companies** subgraph



4. The platform > Model



4. The platform > Model

Architecture

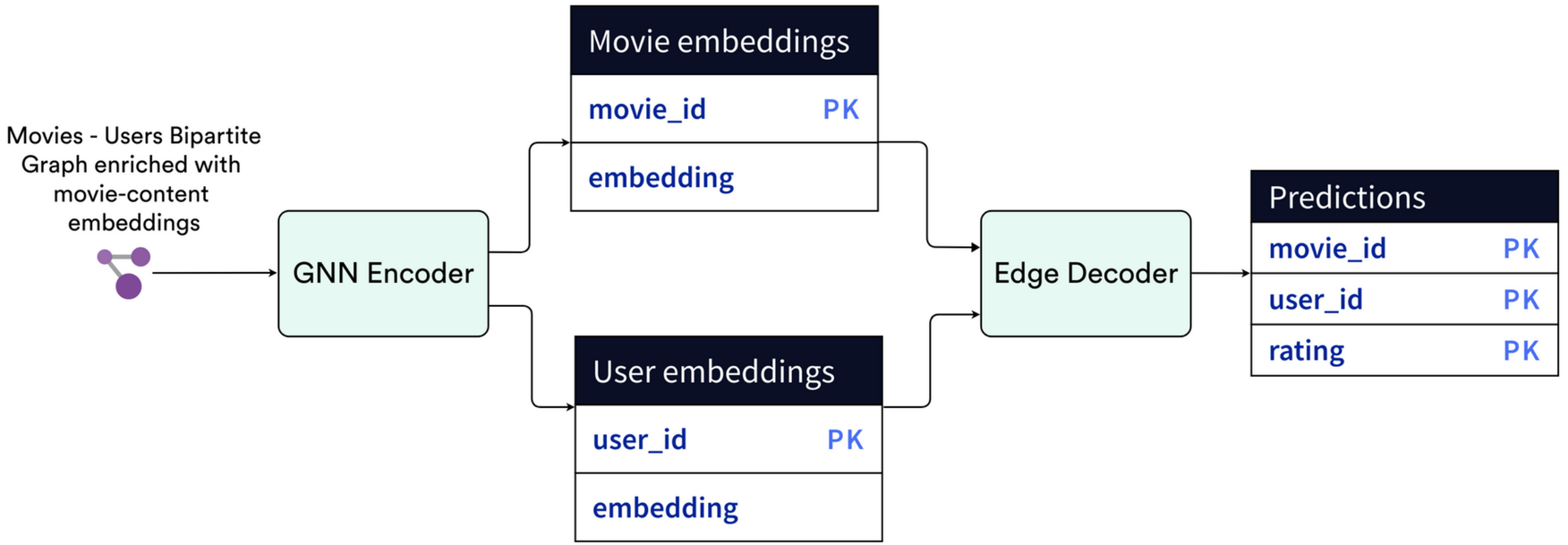
Main components

GNN Encoder

A GNN that generates node embeddings for movies and users

Edge Decoder

A feed-forward deep neural network that predicts the rating that a user will give to a movie



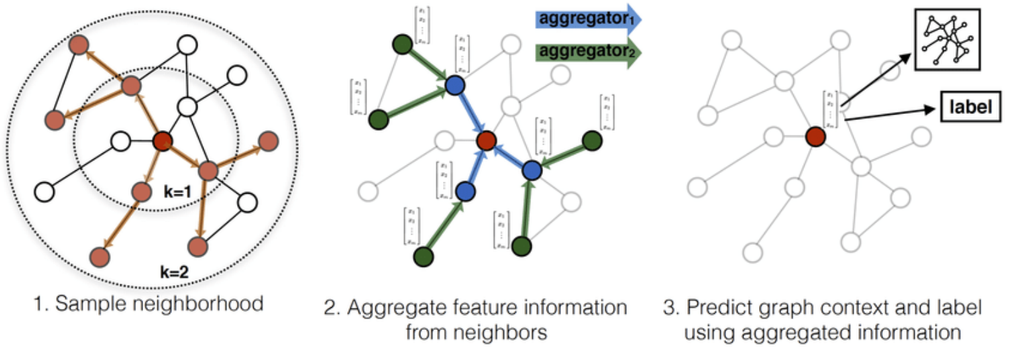
4. The platform > Model

The 2 GNN Architectures that performed better in our GNN Encoder

GraphSAGE

SAmple and aggreGatE

It **samples** the node's neighbors on the aggregation step, and **learns a weight matrix** to perform the aggregation and update steps



1. Sample neighborhood

2. Aggregate feature information from neighbors

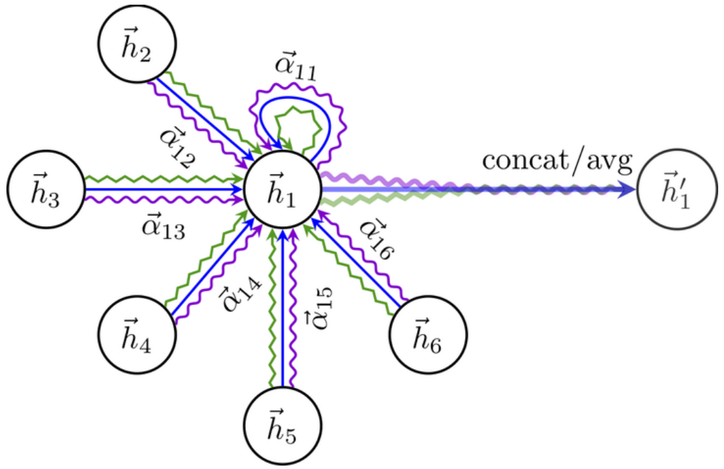
3. Predict graph context and label using aggregated information

