Nurzhas Abdikenov
BDA-2103

Advanced Programming

Final Project:
Movie Recommendation System

Introduction

Problem

It's really hard to find a great movie that would suit you. You waste a lot of time trying to find it. Nowadays, there tones of movies and you just can't decide which one to watch.

Literature review:

1. How to make a movie recommender: creating a recommender engine using Keras and TensorFlow. This tutorial uses ANN for a recommendation system. Trains the model using ANN and in the result outputs top k movies for particular user from the dataset.
2. Movie-Recommendation-System-Using-KNN-Algorithm. This solution uses KNN and I initially planned to use it. But I couldn't find elements of DL here. Simple ML.
3. MovieLens-Hybrid-Movie-Recommendation-System. This project has an NCF model. Uses "timestamps", rating-based and uses "user_id" to identify unique reviews. I took the idea of my project from here.
4. NLP-based Movie Recommendation System it's similar to [1] solution. Uses NLP to output top movies that's similar to the movie we entered.

Current Work(Most of the explanation are on the notebook)

1. I've found a dataset on MovieLens(25M Dataset).
2. Based on the solutions and tutorials above I decided to use NCF model.
3. I preprocessed the dataset
4. Split it using Leave-one-out methodology
5. Dropped the unnecessary columns for the project

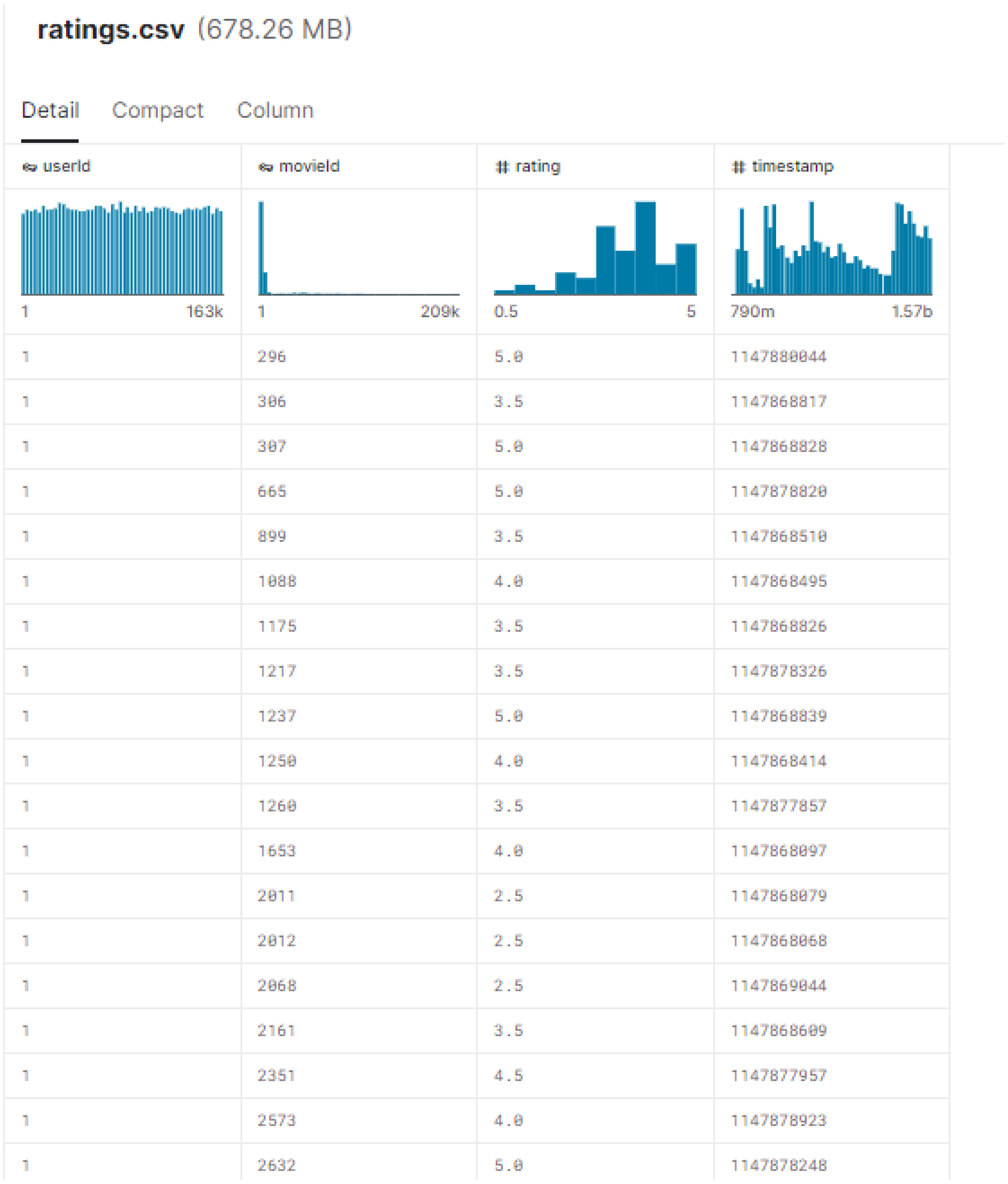
6. Converted explicit feedback(ratings column) to implicit feedback. Converted them to 1’s
7. With 4:1 ratio created negative review samples. That was done to movies that weren’t interacted with by the user.
8. I’ve created Pytorch Dataset class to facilitate training of my model.
9. In notebook, explained how NCF model works and provided illustrations, example + article
10. Created the NCF model itself
11. Instantiated it and trained for 5 epochs and 1 google collab gpu.
12. Explained how evaluation for recommendation system works on bullet list
13. Evaluated model with Hit Ratio 25. Result: 90%.

