Project: Movie recommendation

Ali and Isabella

January 8, 2022

Abstract

What is better than knowing what to watch next on netflix? Instead of continuous random scrolling through the movies, we develop a recommendation system for netflix movies. The model we develop fuses both collaborative filtering and content-based recommendation. The work is based on the Neural Collaborative Filtering framework designed in [HLZ+17] and the Sentence-BERT transformers [RG19]. On a high level, our method is shown below, more details about the steps will be discussed in section 3:

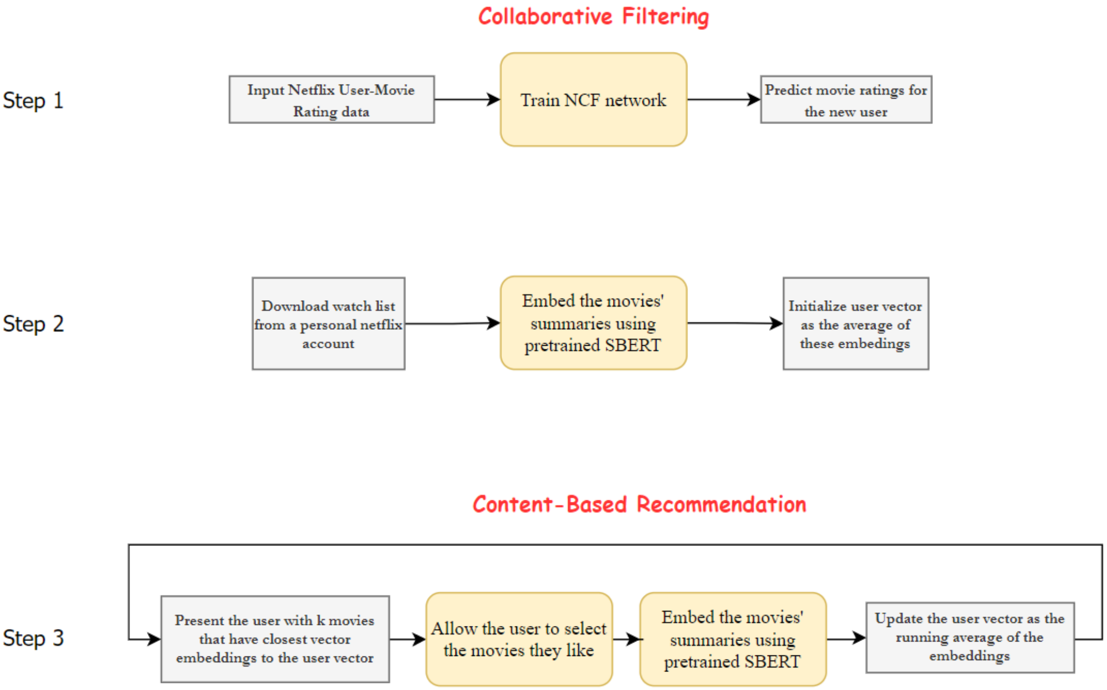

Collaborative Filtering
Step 1
Input Netflix User-Movie Rating data
Train NCF network
Predict movie ratings for the new user
Step 2
Download watch list from a personal netflix account
Embed the movies' summaries using pretrained SBERT
Initialize user vector as the average of these embedings
Content-Based Recommendation
Step 3
Present the user with k movies that have closest vector embeddings to the user vector
Allow the user to select the movies they like
Embed the movies' summaries using pretrained SBERT
Update the user vector as the running average of the embeddings

Figure 1: High level algorithm

The dataset and the weights files can be found at `https://drive.google.com/drive/folders/1-HWiXi7tfHvH3xVOoLOX6IK3fciPGi3w?usp=sharing`.
The code can be found at `https://github.com/alishibli97/Scalable-Machine-Learning-and-Deep-Learning/tree/main/project`.

