Group 12

Final Project: Recommendation Systems

Paper: Deep Neural Networks for YouTube Recommendations

By Paul Covington, Jay Adams, Emre Sargin

February 2023, RUNI

Overview

In this project, we research the paper by Covington, Adams and Sargin, presenting the YouTube recommendation system. The authors open by listing the main challenges faced by YouTube:

1. Youtube needs to handle millions of items, at a short serving time. Classical recommendation algorithms fail on that scale and so a two stage solution was used to mitigate the problem.
2. In order to be relevant, the recommendations need to incorporate the latest actions by the users and the latest videos uploaded. Therefore, the model needs to include a content based component, which also helps item cold start.
3. The available data is very noisy. With millions of options, the feedback data is very sparse and is mostly implicit, with many factors affecting the user's choices. The algorithm still needs to overcome this by using robust features.

System Approach

The YouTube model comprises two networks, working in succession and trained separately.
The first network extracts a few hundreds of candidates out of millions of videos. These videos are meant to be broadly relevant to the user (high precision), based on collaborative filtering.
The second network then has the resources to rank the candidates using much more features on only hundreds of items and obtain high recall. The final top ranking items are presented to the user.

Candidates Generation Model - A user representation is learnt, based on an average embedding of previously watched videos, search history and general user data (gender, age, demographics). The model tries to predict the next watched video out of all videos as part of a multiclass classification problem. The similarity measure used is the dot product between final user and item embeddings. In training, a positive sample is defined by a fully watched video. In practice, training is done against a subgroup of negative samples with importance weighting in order to speedup the training. At serving time, a sublinear KNN model built with all embeddings is used for fast retrieval of the candidates.

Ranking Model - Each candidate from stage 1 is now given a score, as the output of a DNN. The inputs are based on user history, additional user data, item features and impression data. YouTube further optimizes over the expected watch time by weighting the samples with the watch time.

Dataset, Project Approach and Adjustments

Since we do not have the YouTube dataset, we chose to apply this model to the well known MovieLens 1M dataset. This dataset, primarily holding movie ratings, has rating timestamps that can be used for user history, and additional (not so rich) user and item content. In the attempt to comply with the YouTube model, the following choices were made in producing the training, validation, and test sets:

Candidates Model

- The full viewing history of each user is created by sorting the ratings by timestamp. From that, a sample is defined by any of the movies chosen as the target, while all previous movies define the movie history. This creates samples with changing length of viewing history, including empty history (cold start). The history is padded with 0's or cut to a constant length of 50.
- The test set is chosen as the last rating event for each user, and the validation set is the second to last.
- As recommended in the paper, we give equal representation to each user with 50 training samples per user. Analysis and justification can be found in the relevant notebook.
- The basic user data is added to the candidates model. Age and occupation are embedded in a small space (~log(size)). Gender is added as a binary feature and "sample age" is taken as the normalized time from the target timestamp to the latest timestamp in the entire dataset.
- We did not use movie titles (as search history) or zipcode, estimating low cost effectiveness.
- 100 negative samples are chosen per sample, excluding everything the user has ever seen. With the implicit approach, the target is labeled 1 and the negatives are 0. Ratings are not used here.
- The model is trained with cross-entropy loss, with importance weights for the negative samples
- For candidates serving and evaluation, a full KNN model is fitted after each epoch. Several solutions for approximate nearest neighbors were attempted but either had compliance problems with Google colab implementation, or showed inferior performance.

Ranking Model

- After training the candidates model and obtaining the candidates for each original sample, a second stage of processing is needed to produce datasets for the ranking model. In the ranking model each target movie is scored and compared to the other suggested candidates. Each such candidate is now set as a separate sample, where item content is added to the viewing history.
- The item content added includes the embedded candidate ID, the genre classification, the release year (normalized using quantiles) with powers, and the normalized average rating in the dataset.
- Two labels are given: one to indicate if the movie was viewed by the user, and another for the user rating. During training, the loss is weighted by the movie rating as a substitute to viewing time in the YouTube model. This way, the model learns the log-odds that approximate the expected rating. Movies that were not viewed are still given a "rating weight" of 1 accordingly.
- Due to the huge amount of potential samples, we break apart only one candidate-set for each user, belonging to the last within its original samples.

- There are a few tricky points here:
    a. Since the YouTube model at this stage is interested only in the expected watch time, there may be more than one positive label (the target). These include all suggested candidates that the user watched. Target rank evaluation is only part of the objective.
    b. The candidates model might not return the target movie as a candidate at all. While we train with the target, we perform both ranking evaluations: one of the ranking model alone, assuming we have the target and looking for its rank within all candidates, and one where the result is 0, if the target is not included in the candidates.
- The DNN model produces a single logit score and is trained with BCEWithLogitsLoss.

