Recommendation Systems Final Project Report

Maytar Shalit and Oren Avidan (Group #4)
Reichman University
February 2023
GitHub link

Original Paper Summary

Description of the paper

The anchor paper is - Deep Neural Networks for YouTube Recommendations from 2016.

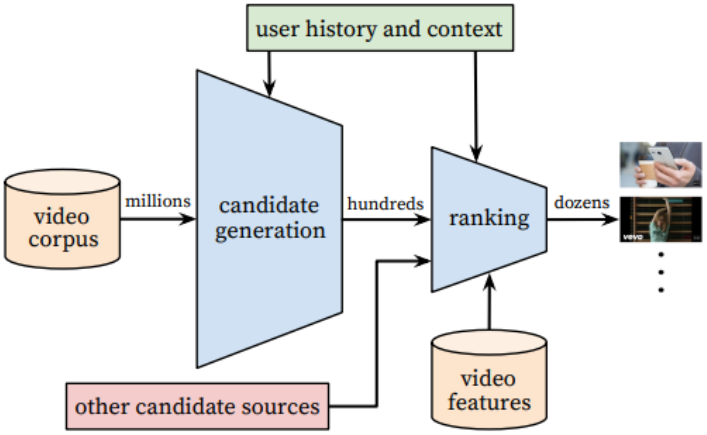
The main objective of the paper is to present the model and architecture that stand behind YouTube's recommendation system. The paper presents the structure of the model, the conclusions and reasons for choosing certain parameters, and the improvements vs previous methods that were used in the past. The article also describes the lessons and insights the writers gained.

The paper describes the structure of the Neural Networks (NN) that enable YouTube to recommend a few videos out of a gigantic corpus. The first NN is responsible for the Candidate Generation and the second one is for the Ranking. The algorithm described in this article explains how it manages to recommend new and "fresh" videos and overcome problems of sparse data and other factors that are not included in the data gathered (such as videos the user has watched because they were shared by his friend, what was the exact satisfaction of a user from a video, or missing meta-data such as tagged genres).

The model in the article relies on TensorFlow, of which we also relied in our work (using Keras).

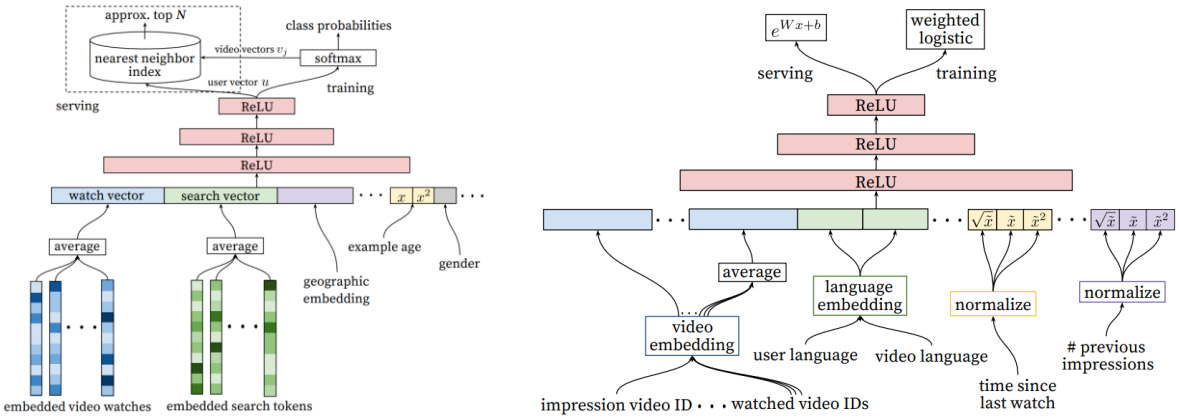
Building the model:

General model architecture is as follows:



As suggested in class, and due to the fact that the dataset we were using is pretty small (100K ratings over 1682 movies, instead of millions of movies in YouTube), there was no need for those 2 separate networks. In the project, we input to the first layer of the model all of the data used in the model described in the paper – including the normalized age, the normalized age squared and rooted, the "age" of the movie normalized, squared, and rooted, the occupation, gender and age of the user, etc.

The following pictures show the architecture of each of the models – to the left is the Candidate Generation model, and to the right is the Ranking network.



We can see that the networks are pretty similar, but the Ranking network receives also additional video features, and engineered features, and the training is done using a different last layer.