1 Introduction

The crucial component to a custom-made recommender system is in modelling users' preference on elements based on their history of interactions, also known as collaborative filtering. There are many collaborative filtering techniques but matrix factorization is the most common one. Matrix factorization projects users and items into a shared latent space, using a vector of latent features to represent a user or an item. After that a user's interaction on an item is modelled as the inner product of their latent vectors.

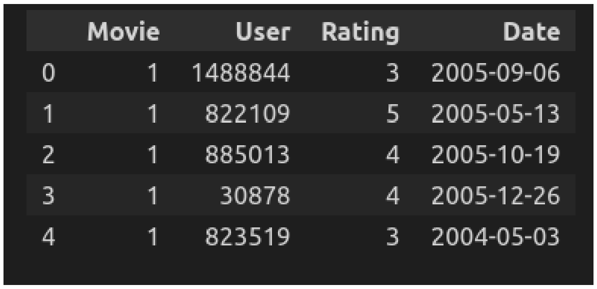
A great deal of investigation work has been dedicated to improve matrix factorization, for example integrating it with neighbor-based models, combining it with topic models of item content, and extending it to factorization machines for a generic modelling of features. Regardless of the strength of matrix factorization for collaborative filtering, it is known that its results can be disrupted by the choice of the interaction function — inner product.

For instance, for the assignment of rating prediction on explicit feedback, it is acknowledged that the performance of the matrix factorization model can be upgraded by incorporating user and item bias terms into the interaction function. Although it can seem to be a minor tweak for the inner product operator, it points to the positive effect of composing a superior interaction function for modelling the latent feature interactions between users and items. The inner product, which connects the multiplication of latent features linearly, may not be enough to capture the complicated structure of user interaction data. Our project is inspired by Neural Collaborative Filtering [HLZ^{+}17] which explores the use of deep neural networks for learning the interaction function from data.

We also include content-based recommendation using SBERT on the summary of each movie (content). Sentence-BERT (SBERT) is a modification of the BERT network using siamese and triplet networks that is able to derive semantically meaningful sentence embeddings.

2 The Dataset

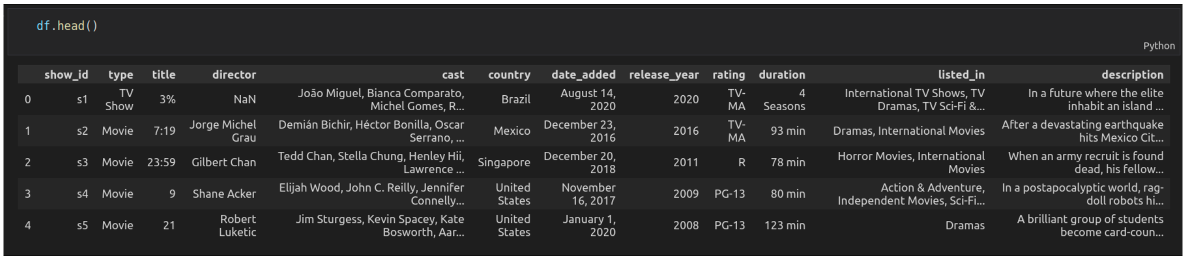
Our dataset is the netflix movie-user ratings dataset [net]. In total the dataset has 100 million unique movie-user pairs. Due to feasibility and computational issues, we take only 25 million pairs of movie-user ratings. With them we train the NCF for 2 hours on 6x A6000 RTX GPUs on the cloud. For this the dataset looks like:



	Movie	User	Rating	Date
0	1	1488844	3	2005-09-06
1	1	822109	5	2005-05-13
2	1	885013	4	2005-10-19
3	1	30878	4	2005-12-26
4	1	823519	3	2004-05-03