https://arxiv.org/abs/1706.02216

GAT

Graph Attention Networks

Use a **self-attention** mechanism during the aggregation step



https://arxiv.org/abs/1710.10903

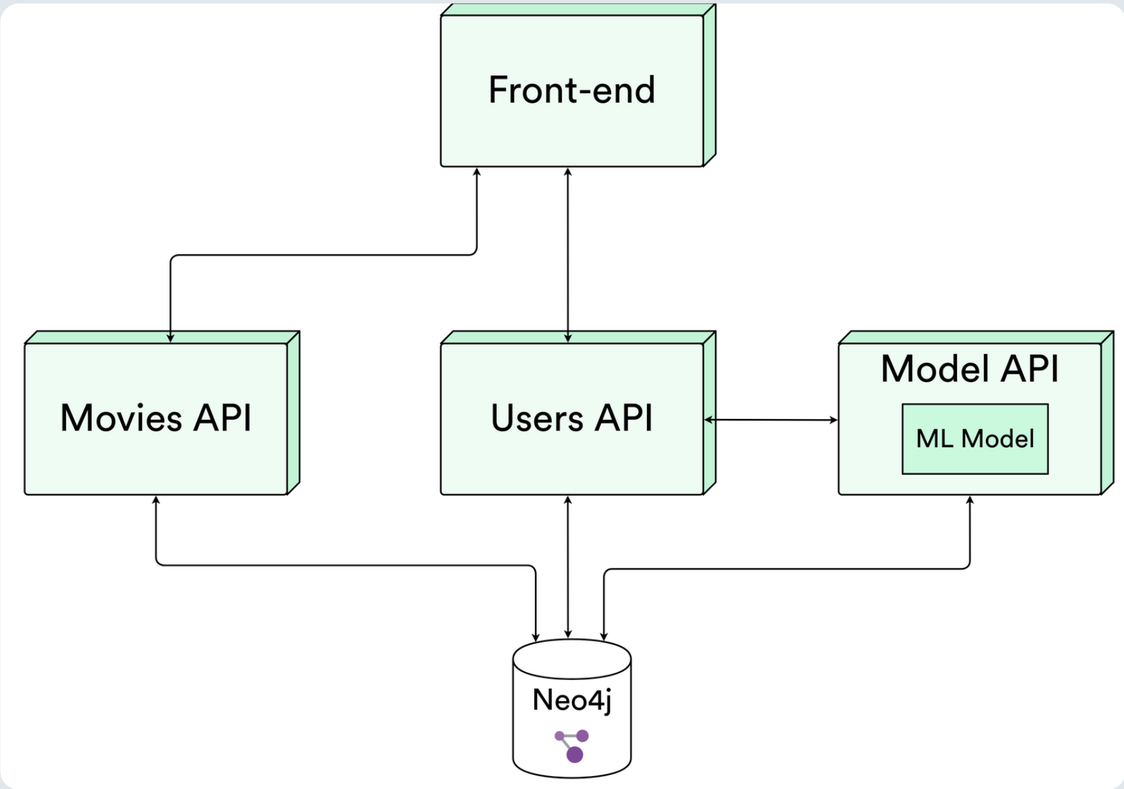
4. The platform > REST APIs & Front-end

Movies API

- Responses about movies and their metadata

Users API

- Responses about users, authentication, ratings
- Authentication middleware



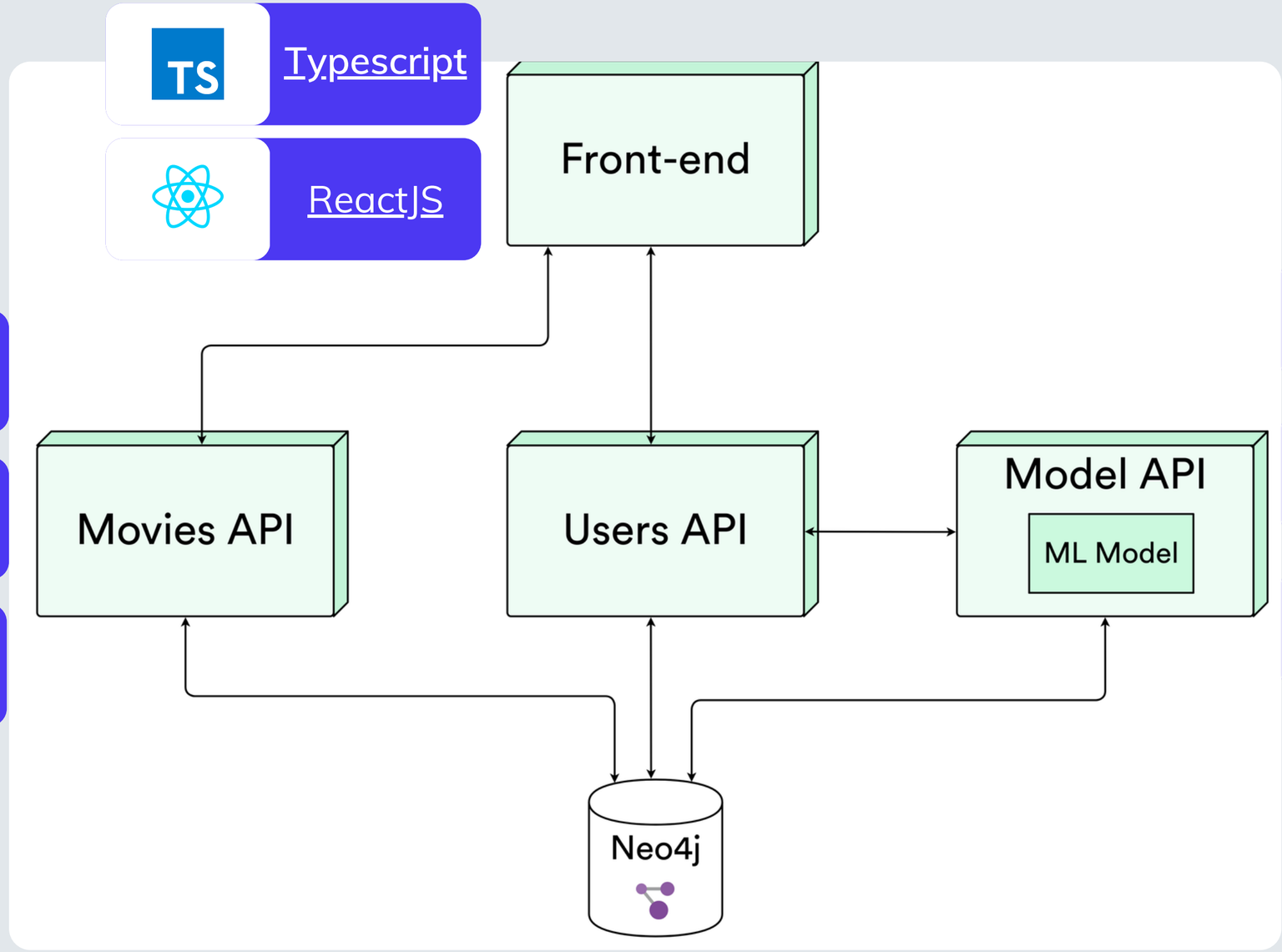
Front-end

- Easy navigation
- Insightful visualizations of the underlying graph
- Personalized recommendations

Model API

- Generate predictions for a user
- Recommend top k movies to a user
- Re-train the model
- Re-fresh the in-memory graph