Baseline Algorithm - MF

We chose the classic Matrix factorization with implicit feedback approach as a baseline for comparison. Each sample is only made up of a user index and item index pair, labeled 0 or 1.
The exact same data from stage 1 is processed to produce the datasets. The test set is the last watched movie for each user and the validation is second to last. The same negative samples are used for each positive training set, which makes up a huge ratio of 1:100.
The model is required to predict the last movie out of all possible movies. Therefore, evaluation is done by ranking the score of the target movie relative to all other movie scores.

Improvements

Two improvements were attempted. Both were integrated into the candidates model:
1. Applying an RNN to the sequence of viewing history embeddings, instead of plain averaging. If there is some trend in the sequence itself, this may improve the results. Both GRU and LSTM were attempted. Note - history padding had to be reversed, so 0's come first.
2. Using Markov-chain transition predictions. We consider the viewing history as a Markov chain where the movie's genre represents the current state. For each user, we calculate the transition probability matrix between genres (normalized). Each sample is added a vector representing the predicted genres distribution according to the last movie's genre. In the model, the predictive dot product will be made out of two parts: the user final representation is multiplied by the movie embeddings as before, and the predicted genre vector is multiplied by the candidate genre vector.

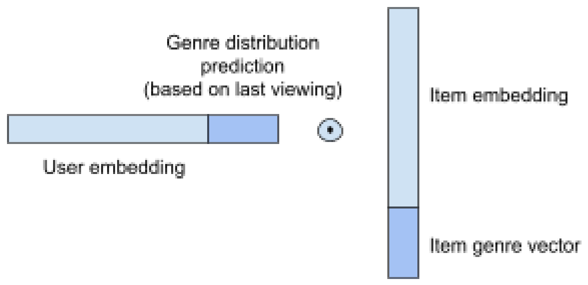


Figure 1. Illustration of improvement 2 dot product

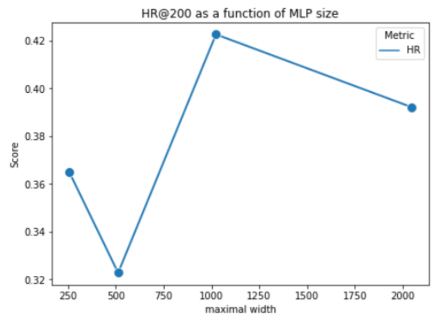
Experiments and Results

Candidates model evaluation

Evaluation is mainly based on the ranking of the target item within the returned candidates (note - not within the negative samples). We explore returning either 100 or 200 candidates and so the primary parameter tracked is HR@200. We also calculate MRR and NDCG @100/200.

The paper also presents MAP results, but does not specify how this is measured. To calculate MAP we used the softmax output to simulate probability and aggregated the target plus another negative sample result for each user.

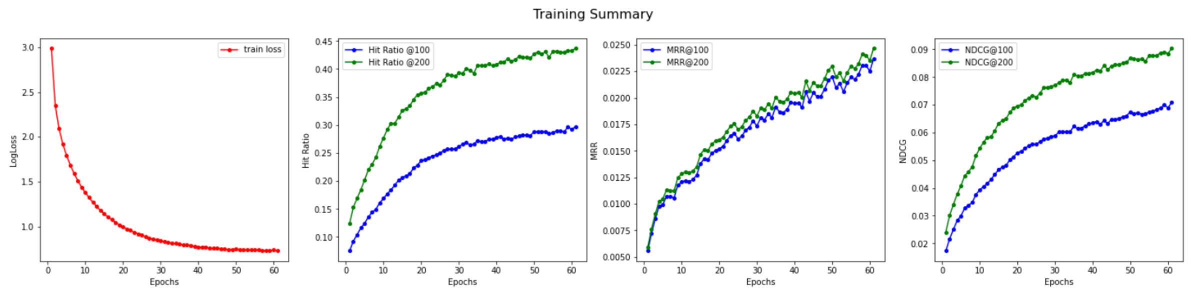
We searched the hyperparameters space in low resolution for learning rate and MLP size. Evaluation was done on the validation set. The results for MLP size are presented here, as a function of the first layer width. All final layers return to the original embedding size of 256.



The best HR@200 results were obtained with layers: 1024 ➔ 512 ➔ 256

Using these parameters we trained the model and finally evaluated results on the test set.

The training summary is plotted below (validation set).