Figure 2: Movie-User ratings dataset

In addition, each movie has a full description, we use this for the content-based recommendation. This dataset looks like:



```
df.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	TV Show	3%	NaN	João Miguel, Bianca Comparato, Michel Gomes, R...	Brazil	August 14, 2020	2020	TV-MA	4 Seasons	International TV Shows, TV Dramas, TV Sci-Fi &...	In a future where the elite inhabit an island ...
1	s2	Movie	7:19	Jorge Michel Grau	Demián Bichir, Héctor Bonilla, Oscar Serrano, ...	Mexico	December 23, 2016	2016	TV-MA	93 min	Dramas, International Movies	After a devastating earthquake hits Mexico Cit...
2	s3	Movie	23:59	Gilbert Chan	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	Singapore	December 20, 2018	2011	R	78 min	Horror Movies, International Movies	When an army recruit is found dead, his fellow...
3	s4	Movie	9	Shane Acker	Elijah Wood, John C. Reilly, Jennifer Connelly...	United States	November 16, 2017	2009	PG-13	80 min	Action & Adventure, Independent Movies, Sci-Fi...	In a postapocalyptic world, rag-doll robots hi...
4	s5	Movie	21	Robert Luketic	Jim Sturgess, Kevin Spacey, Kate Bosworth, Aar...	United States	January 1, 2020	2008	PG-13	123 min	Dramas	A brilliant group of students become card-coun...

Figure 3: Netflix movie dataset

1. For the Collaborative filtering (NCF neural network) training, we represent the movie-user as one-hot encoded data.
2. For the Content-based Recommendation (SBERT), we embed the the descriptions of the movies.

3 Method

In this section, we describe the 3 steps in our method:

3.1 Step 1

We build the Neural Collaborative Filtering (NCF) network [HLZ+17]. The architecture looks as follows:

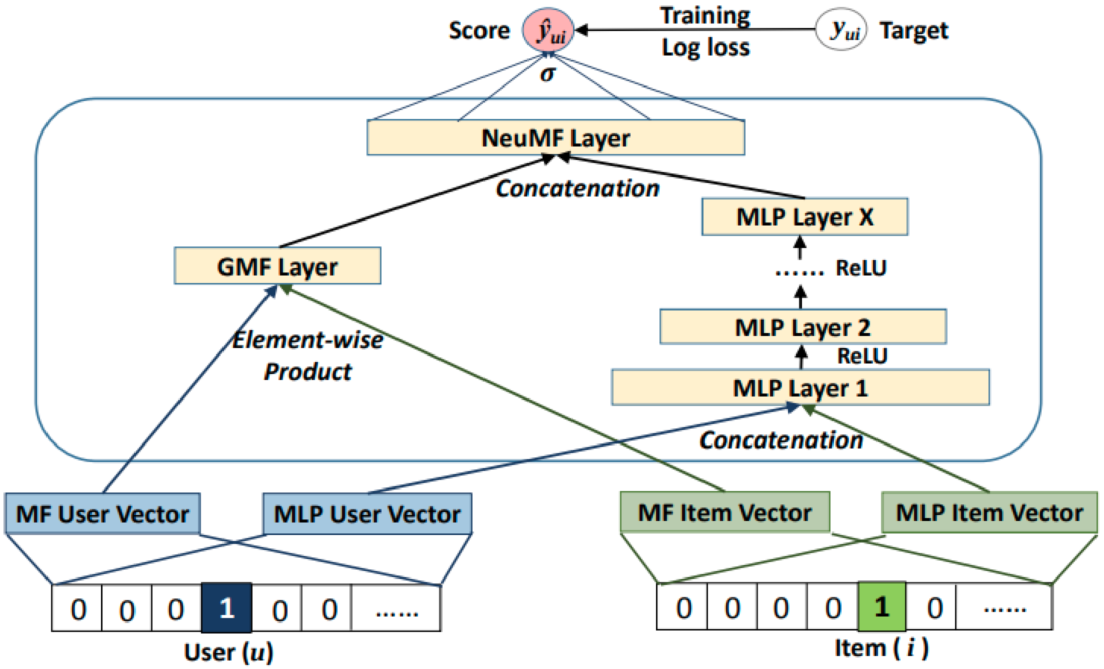


Figure 4: Neural matrix factorization model

This model combines 2 approaches to collaborative filtering: matrix factorization (or generalized matrix factorization GMF) and multi-layer perceptron (MLP). GMF and MLP are two approaches to collaborative filtering. GMF uses dot product between one hot encoded data while the MLP uses a feedforward neural networks. The neural matrix factorization model combines both approaches by allowing each model to obtain its own embeddings, and then concatenating the embeddings at the NeuMF output layer. Thus, we have 2 copies of the User (u) vector and 2 copies of the Item (i) vector. Each copy goes to one model, as can be seen in the diagram above.

