Deep neural network for YouTube video recommendation – final project

Recommendation systems 2023, Reichman University. Guy Zepko, Assaf Caftory

Deep Neural Network for YouTube video recommendations/ P.Covington, J.Adams, E.Sargin (Anchor Paper).

"In recent years, YouTube has become the go-to platform for video sharing and consumption. With over 2 billion monthly active users, it has become an integral part of the digital landscape. Its success has been attributed to its personalized recommendation system, which suggests relevant videos to users based on their viewing history and preferences. As our final project, we will investigate YouTube's recommendation system and propose a method to improve its accuracy and effectiveness using a different dataset." (ChatGPT).

This paper proposes a Deep Neural Network approach for recommendation systems based on implicit feedback using two types of neural networks: generating candidates and ranking candidates. Their main objective is to predict the next video a certain user will watch based on his previous watch history, search tokens, and other metadata related to the user and his interaction with the platform. Their recombination system aims to maximize the user's watch time. It was reported that their method outperformed previous matrix factorization approaches and linear and tree-based methods used at YouTube to predict watch time. According to the authors of this paper, their framework faces three main challenges:

- **Scale** - Many existing recommendation algorithms have been proven to work well on small problems but fail to operate on YouTube's scale.
- **Freshness** - YouTube has an enormous corpus of videos uploaded every second, making it dynamic.
- **Noise** - Due to sparsity and various unobservable external factors, historical YouTube user behavivariousntly hard to predict.

Method:

The authors describe a two-step algorithm that predicts the next video to recommend based on a user's previous interaction with the platform. During the first step, referred to as "Candidate Generation", they attempted to perform "smart" filtering to narrow their enormous video collections to hundreds of videos that may interest the user. Candidates should be filtered in a way that is efficient at runtime. As a result, its inputs are relatively simple and not modeling sophisticated interactions between the user and the videos. The input includes the user's watch history, search history, the age of the watched videos, and the gender of the user. By doing so, the ranking step can be based on a much smaller subset of the original corpus, enabling more complicated calculations. Based on user U and context C, they describe this step as an extreme multiclass classification, where the prediction problem is to accurately classify a specific video watch W_T at time t among millions of videos (i classes) in a corpus V.

$$P(w_t = i|U, C) = \frac{e^{v_i u}}{\Sigma_{j \in V} e^{u_j u}}$$

where $u \in R^N$ represents a high-dimensional “embedding” of the user, context pair and the $v_j \in R^N$ represent embeddings of each candidate video.

According to the authors, the "example age" feature is a key feature. By modeling the video's age with respect to "Now," the model can be "fresher" and assign more accurate probabilities than it would be without using these features.
Another neural network called "Ranking" will rank those candidates in the second step. As fewer candidates are to evaluate in this step, more personalized features will be considered. Described in the paper are a number of features, both continuing and categorical. For example, features describing a user's prior interactions with the item and similar items, take into account the user's past history with the channel that uploaded the video being scored - how many videos have the user viewed from this channel in the past? When was the last time the user viewed a video on this topic? In practice, they use hundreds of features in that step roughly split evenly between categorical and continuous, where categorical features were embedded, and continuous features were normalized in different ways. This ranking model has been trained using logistic regression to predict which video of those candidates the user will watch for the longest period of time. The training examples are either positive (the video impression was clicked) or negative (the impression was not clicked).

The objective is to predict the expected watch time, in real-time, the user will then be presented with a subset of videos sorted according to their score.
As part of their research, they used a technique called "weighted logistic regression" which involves weighting positive samples according to the observed watch time on the video, while negative samples are weighted according to a unit of weight. A cross-entropy loss was then used to minimize the error between the model predictions and the annotations of the samples. Additionally, they discuss the importance of withholding discriminatory signals from the model to achieve good results. Otherwise, the model would overfit the surrogate problem and not transfer well to the homepage.

Our Work and Improvement

Data preparation

This approach aims to predict which video a particular user will watch next, which means we deal with sequential data. Additionally, we did not have access to the original dataset nor the resources to handle such an enormous amount of data, so data engineering was a major part of our work. Since Movilens 1M contains a timestamp for each entry, we used that dataset. The following steps were taken to prepare the data:

- Gathering entries per user from ratings matrix
- Sort those entries by their timestamp.
- Define a window size (we tried several sizes between 1 - 5).
- Iterate on the entries with a window and create a learning example such that the window is the watch history, and the label is the next movie in line.
- Additionally, we store supplementary data such as user gender occupation, etc.
- For testing we randomly sampled 1 training sample from each user