Data and Methods

Information about the data (probably analysis of the data with some visualisations)

MovieLens Dataset contains 25million rows with 4 columns.

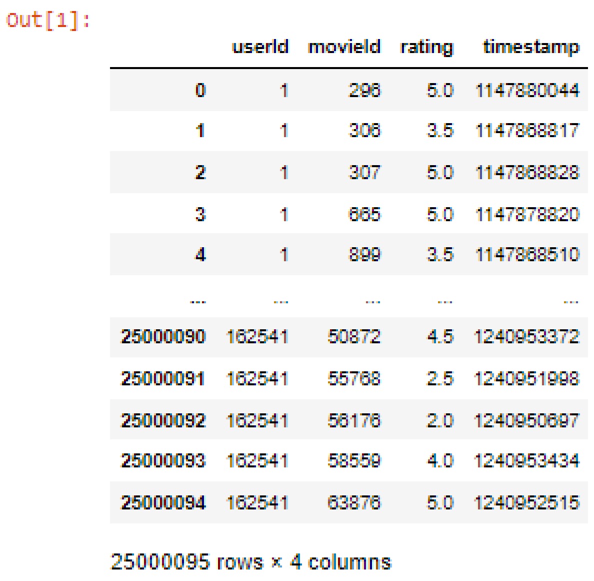
- userId, movieId, rating and timestamp



ratings.csv (678.26 MB)

userId	movieId	rating	timestamp
1	296	5.0	1147880044
1	306	3.5	1147868817
1	307	5.0	1147868828
1	665	5.0	1147878820
1	899	3.5	1147868510
1	1088	4.0	1147868495
1	1175	3.5	1147868826
1	1217	3.5	1147878326
1	1237	5.0	1147868839
1	1250	4.0	1147868414
1	1260	3.5	1147877857
1	1653	4.0	1147868097
1	2011	2.5	1147868079
1	2012	2.5	1147868068
1	2068	2.5	1147869044
1	2161	3.5	1147868609
1	2351	4.5	1147877957
1	2573	4.0	1147878923
1	2632	5.0	1147878248

Here is the dataset:

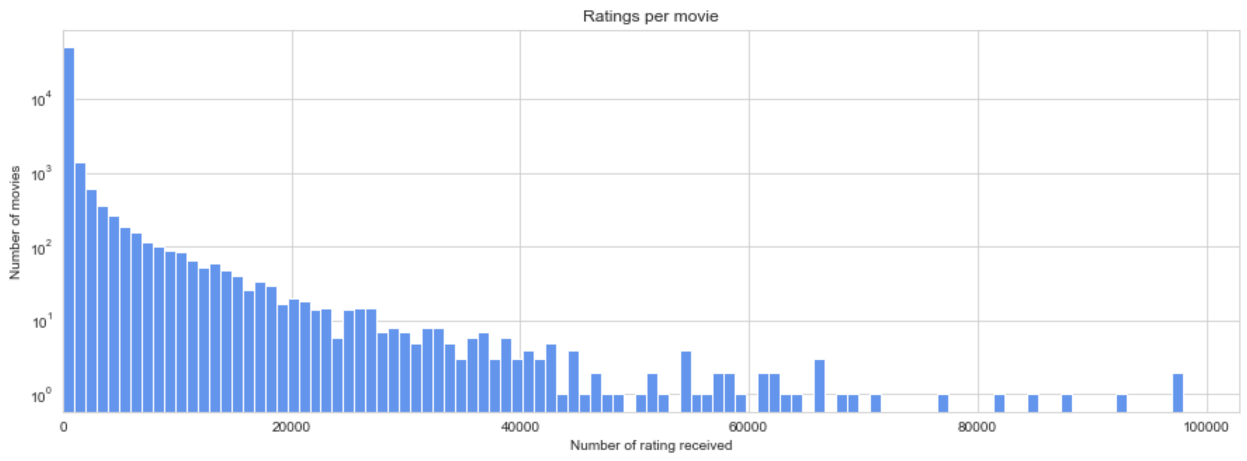


Out[1]:

	userId	movieId	rating	timestamp
0	1	296	5.0	1147880044
1	1	306	3.5	1147868817
2	1	307	5.0	1147868828
3	1	665	5.0	1147878820
4	1	899	3.5	1147868510
...	...	...	...	...
25000090	162541	50872	4.5	1240953372
25000091	162541	55768	2.5	1240951998
25000092	162541	56176	2.0	1240950697
25000093	162541	58559	4.0	1240953434
25000094	162541	63876	5.0	1240952515

25000095 rows × 4 columns

62,000 users and 25m ratings:



Source of the picture above

Description of the ML models you used with some theory

Model - Neural Collaborative Filtering (NCF)