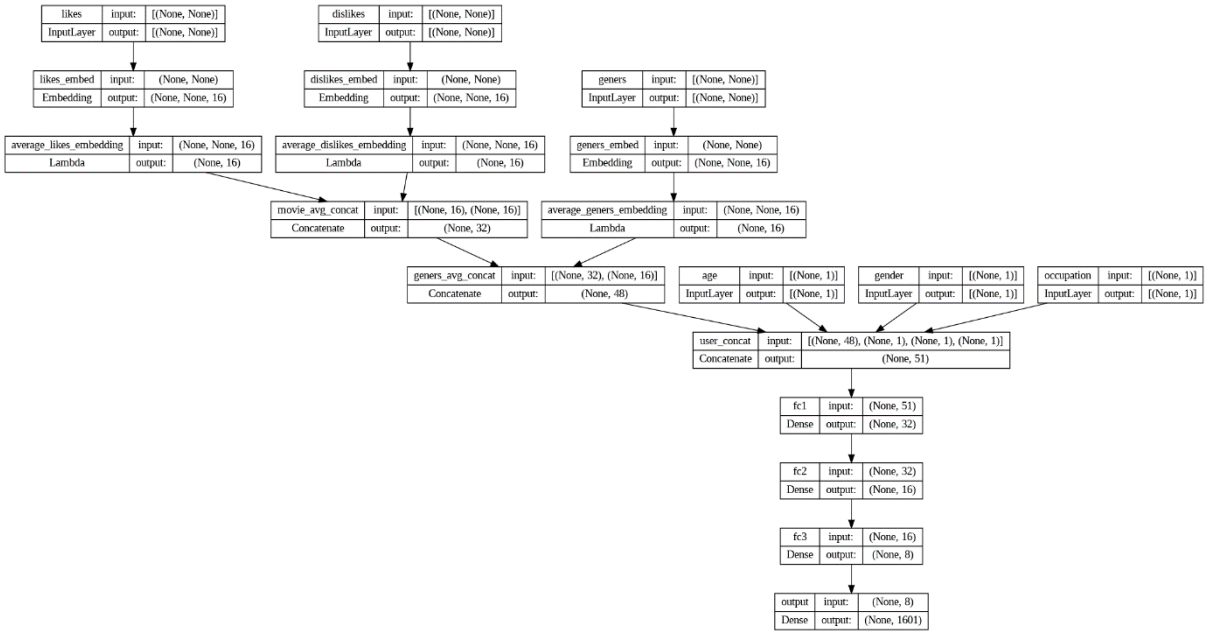
Thus, the parameters we used in the implemented model were –

1. IDs of videos watched by the user
2. IDs of videos disliked by the user
3. (Search queries) - we wanted to use the names of the videos but understood this will be identical to (1) – more on that in the conclusions section)
4. User demographics – in our dataset we used gender, age, and occupation
5. Movies age
6. Movies genres – we used the most liked genres for each user

Network structure -

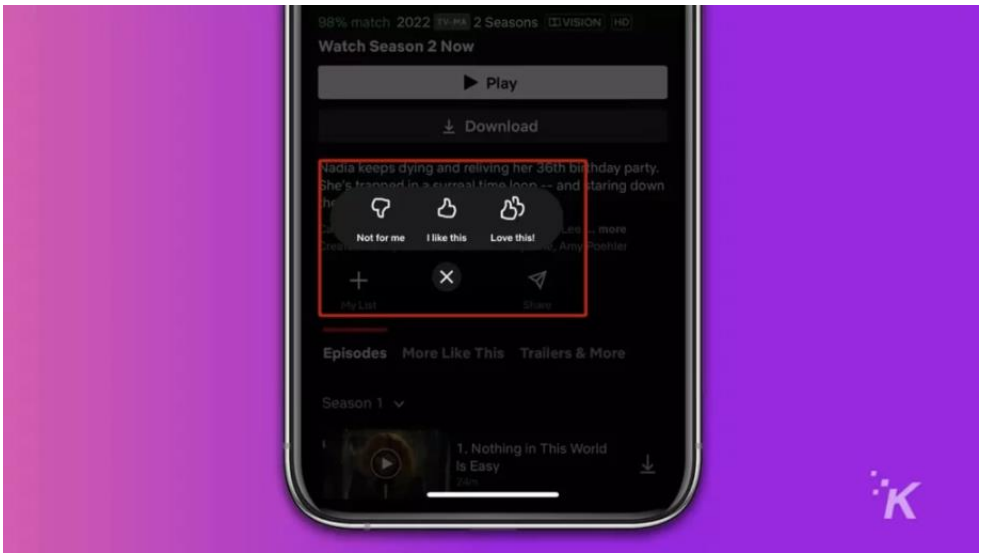
The structure of the network will be as follows -