We train this network on the 25 million rows of the Netflix Dataset that contain a movie-user ratings, for 2 hours on the cloud using 6x A6000 RTX GPUs. The training loss, where loss was defined as the Mean-Squared-Error between the predicted score and the actual score, can be shown below:

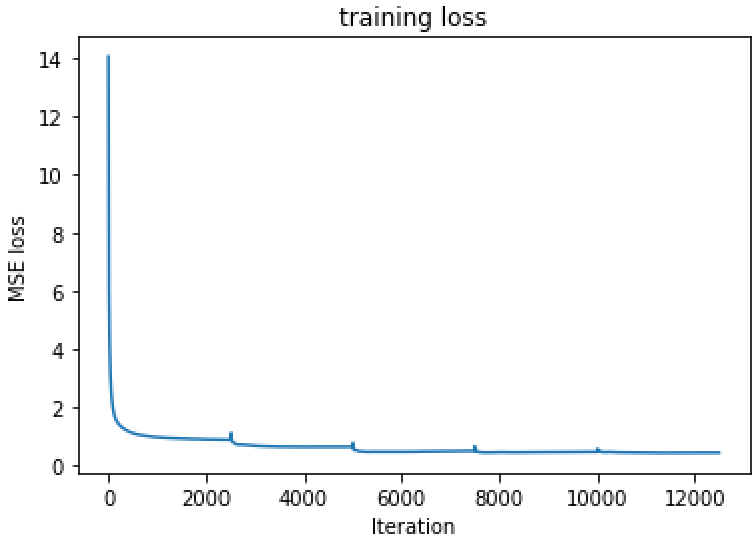


Figure 5: Training loss

The output of the model is a score between -1 and 1, indicating how likely a new user will like a particular movie, 1 being highly likely and -1 highly unlikely. Given this recommendation model, we take a new user and predict the scores of all the movies for this new user. We sort this list by score, and save it to disk.

3.2 Step 2

So far we have done collaborative filtering. To further adjust our model to update to the personal user preferences, we define a *user_vector* that indicates which movies he liked by movie description. We download the watch list of one of our personal netflix accounts (*Bella's account*). We collect the summaries of the movies from that watch-list, and embed these summaries using pre-trained SBERT [RG19]. The user vector is then initialized as the average of these vectors.

3.3 Step 3

In the final step that accounts for content-based filtering, we present the user with a graphical user interface GUI. In this GUI, we allow the user to select the movies they like from a list of given movies with their descriptions (so if they don't know the movies can check their description). Once they select and submit the liked movies, the *user_vector* updates as the running average of the vectors of the embedded descriptions using SBERT:

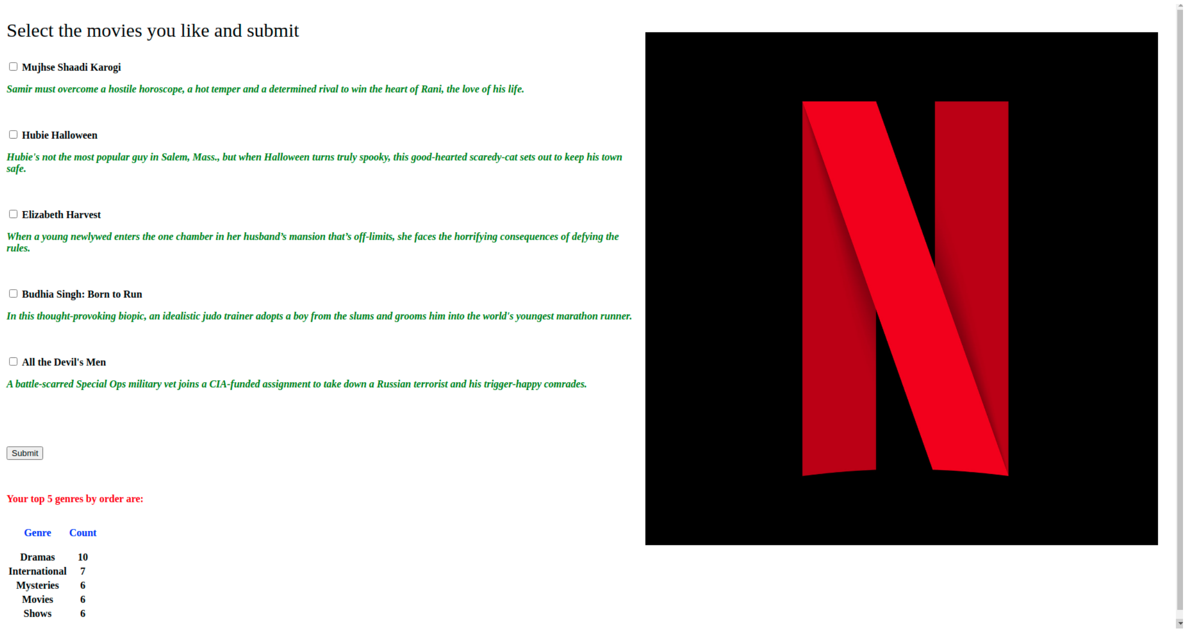
$$user\_vector = \frac{\Sigma_t v_t}{N_t} \tag{1}$$