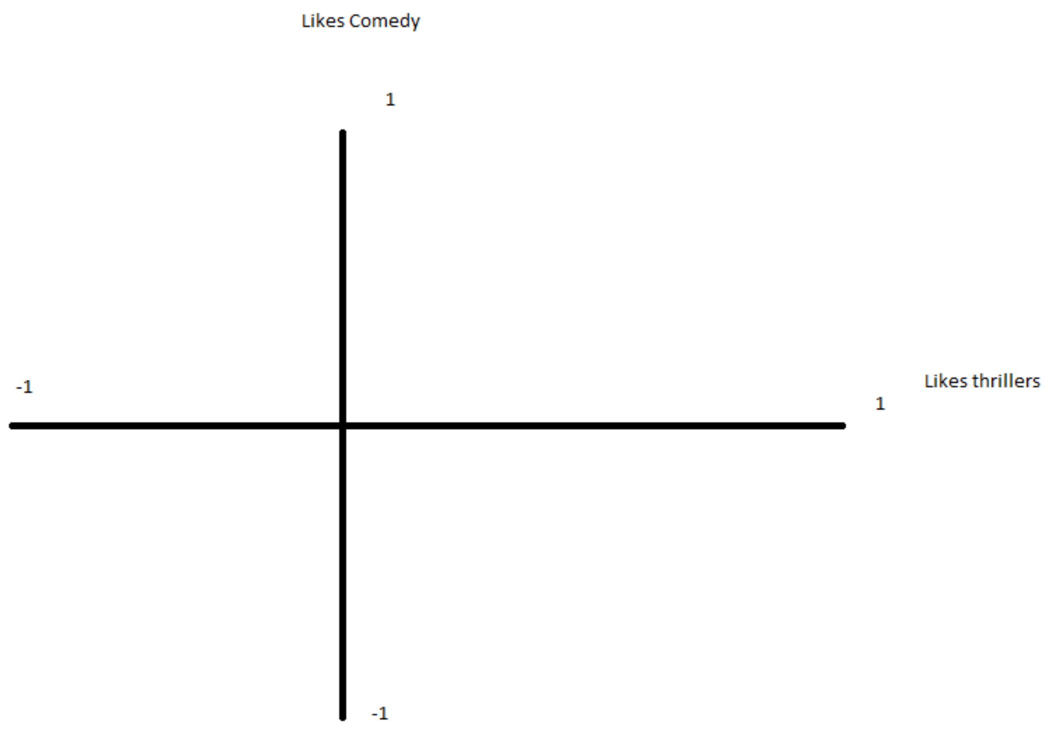
Article

User Embeddings

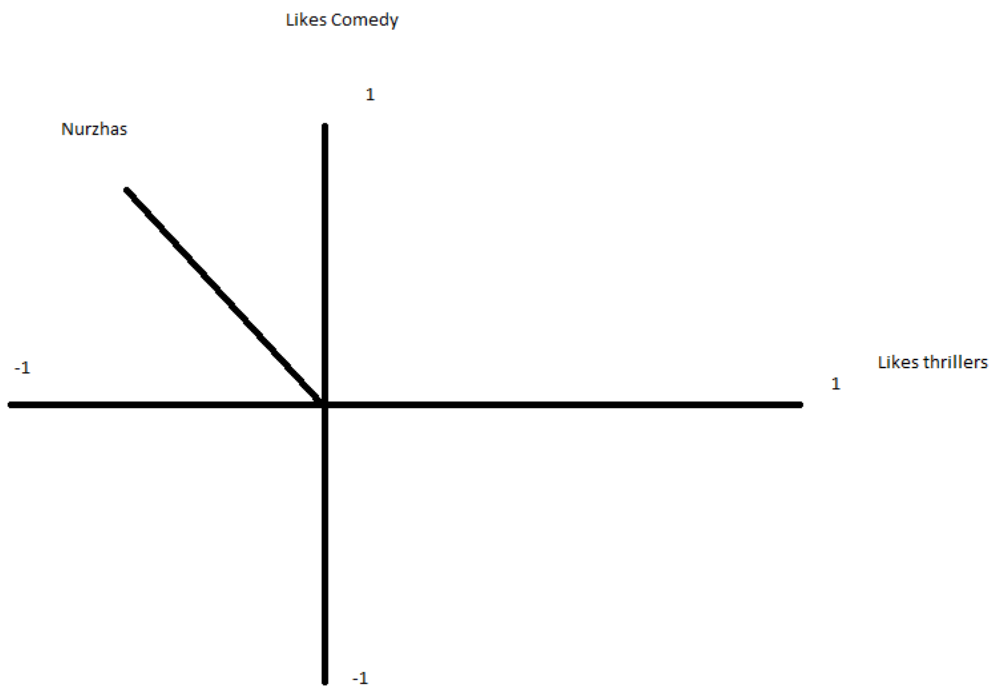
An embedding is a low-dimensional space that captures the relationship of vectors from a higher dimensional space.

Ex.: User loves 2 genres - Comedy and thrillers movies.