1. The network embeds the videos IDs of liked and disliked movies, and averages them.
2. The genres (as numbers) are also being embedded.
3. The gender, occupation, and age are feeded as that into the model (gender and occupation are as categories)
7. All of the previous are then concatenated into one fully connected layer.
8. The concatenated layer is then passed through 3 FC layers with ReLU activation function.
9. The model treats the problem as a classification problem, so in the end, we used Softmax function to provide each of the possible movies the probability of being the next movie a user will watch. For example, for the train test that included 189 users, we received a 2D array at a size of 189*1665 (1665 is the number of movies that were rated by any user with a score of 3 and above)
10. All of the model, including the embeddings, is learned with all other model parameters through gradient-decent.
11. Due to the use of Softmax, we need to use Cross-Entropy for measuring the loss.



In serving time we will choose the N most likely videos.

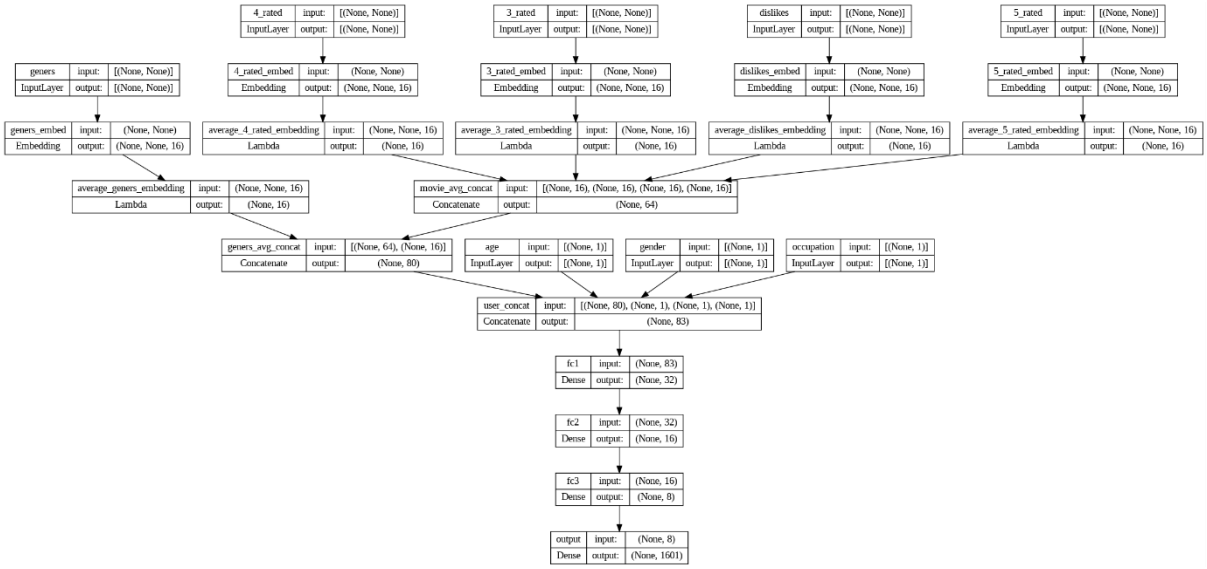
Innovations and Improvements



The improvement we thought of was combining the model with explicit ratings instead of implicit ratings. Our thought was that under the current model, the weights given to the “liked” movies are all the same, while some of the movies are “liked more”. Our research can show that it is worth to the YouTube programmers to add some more explicit rating methods to improve their recommendation algorithm.

What we did was to “divide” the like movies into 3 different categories – rated 3, 4, and 5. By thus we hoped to give higher weights to the movies in the 5-star, rather than the movies in the 3-star.

The structure of the model is as follows -



Baseline Algorithm

Due to the fact that the predecessor of the mentioned model was based on matrix factorization, we also decided to use this as our baseline model.

Chosen Dataset and Adjustments Made

Dataset Selection

The dataset we chose was the MovieLens at the size of 100K ratings, for a few reasons:

1. **Fits the article** - The general idea of the data fits the idea of the model described in the paper. They both handle movies, users, and ratings, and despite the fact we had missing data used in the YouTube model, we managed to overcome the problem with a few manipulations and data processing we made to the data.
2. **Computing power** - We would have been happy to choose the ML-1M dataset, but due to computing power restrictions we had to choose a smaller dataset.
3. **Easy to use and intuitive** - The dataset is easy to use and very straight forward in its structure. We were familiar with the dataset from previous homework, and it made it easier to work with instead of spending time on technical issues with other datasets.