4. The platform > Technologies



Typescript

ReactJS

Typescript

Node JS

Express JS

Python

Pytorch Geometric

Flask

4. The platform > Model API

Challenges in integrating the model into our platform

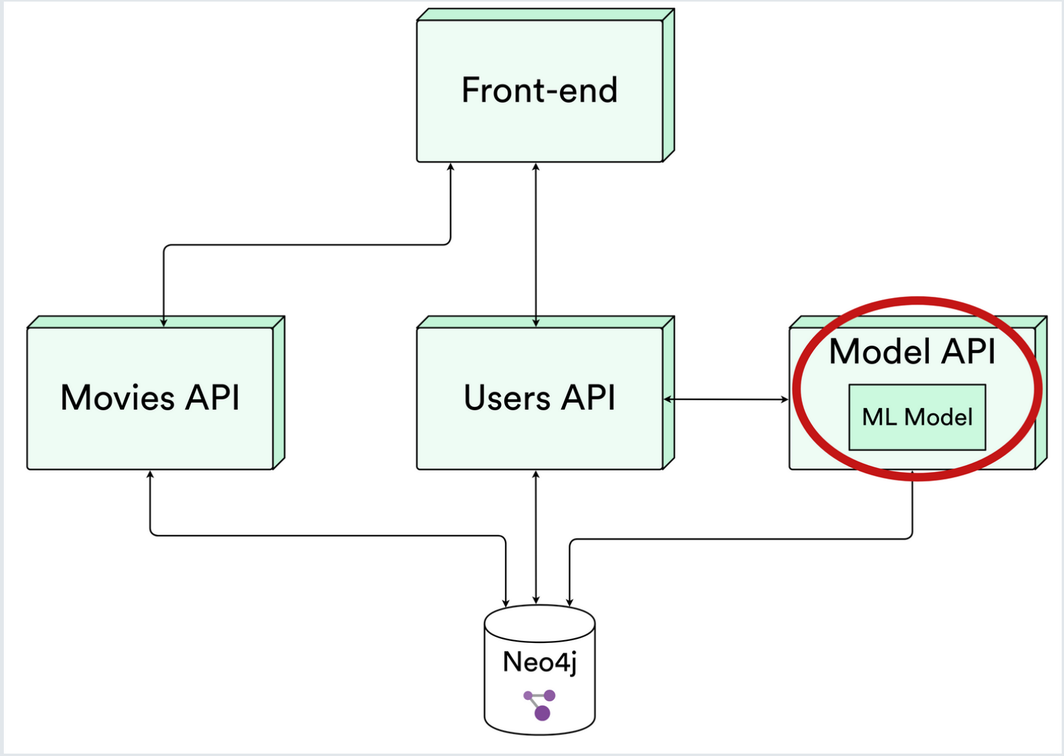
Model API availability

Challenge

Intense operations are **blocking** the Model API

Solution

Use **threads** to execute them in the background



4. The platform > Model API

Challenges in integrating the model into our platform

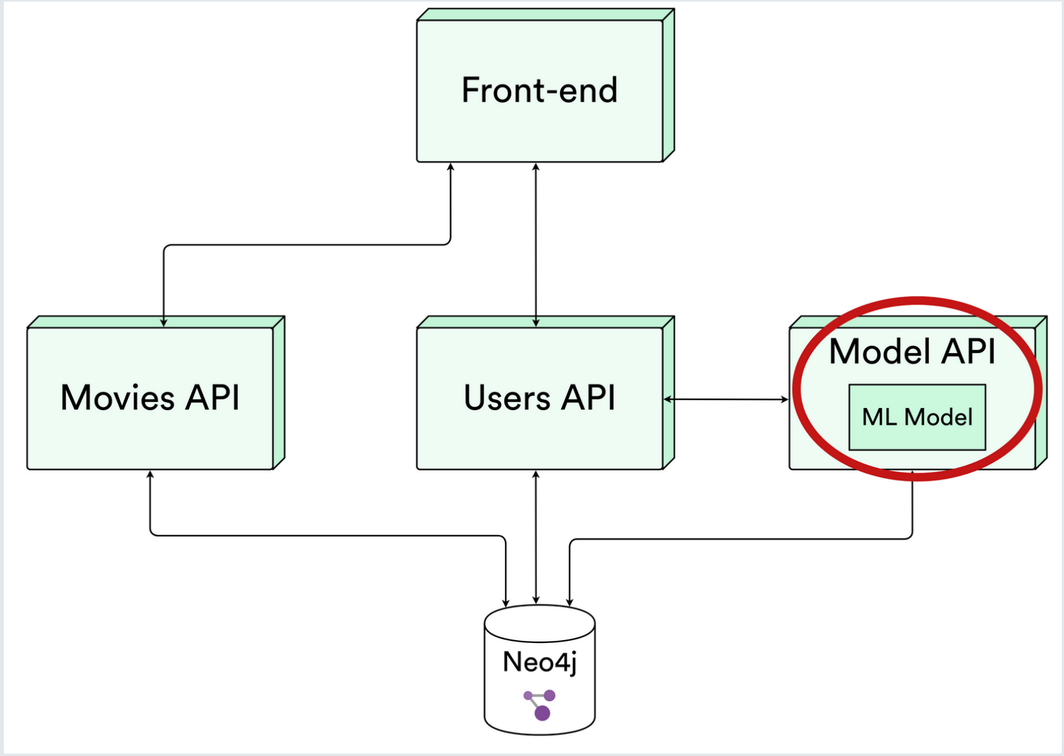
Handling new users without retraining

Challenge

- Model developed under the **transductive** setting
- **Weight** matrices are **dimensioned** based on the **initial graph**

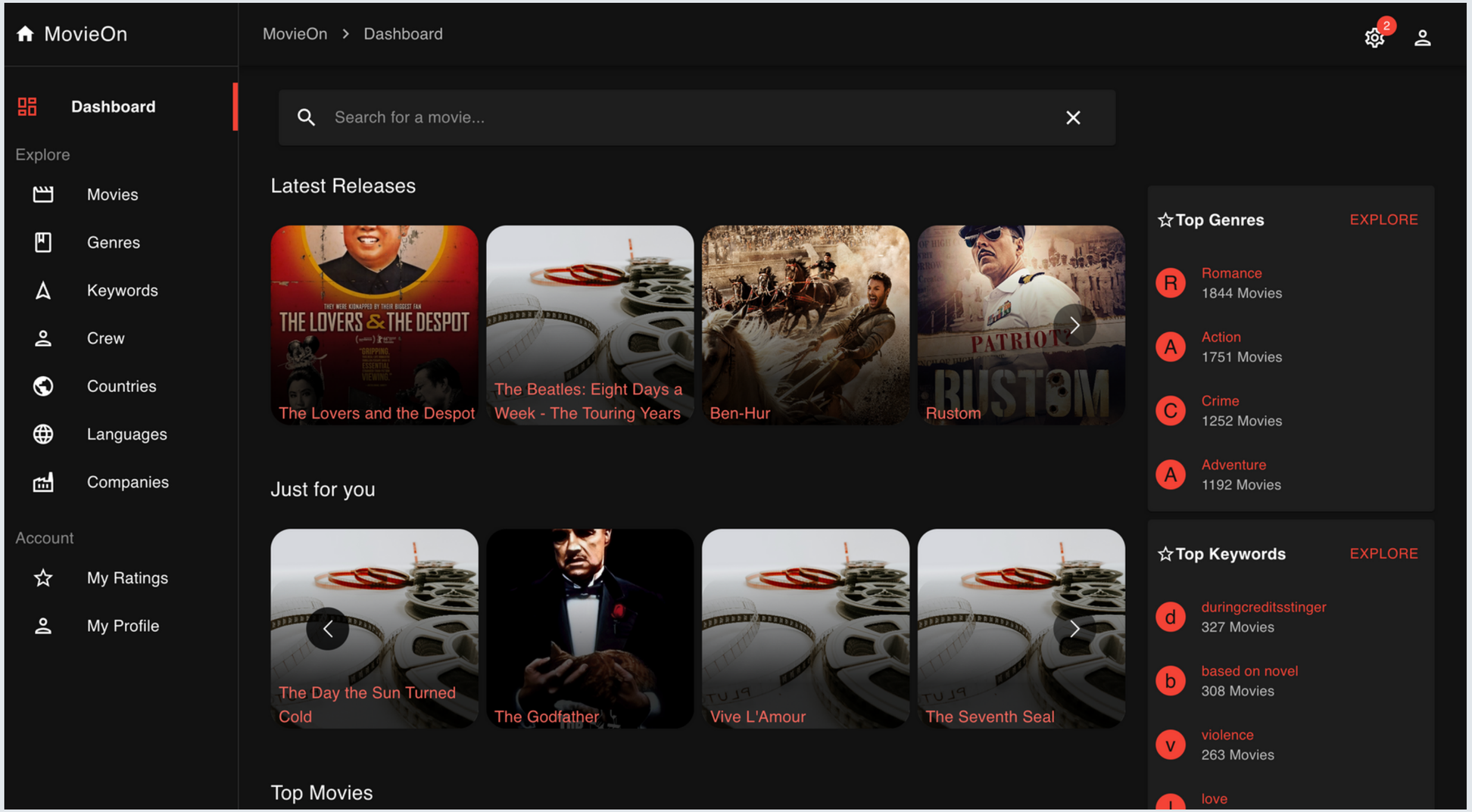
Solution

- **Add columns** to weight matrix **dynamically**
- **Initialize** them with the **average user**'s weights



4. The platform > Front-end

Landing page



5. Experiments

5. Experiments > Outline

Hyperparameters Groups

GNN Encoder Architecture

- GraphSAGE
- GAT
- GraphConv (k-dimensional GNNs)
- GIN

Movie-content utilization

- **Collaborative**-like approach
- **Combined** movies-content in one embedding
- **Separate** embedding for each related entity type