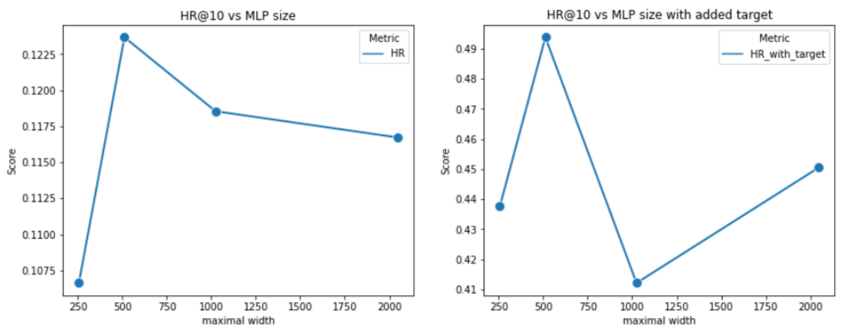
Conclusions:

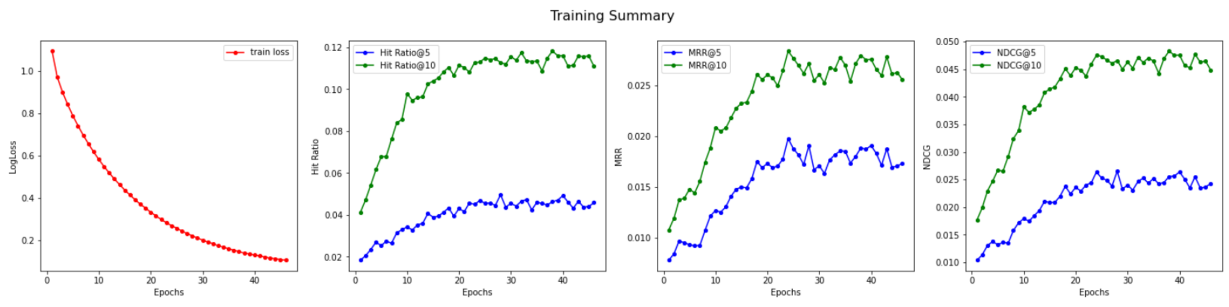
- In 28.7% of the cases, our target item is one of the 100 first candidates retrieved. It would be 41.7%, if we returned 200 candidates. This is our starting point for the ranking stage.
- While the Hit Ratio is very different for returning 100 vs 200 candidates, the MRR metric is less efficient, since the additional hits are positioned towards the end of the long list.

- Interestingly, the evaluation loss is increasing while the accuracy is increasing (graph in notebook). This seems necessary to gain higher recall. It is possible that other choices are better for real A/B testing. We did not have time to explore regularization options.
- The same trend can be found in the MAP results (graph in the notebook). MAP is shortly increasing, but then starts to decrease. Since MAP is using one threshold for all users, its objective is also not necessarily inline with ranking, and it may not be a good predictor.

Ranking model evaluation

We measure all ranking metrics at 5&10. As mentioned, this is done both for the evaluation of the returned candidates only (whether or not the target was returned), and when adding the target to the list to evaluate the ranking model in itself.
We ran a coarse hyperparameters search for the learning rate and MLP size. The results as a function of the MLP size are presented below. The best performance parameters were 512 ➔ 256.
We trained the model with the best parameters and evaluated it on the test set.





Conclusions:

- The combined candidates and ranking model was able to place the target movie within the first 10 places out of 3952 in almost 10% of the cases.
- If the target item is considered as a candidate, the ranking model is able to place it within the first 10 places in about 49% of the cases.
- It seems that the bottleneck for performance is the candidates producing model.

Baseline MF evaluation

Due to the high ratio of negative to positive samples, a training epoch takes very long. The evaluation shows that the loss is close to convergence after the first epoch. Hyperparameters search was done for learning rate and latent dimension size parameters (graph in the notebook). The basic MF model, without content, performs poorly on the general task relative to the YouTube model, reaching HR@10 of about 2.3%.

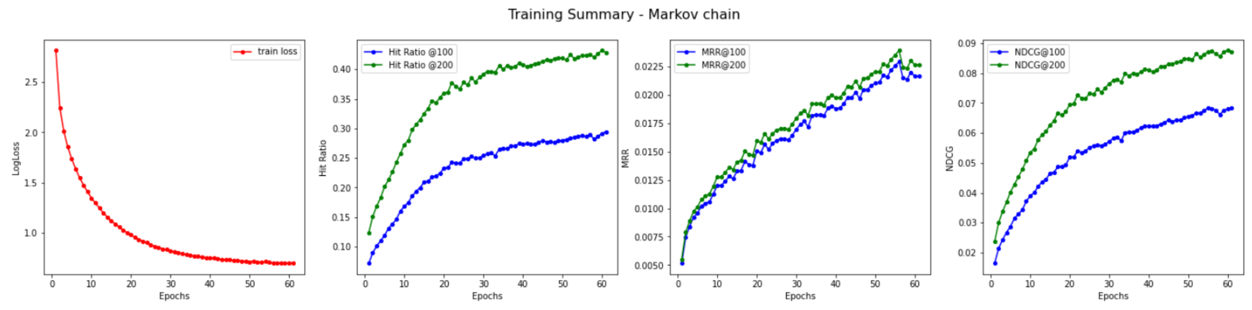
Improvements - RNN

We tried both GRU and LSTM as RNN units. The configuration was always one-directional with 2 layers. For the final RNN representation, we tried using both the last hidden state, as well as the average of all history states. We were not able to improve the results beyond those of the basic averaging model. LSTM performed much better than the GRU and got close to our best results with 40.3% HR@200