Adaptations Made

In order to overcome the discrepancies and differences between the data described in the article and the data we used, we made so pre-processing, as follows –

1. **Uploading date** – we used the “release date” of the movie as the uploading date of the movie – since we know that users prefer newer movies on older ones.
2. **Latest videos watches/searched** – we translated the timestamp in the ratings dataset to discover the “watching time” of the user – to find the latest movies watched by the user. In the paper they mention that users like videos similar to the last videos they’ve watched, so that helps us in finding those videos.

3. **Personal data** – instead of using YouTube information on users, we used the age, gender, and occupation, as information that will go into the model, assuming that people with similar interest and ages will have common/similar preferences.
4. **Normalization** - for all verbal data such as occupation, gender, and movie genre with used categorization methods to change them to numbers. All continuous data such as age or time passed from release, we normalized and used the normalized value squared and rooted, as suggested in the article.

The final data used in the model looked as the following –

	user_id	likes	dislikes	liked_movie_names	age	gender	occupation	geners	days_passed_from_release_normalized	next_movie_to_watch	age_normalized
0	1	[256, 242, 111, 5, 171, 270, 189, 32, 209, 6, ...	[155, 266, 74, 27, 260, 145, 140, 225, 120, 10...	[Kolya, Copycat, Delicatessen, When the Cats A...	24.0	1	0	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.017466314963998003, 0.0227062094537405, 0...	257	0.257576
1	2	[304, 289, 300, 310, 281, 316, 272, 308, 269, ...	[314, 315, 294, 309, 10]	[FairyTale: A True Story, Good Will Hunting, R...	53.0	0	1	[0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14...	[0.027447066371996863, 0.0237755756548299 6, 0...	296	0.696970
2	3	[343, 331, 340, 347, 344, 271, 322, 181, 329, ...	[335, 245, 337, 323, 294, 332, 341, 300, 324, ...	[Cop Land, Murder at 1600, Prophecy II, The, M...	23.0	1	2	[0, 1, 4, 5, 6, 7, 9, 10, 12, 13, 14, 15, 16]	[0.023526056890283024, 0.0139730519711984 03, 0...	346	0.242424
3	4	[264, 328, 360, 362, 359, 324, 327, 11, 329, 2...	[358]	[Cop Land, Ulee's Gold, Blues Brothers 2000, W...	24.0	1	0	[0, 1, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16]	[0.01522064589719826, 0.0157196834675982 05, 0....	301	0.257576
4	5	[40, 163, 24, 194, 189, 426, 101, 94, 169, 408...	[439, 225, 110, 454, 424, 376, 231, 377, 394, ...	[To Wong Foo, Thanks for Everything! Julie New...	33.0	0	1	[0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14...	[0.04958294717330862, 0.3229842446709917, 0.03...	174	0.393939
...	...	...	...	...	...	...	...	...	...	...	...
938	939	[106, 1028, 285, 411, 15, 717, 283, 9, 280, 59...	[931, 890, 266, 680]	[Star Trek: First Contact, Secrets & Lies, Emm...	26.0	0	5	[0, 1, 2, 3, 4, 5, 7, 8, 10, 11, 12, 13, 14, 1...	[0.036572324802167246, 0.0495829471733086 2, 0....	471	0.287879
939	940	[316, 354, 873, 879, 193, 382, 651, 516, 70, 1...	[153, 355, 610, 1401, 319, 549, 4, 343, 358, 7...	[Peacemaker, The, Picture Perfect, Wedding Sin...	32.0	1	4	[0, 1, 2, 3, 4, 5, 7, 9, 10, 11, 12, 13, 14, 1...	[0.010836244385827334, 0.0089826762671989 73, 0...	313	0.378788
940	941	[147, 124, 294, 1, 273, 919, 300, 408, 298, 76...	[222, 358]	[Toy Story, Return of the Jedi, Lone Star, Wai...	20.0	1	5	[0, 1, 2, 3, 4, 5, 7, 11, 12, 13, 14, 15, 16]	[0.026662864475654097, 0.0304412917943965 2, 0....	1007	0.196970
941	942	[117, 174, 310, 265, 705, 79, 99, 662, 272, 31...	[269]	[Aladdin, Roman Holiday, As Good As It Gets, B...	48.0	0	11	[0, 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, 1...	[0.030940329364796463, 0.2318385969915 1635, 0....	259	0.621212
942	943	[58, 194, 22, 205, 234, 385, 1330, 625, 56, 13...	[570, 1067, 443, 595, 1011, 549, 412, 51, 421,...	[Jaws, Stand by Me, Sneakers, Maverick, Net, T...	22.0	1	5	[0, 1, 2, 3, 4, 5, 7, 8, 10, 11, 12, 13, 14, 1...	[0.06259356954444999, 0.3359948670421330 5, 0.0...	28	0.227273

943 rows × 11 columns

Metrics

Since we don't have online data, we can't use the A/B testing or measuring watching time as was used online in the article. Thus, we used the offline metrics suggested – Accuracy and HitRatio.