Movie-Content Node embeddings algorithm

- Fast Random Projection (FastRP)
- Node2Vec
- GraphSAGE

Other traditional hyperparameters

- Hidden layers of GNN Encoder
- Hidden layers of Edge Decoder
- Hidden channels

5. Experiments > Outline

Baseline & Evaluation

Evaluation

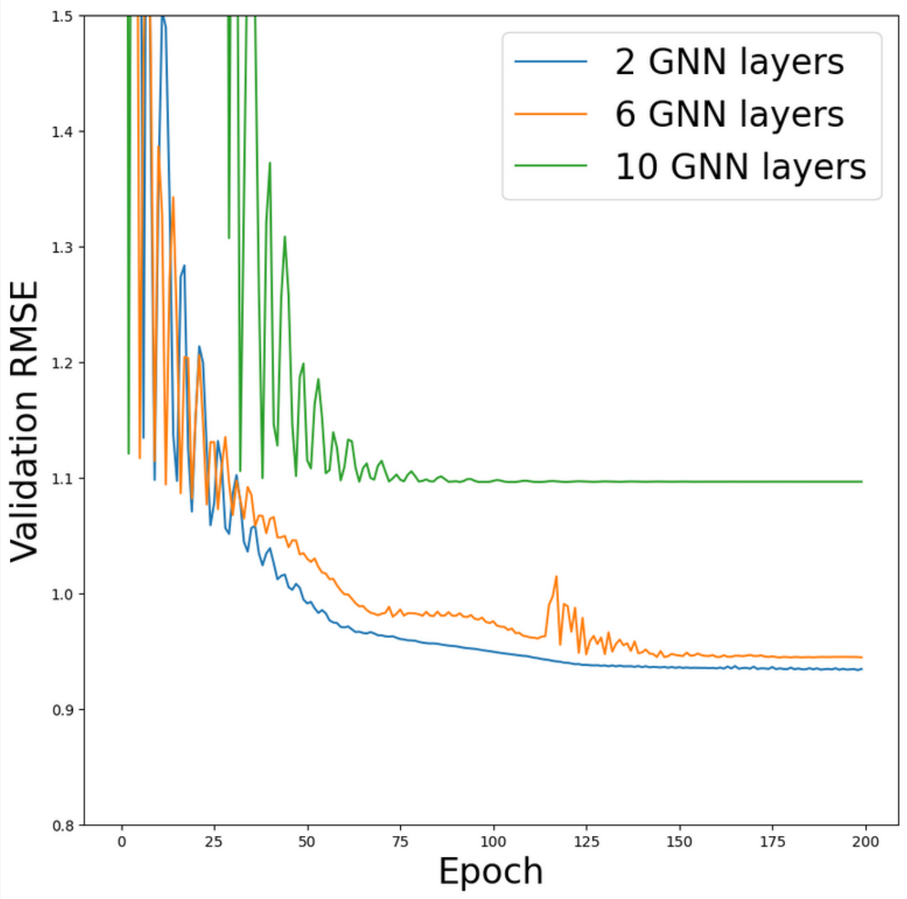
Root Mean Squared Error (**RMSE**)

Baseline

Graph-based **related work** achieves RMSE in the range [0.880, 0.961] on MovieLens 100K

5. Experiments > Most Significant

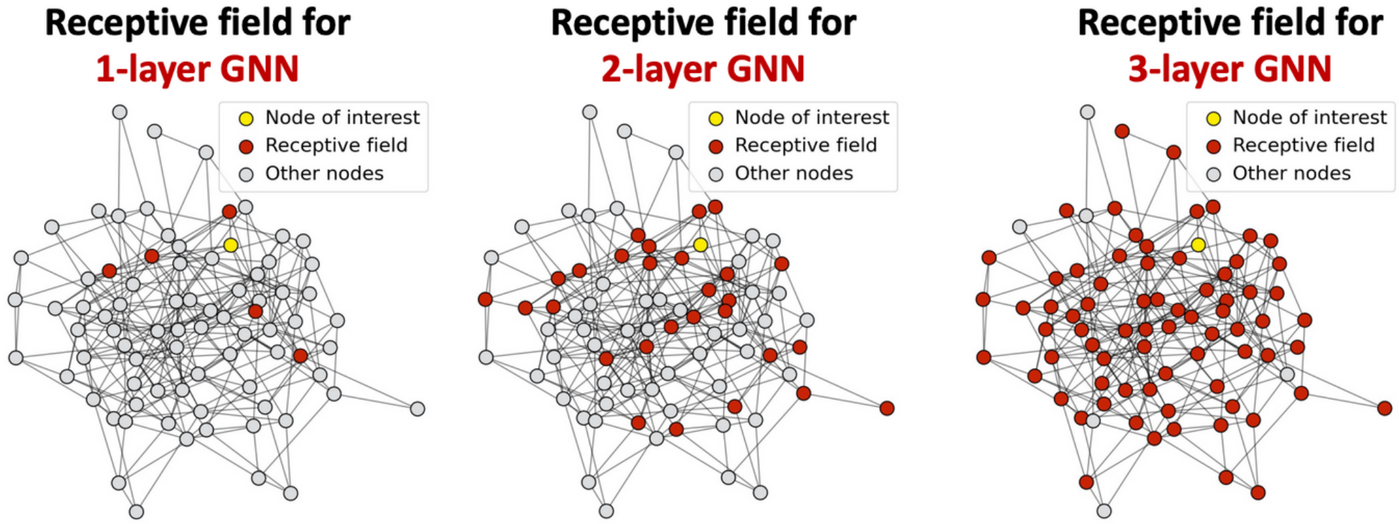
Validation RMSE on a simple GraphSAGE model