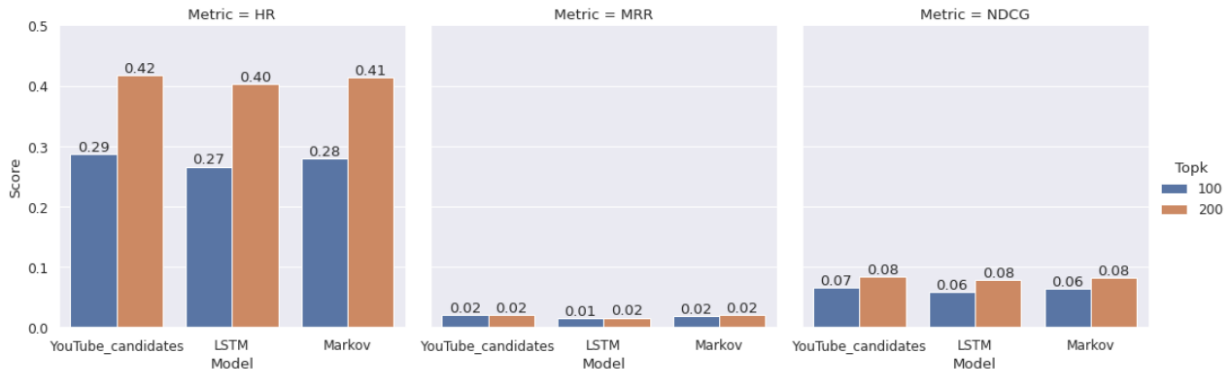
Improvements - Markov Chain

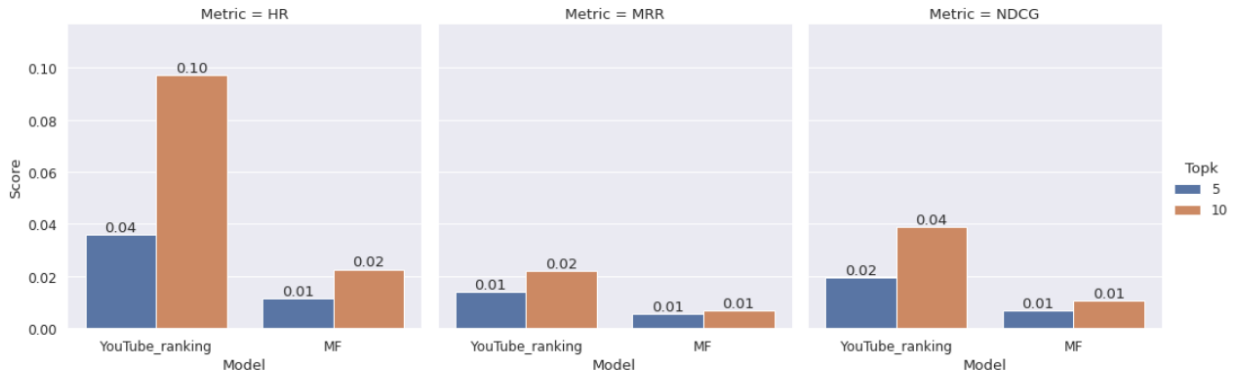
In this experiment, we are adding extensions to the user and item embeddings, representing the expected genres and characteristic genres respectively. Since this extension is parallel to the embeddings as calculated in the basic model, we did not optimize the basic model again, but chose to experiment with the weight given to each component in the dot product. The results:

1. Best results were obtained with equal weight to the original embedding and the genres.
2. The new model results were very close to those of the basic model.
3. The best HR@200 result was 41.4% on the test set.



Model Performance Comparisons





Summary

- We implemented the YouTube recommendation system on the ML-IM dataset and showed that the model reached HR@10 for the next movie (out of all items) in almost 10% of the cases. This is compared to 2.3% for the MF algorithm.
- The candidate producing model seems to be the bottleneck for the process.
- Attempts at improving the candidates model did not show significant improvement.
- Our solution is trained on viewing history of all lengths to give good performance on user cold start.
- Training times are presented with the results. However, it is very difficult to compare times. For example, the MF model training should be compared to both candidates and ranking models together, but they are run separately and use intermediate evaluations.

Future work suggestions

- Create a unified serving model from both models. Item data for the candidates needs to be added to the ranking model inputs in between.
- Try this model on data with richer features.