where v_t is the movie summary embedding vector t, N_t is the total number of liked movies, and t runs over all the liked movies of the user.

At each iteration, the new movies recommended are the ones that have closest description embedding vector to the *user_vector*. We use the distance defined in [MM99], which performs approximate kNN to get closest points.

4 Results

The GUI looks as follows:



Select the movies you like and submit

□ **Mujhse Shaadi Karogi**

Samir must overcome a hostile horoscope, a hot temper and a determined rival to win the heart of Rani, the love of his life.

□ **Hubie Halloween**

Hubie's not the most popular guy in Salem, Mass., but when Halloween turns truly spooky, this good-hearted scaredy-cat sets out to keep his town safe.

□ **Elizabeth Harvest**

When a young newlywed enters the one chamber in her husband's mansion that's off-limits, she faces the horrifying consequences of defying the rules.

□ **Budhia Singh: Born to Run**

In this thought-provoking biopic, an idealistic judo trainer adopts a boy from the slums and grooms him into the world's youngest marathon runner.

□ **All the Devil's Men**

A battle-scarred Special Ops military vet joins a CIA-funded assignment to take down a Russian terrorist and his trigger-happy comrades.

Your top 5 genres by order are:

Genre	Count
Dramas	10
International	7
Mysteries	6
Movies	6
Shows	6

Figure 6: Graphical User Interface

We present the user with 5 movies with their summaries at each iteration. At the bottom page, we present the genres that the user have liked mostly over time.

5 How to run the code

We implement our code in python. To run the code, you need the following libraries installed:

1. flask
2. tensorflow
3. sentence_transformers
4. scipy
5. pandas
6. numpy

In the directory of the code, you just run *python flask run*.

References

[HLZ+17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering, 2017.

[MM99] Songrit Maneewongvatana and David M. Mount. Analysis of approximate nearest neighbor searching with clustered point sets, 1999.

[net] https://www.kaggle.com/shivamb/netflix-shows/.

[RG19] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.