GNN layers number

Oversmoothing phenomenon

Shallow GNNs perform better

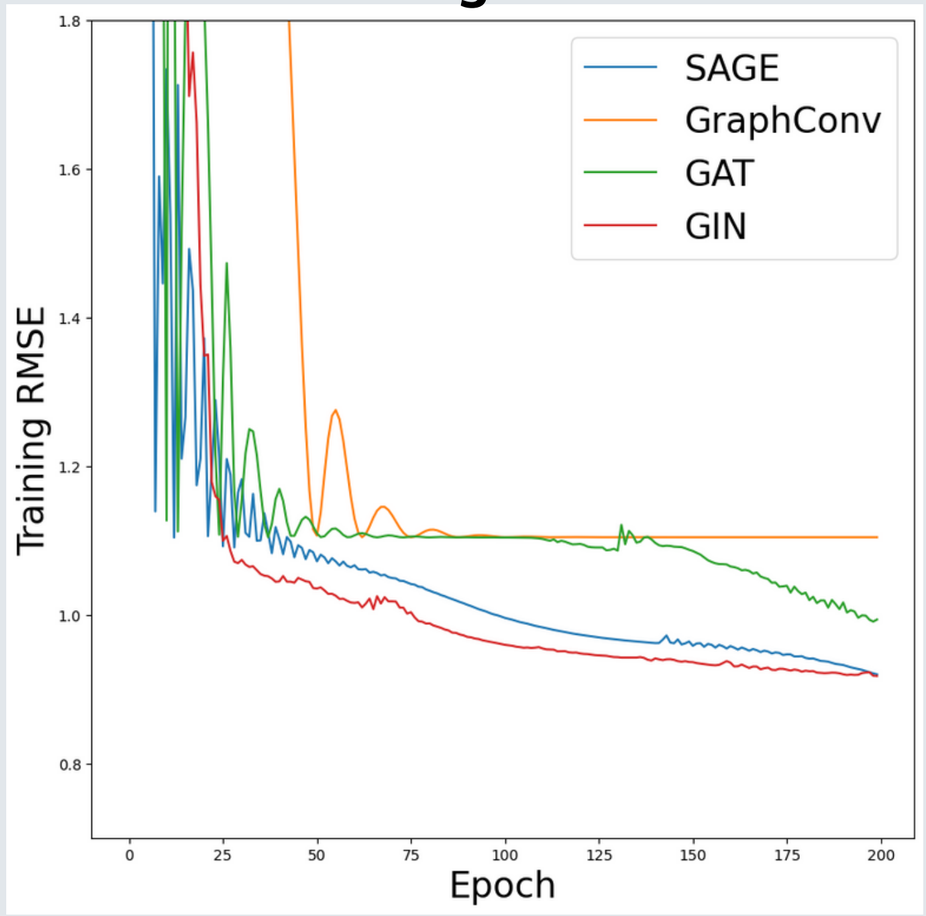


Source: http://web.stanford.edu/class/cs224w/

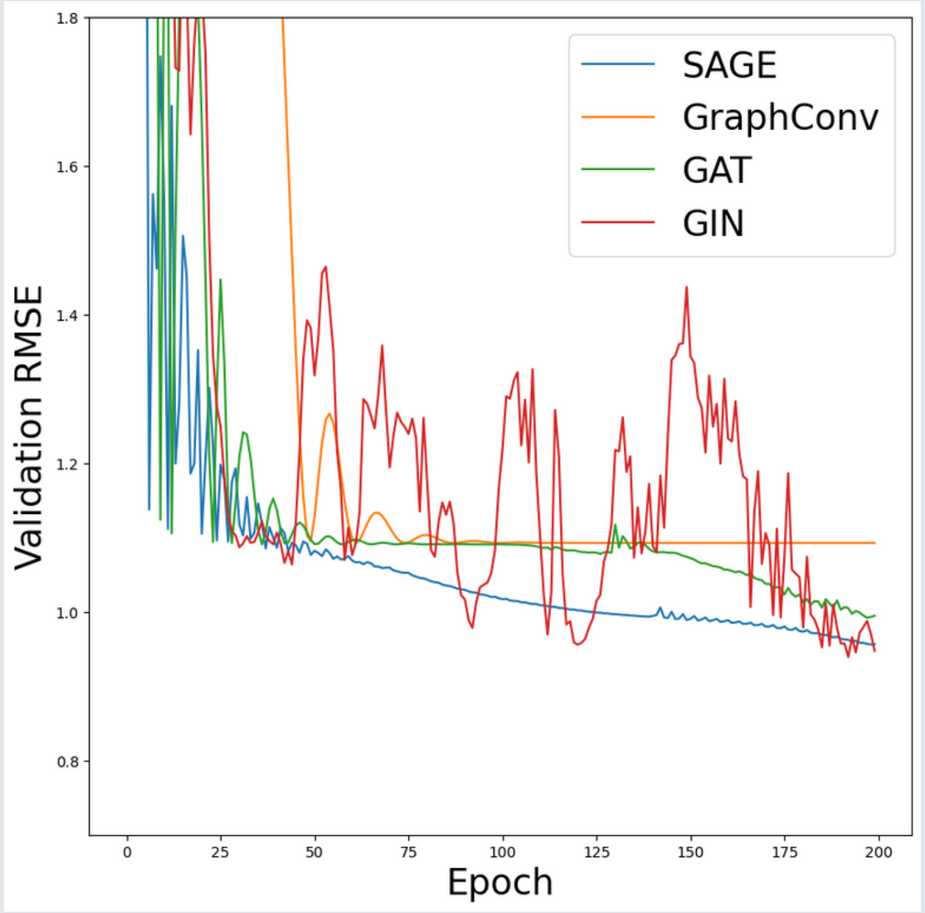
5. Experiments > Most Significant

GNN architecture

Training RMSE



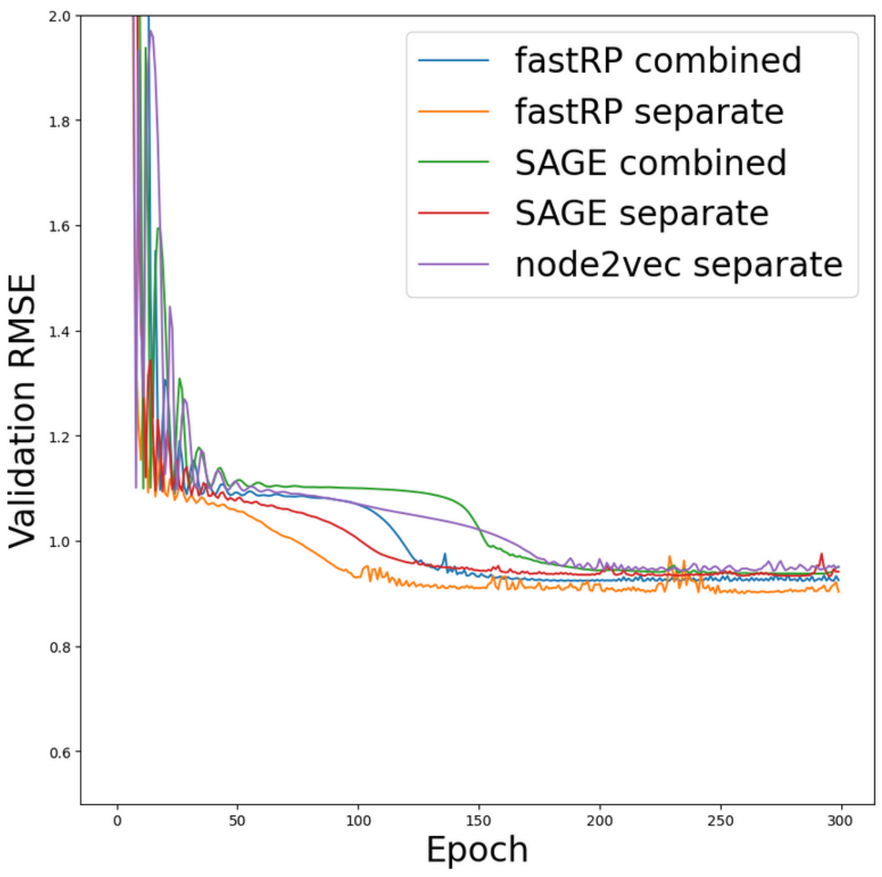
Validation RMSE



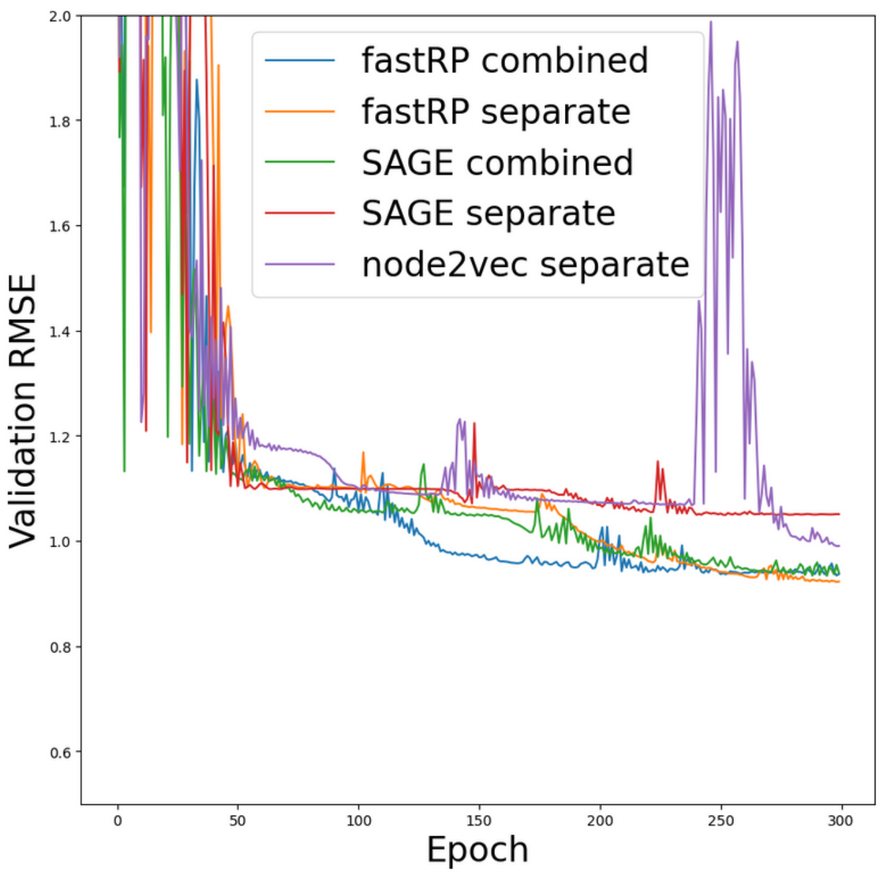
5. Experiments > Most Significant

Movie-content node embeddings (algorithm & subgraph)