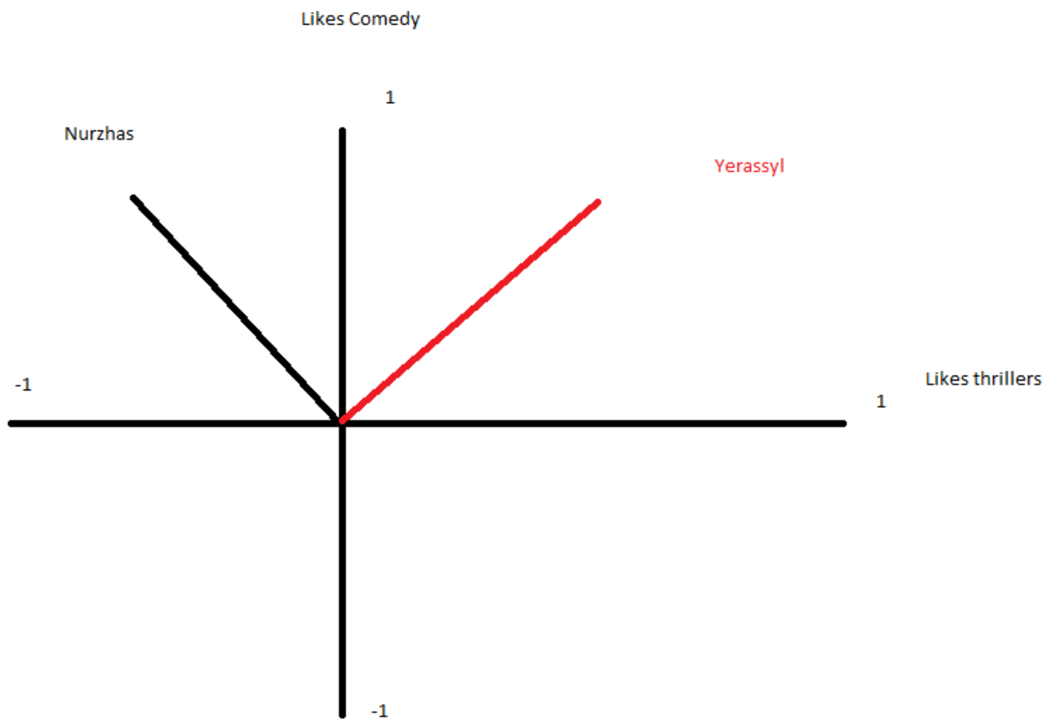
1st dimension is - how much he loves comedy movies. 2nd dimension is - how much he loves thriller movies.



Let's say, Nurzhas loves thrillers, but doesn't likes thrillers. So, in graph, he would look like this:



Other user, Yerasyl is opposite of Nurzhas.



This is known as 2d embedding. Here we can easily track preferences of users in 2d linespace.

More dimensions = more traits of users. In this model exactly, I used 8d model.

Learned Embeddings

Seperate movie embegging will be used to represent the traits of movies(in lowerd dimension).

How to automatically get weights for embeddings? Answers is - Collaborative Filteging.

By using the ratings dataset, we can identify similar users and movies, creating user and item embeddings learned from existing ratings.