Conclusions

Building the dataset

- The data pre-processing part was demanding and required a massive amount of time to prepare and make the data ready for the model.
- Firstly, while building the dataset for the model, we had to use all of the available data from all of the 3 datasets we had, and combine them together to suit our anchor model. That required a lot of merging data from one table to another. This was made to by complied with the data structure described in the paper.
- After joining the data together, we had to make adjustments to the data so it can be served into the model. "Categorial data" such as gender, genres and occupations, had to be "categorized" into numbers so the could be served to the model. "Continuous data", such as age, movie age, time since watching, and more, had to be normalized and than manipulated (squared and rooted) to be served to the model.
- In the end, we had to use Keras's padding funciton to make all the data the same size for the model - because some users had more rated movies than the others.

Model architecture

- The model architecture described in the paper included 2 NNs - Can Gen network and a Ranking method. As suggested in class, we combined those two networks and feeded the model with all of the data from the beginning. The structure of the two networks is pretty similar, except for the loss function in the end and a few additional features given to the Ranking NN.
- Due to the lack of time, we couldn't add the squared and rooted normalized continuous features.
- Our suggested improvement included even a more complex strucutre - that in real world could be too complex. Nevertheless, we beleive that it can increase the accuracy of the rec sys of YouTube.

Measuring the results

- In contrast to the paper, we missed a few crucial features to enter the model, and also our measuring techniques are different. We couldn't use A/B testing as the paper suggested, and thus we used HitRatio and Accuracy as they suggested. (the paper uses also precision and recall).
- We can see that we didn't get high accuracy in the original model and the improved one - that is apparntly due to few reasons.
- The first reason could that the model is just too complicated for this small dataset. We used a lot of features but the dataset is pretty small to provide different samples.
- The second reason the results are not that impressive is probably because the IDs of the movies don't really represent anything "informative", and they are pretty random. In other words, there is no real correlation between the ID of the movie and the next movie to watch (i.e., as high the movie' ID is, the more "likeable" the movie is). If we compare that to a linear regression problem in which we try to predict the price of a house based on its size, number of bedrooms, and number of bathrooms, we can assume that as the number of each of those increase, so do the price of the house. Nevertheless, in this case, since we trying to predict the next movie - we see that this information is not helping.
- To solve this problem, we wanted to use the movie name as the "search history" of the user, but since the model understands the movie name as a whole (meaning that "Batman 1" and "Batman 2" are completeley different movies, and each of them will get a different embedded number, then there was no point in entering them to the data, as we will get the exact same thing as the movie ID.
- Another solution we had in mind was to use NLP models such as Word2Vec - to actually understand the meaning of similar words, and by thus increase the accuracy of the model. By that, "Batman 1" and "Batman 2" will be "similar" to each other, and a user that liked "Batman 1", will have higher probabilities of being recommended "Batman 2". In addiiton, movies with names that include similar words (such as "lughing something" will probably be a funny movie).
- We can see that our solution helped in incresing the Hit Ratio of the model.

Hyper-parameters optimization

- We can see that the numbers didn't do much to the loss, and the losses convereged more or less to the same areas. We set the early stopping of the model to stop when the val_loss is not improving more than 3 times in a row. We noticed that a lot of the training processes stopped very early - that is probably due to overfitting of the model. This could derive from a few causes -
    - Overfitting - can occur when the model is too complex and starts to fit the noise in the data instead of the underlying patterns. As a result, the model performs well on the training data but poorly on the validation data. Overfitting can lead to an increase in validation loss as the model becomes more complex.
    - Learning rate - the learning rate determines the size of the steps taken during training. If the learning rate is too high, the model may overshoot the optimal solution and start diverging, leading to an increase in validation loss. Nevertheless, we tried using a variety of recommended learning rates, but that also didn't solve the problem.
    - Dataset size - as mentioned, we think that this is the main reason for the early stopping.
- In summary, we think that the model architecture is just not suitable for the problem we had, using this dataset.

References

Deep Neural Networks for YouTube Recommendations, Paul Covington, Jay Adams, Emre Sargin, 2016