Validation RMSE on GraphSAGE



Validation RMSE on GAT



FastRP algorithm performs better on all models

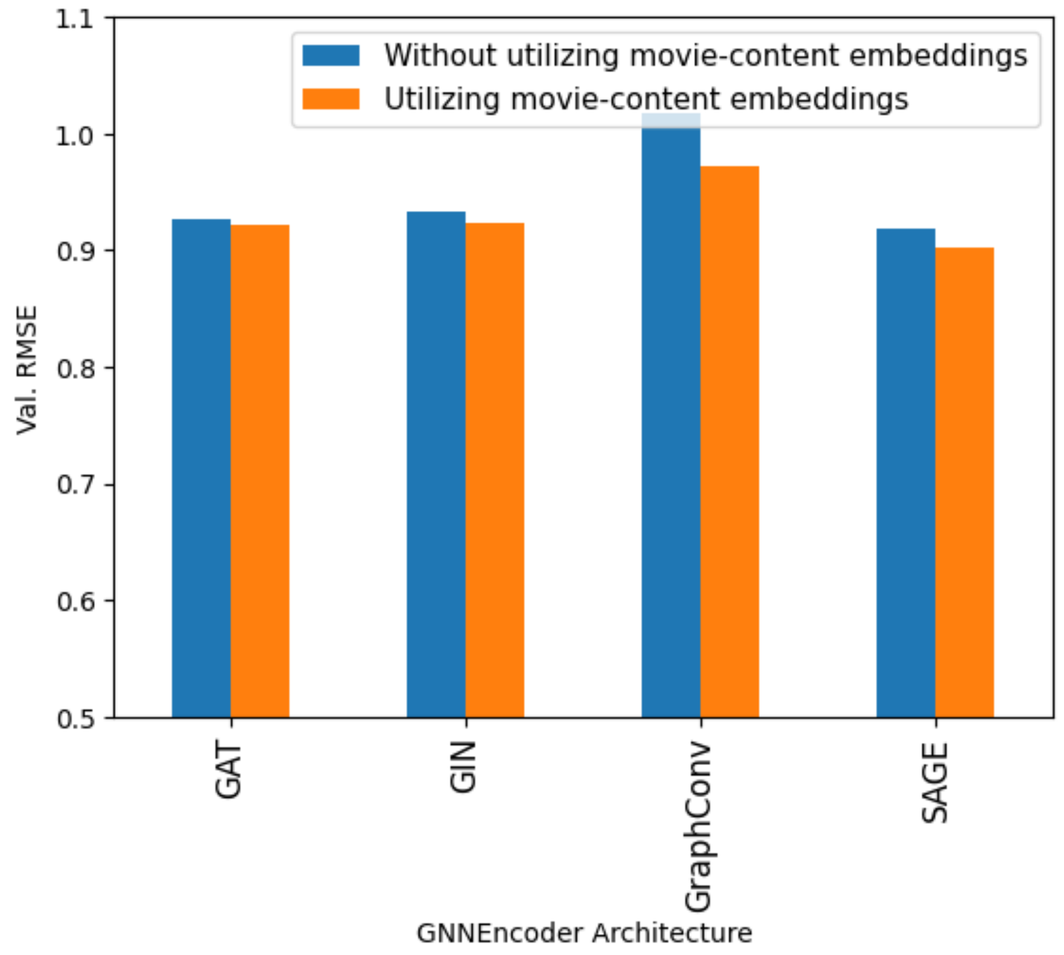
Utilizing all the **separate** bipartite graphs seems to perform better on **GraphSAGE** and **GAT**

One embedding encoding the **combined** movie-content seems to perform better on **GraphConv** and **GIN**

5. Experiments > Most Significant

Movies-content node embeddings utilization

Validation RMSE loss w/o movie-content node embeddings



Node embeddings encoding the movies-content **outperform** simple **collaborative-wise** approach in all models