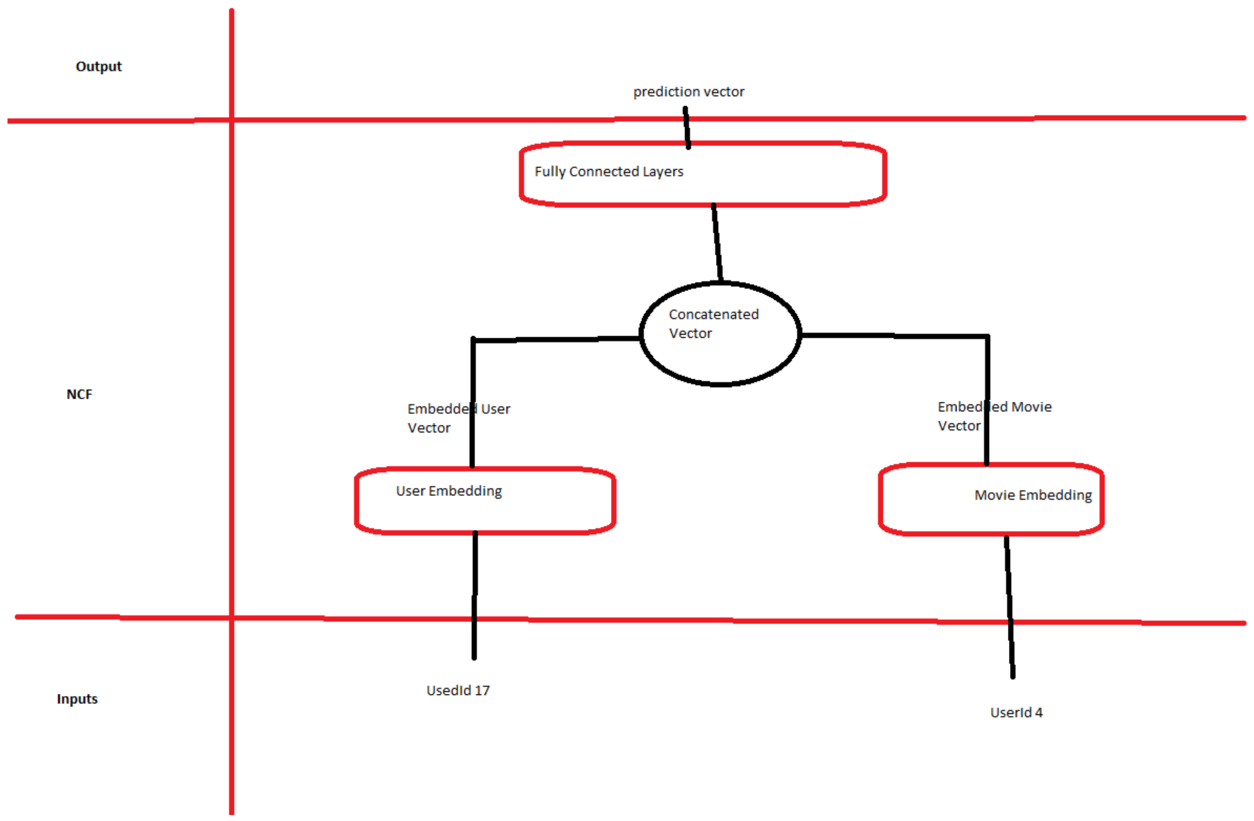
Model Architecture

User and movie(item) embeddings are key to the model.

Let's walk through the model architecture using the following training sample:

userId	**movie ID**	**interacted**
17	4	1

here that's positive review, because user interacted with movie.



The user input vector and item input vector are fed to the user embedding and item embedding respectively, which results in a smaller, denser user and item vectors.

The embedded user and item vectors are concatenated before passing through a series of fully connected layers, which maps the concatenated embeddings into a prediction vector as output.

Finally, we apply a Sigmoid function to obtain the most probable class.

Results

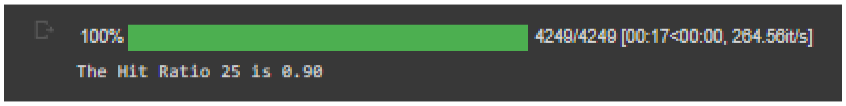
Evaluating our Recommender System

I simulated it's evaluation for 25 items that could be recommended for user.

- User have 99 items that he didn't click on.
- We combine it with 1 test item(remember Avatar movie?)

- Run the model, and rank them according to their probabilities.
- I chose top 25 movies from this 100 initial items. If the test movie is there, then out model works.
- Repeat it for all users. The Hit Ratio is then the average hits.

This is Hit Ratio 25, and it's used in most of the modern recommendation systems.



```
100%|██████████| 4249/4249 [00:17<00:00, 264.56it/s]
The Hit Ratio 25 is 0.90
```

To put this into context, what this means is that 90% of the users were recommended the actual item (among a list of 25 items) that they eventually interacted with.

One of those items will be next movie that you will watch.

Discussion

Critical review of results

I tried evaluating it with accuracy or RMSE, but it didn't fit the recommendation system evaluation.

Why? Because, user don't have to click on every item that he sees. Instead, we need him to click on one of the many items. As long as users obey this pattern, our system works great.

Model showed great results on evaluation,

Next steps

I’ve used one csv of a huge dataset. From now on, I can continue it by actually showing movie titles for every user. I should look more on different solutions, especially MovieLens-Hybrid-Movie-Recommendation-System, so I could further develop the idea of this project.