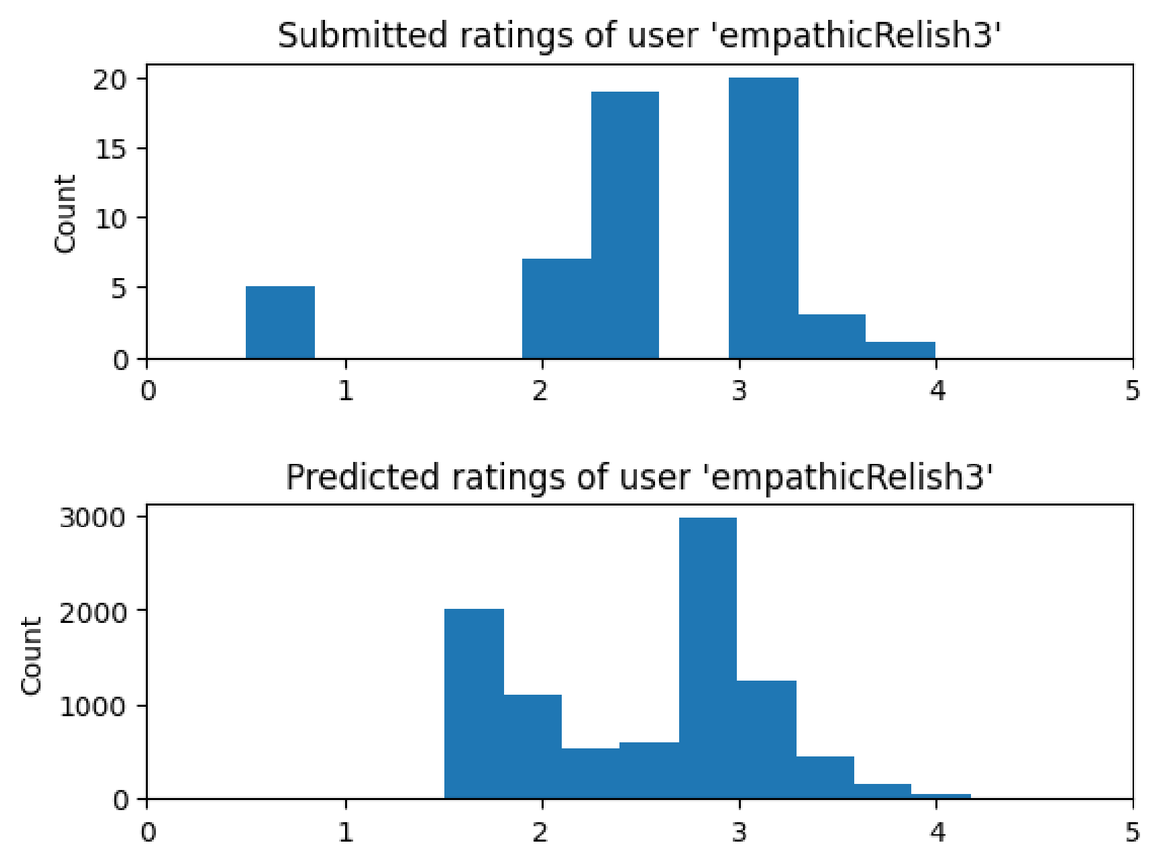
Our best model: **0.904 RMSE**

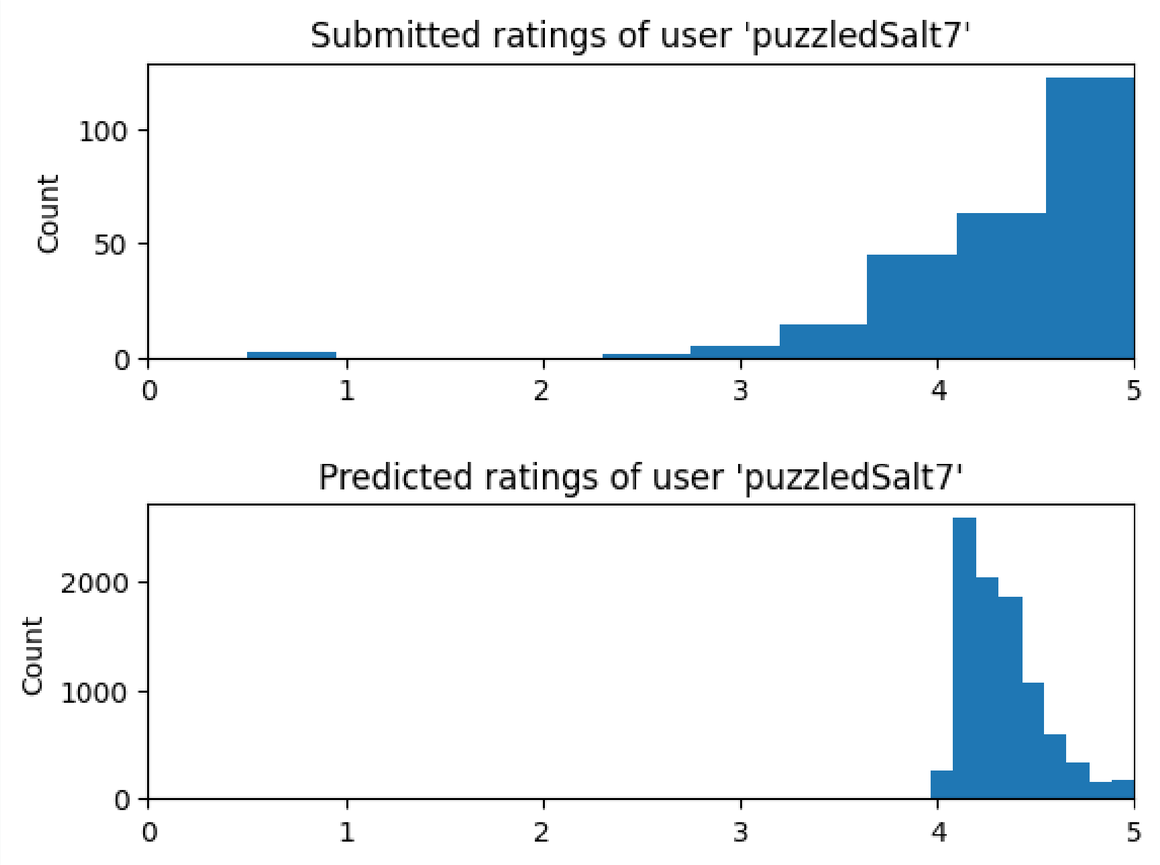
- 2 GraphSAGE GNN layers
- FastRP embeddings
- multiple separate node embeddings for each movie's content

5. Experiments > Practical use cases

Personalized predictions

Distribution of ratings varies for different users



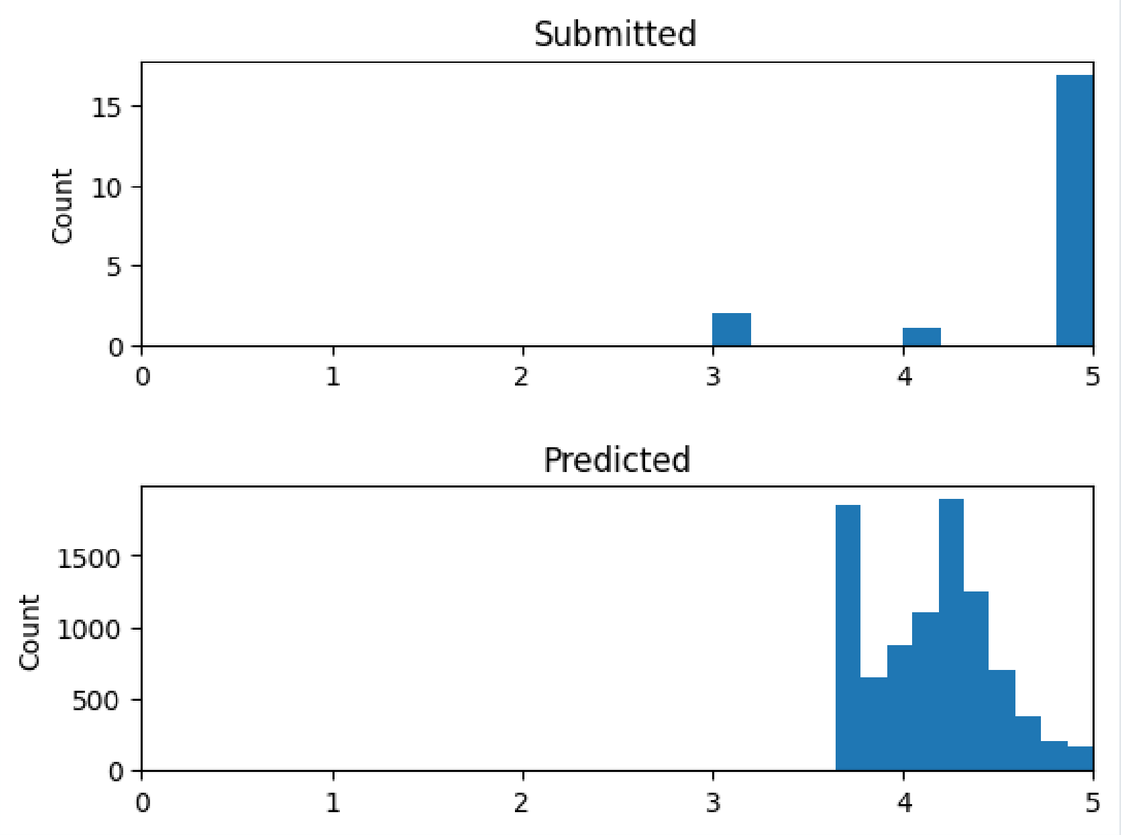


5. Experiments > Practical use cases

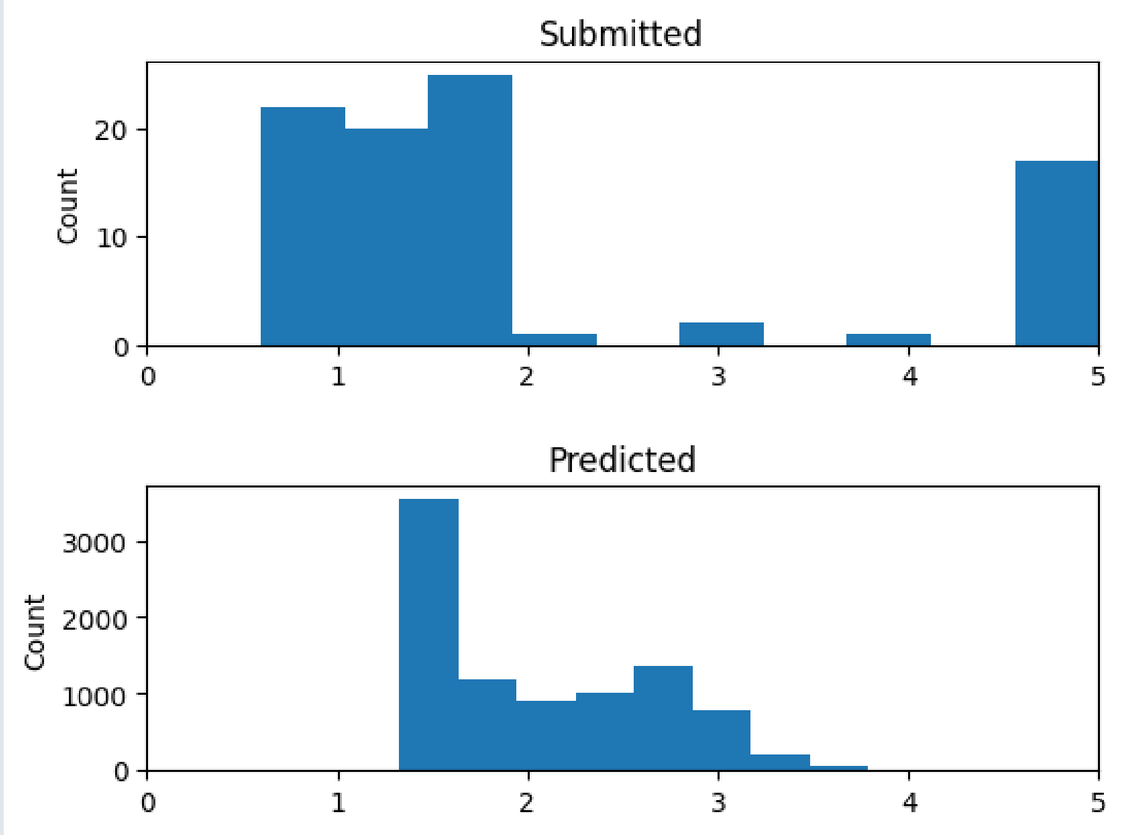
Detecting changes in users behavior over time

Distribution of ratings for a specific user over time

Initial submitted & predicted ratings distributions for a specific user



After submitting some new ratings for the user, and re-training the model, the predicted ratings are shifted significantly to follow the new distribution

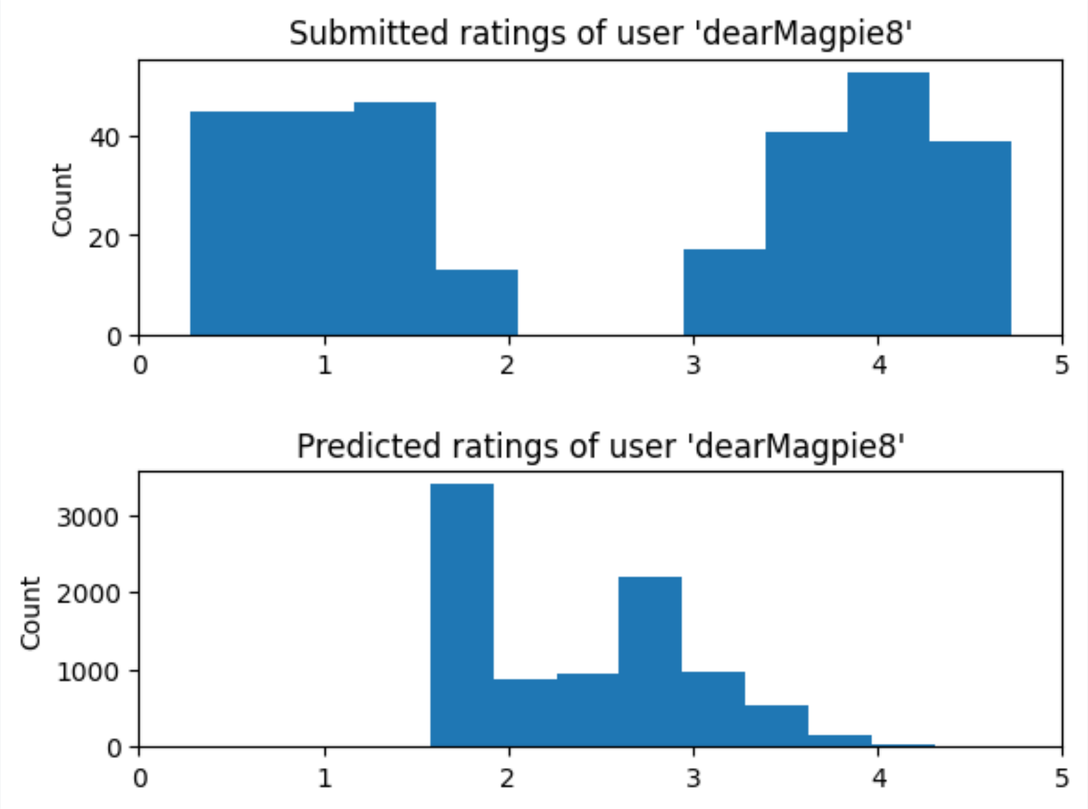


5. Experiments > Practical use cases

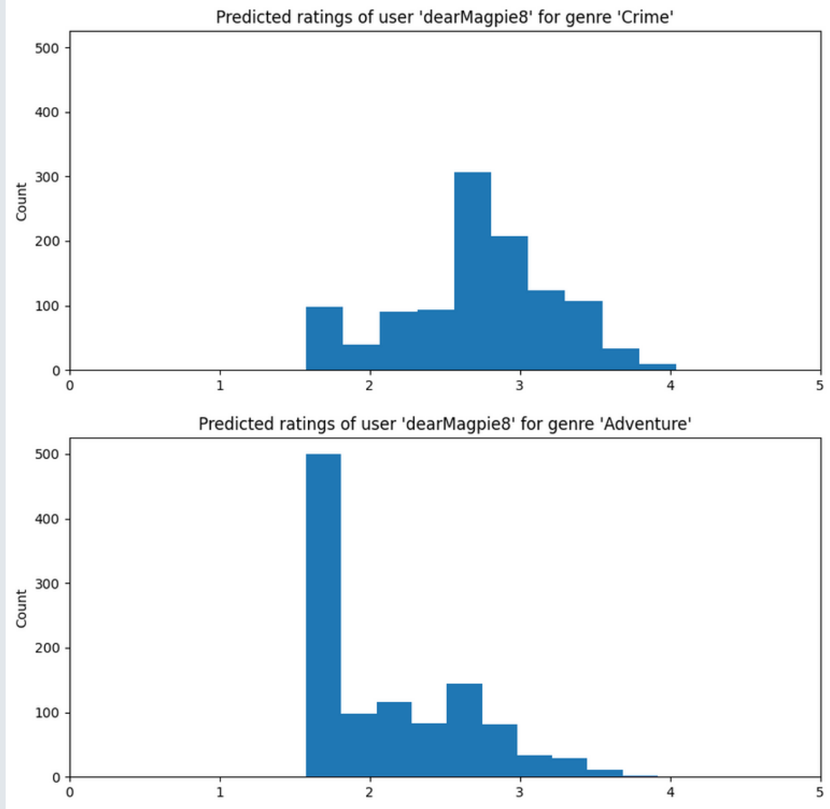
Detecting more complex user tastes

A user that seems to like the genre "Crime" (rates in the range [3.0, 4.5]) and dislike the genre "Adventure" (rates in the range [0.5, 2.5])

Predicting the distribution for the specific user seems to be challenging for the model



The model succesfully tends to predict higher ratings on Crime movies and lower on Adventure movies



6. Conclusion & Future work

6. Conclusion

In this diploma thesis, we:

- Built a whole **movie recommendations web application**
- Implement an ML model using multiple **GNN** architectures to perform the **link weight prediction** task
- Utilized effectively the **rich movies metadata**, with multiple **node embedding algorithms** on the movie-content subgraph
- **Integrated** the model into the web application

Source code

https://github.com/John-Atha/diploma

6. Conclusion > Future Work

Further research/development

- When to **re-train** the model?
- When to **refresh** the in-memory graph on the Model API?

- Experiments in handling **new users**
- Experiment with more **GNN architectures** and **node embedding algorithms**
- Study the reasons behind the **unstable** training of **GIN**

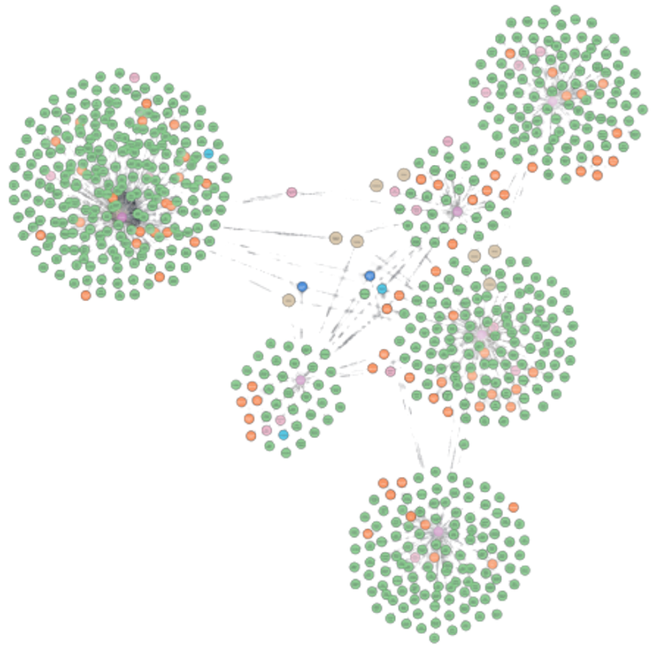
- Experiment with **explainability** on GNNs
- **Deploy** the platform to support the **expansion of the dataset** and **test our model** in a real-world setup

Source code

https://github.com/John-Atha/diploma

Thank you!

Questions ?



Source code

https://github.com/John-Atha/diploma