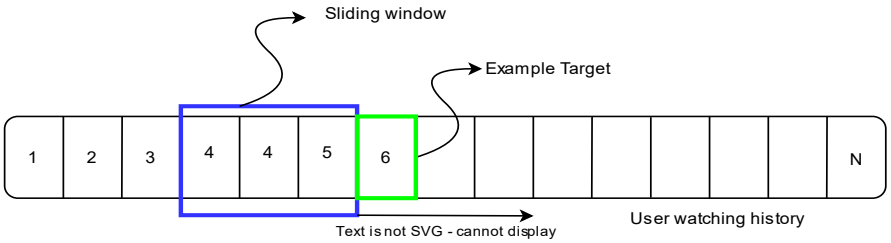
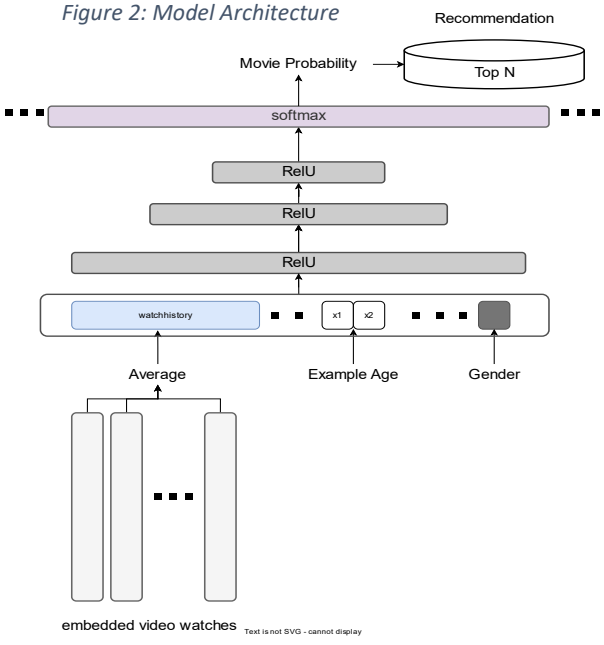


Figure 1 Example of sequential trainig samples generation

Algorithm modification

As we use a much smaller dataset than the original paper, the two-step approach presented in the paper is not necessary. The first step of narrowing down the enormous corpus is redundant, so we used only one network to predict the next movie a user will watch. As we do not have the watching time for each movie, only the timestamp, we decided to use the architecture of the candidate generation model to assign a probability for each movie in the dataset and to recommend the top N. Using this approach, we construct an artificial neural network similar to the candidate generation approach described in the paper. There are similar inputs as well, consisting of the user's watch history, example age, and gender. Each movie is embedded with a different vector, we then average those embeddings to create a fixed-size feature. There is also a gender feature embedded. Afterward, we concatenate all these features into one input vector for our model. The embeddings are learned during the learning process. To train the model to predict the next video based on the inputs, we used "Categorical cross entropy" loss. Amount of movies in our data set is reflected in the size of the last layer.

Figure 2: Model Architecture



Evaluation

We devised a unique method to evaluate our performance since the paper did not specify how the MAP metric was implemented in the context of sequential data. We modified the metrics we used in class (MRR, Ndcg, Hit-Rate)@k in a similar manner to Homework 3. In other words, for each training sample, we computed the index i for the first element of the sorted model prediction which also appears in the relevant items (the method of finding relevant items is described in $figure\ 3$).

$$MRR = \frac{\sum \frac{1}{i+1}}{n} \quad \bullet \qquad \bullet \quad Ndcg = \frac{\sum \frac{1}{\ln i+e}}{n} \qquad HitRate = \frac{\sum \begin{cases} 1 \ if \ i \neq 0 \\ 0 \quad else \end{cases}}{n} \quad \bullet$$

We modified those metrics as they are not suitable for sequential data. We may undervalue our performance if we use them without modification. As maintained, the paper approach is to predict the exact movie a user will watch in the future based on his interactions history. That is very particular and challenging task. Using that only particular movie to evaluate our model does not necessarily provide evidence of the model's true performance since our true goal is to recommend relevant items to users. So instead of use only the next movie in line MRR, NDCG, and, Hit-Rate metrics for a given training example are computed using the entire user viewing history up to the example target, as shown in $figure\ 3$ instead of only the target item.

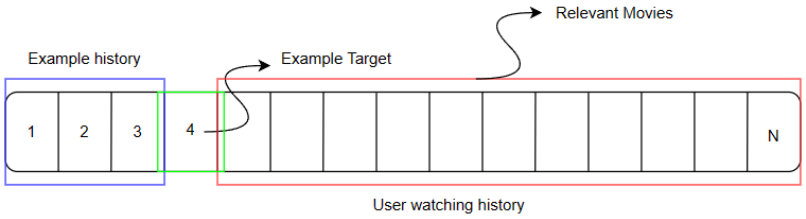


Figure 3: Relevant Movies. Consider user u watched N movies. The blue square represents example history, green square represents the example target movie and the red square represents all the relevant items that will take in consideration when computing MRR, Ndcg and Hit-Rate for that example.

Improvement

In our setup, it was difficult to come up with an improvement because in addition to the novel approach in this paper utilizing two steps prediction (which we were not able to apply), one of the most significant innovations was the feature engineering they used to model implicit interactions between users and the platform.

In this section, we use a very similar approach to what we have already discussed, but we have added some more features to the model, as well as added embedding for the users. We did not have such metadata about the users and their interactions in our use case. Accordingly, we used all the metadata in the dataset, including the user's occupation, gender, time of watching, year of release, and movie genre. We used embedding layers for categorical features and normalization layers for numerical features. A major improvement in our work was the use of embedding layers for each user.

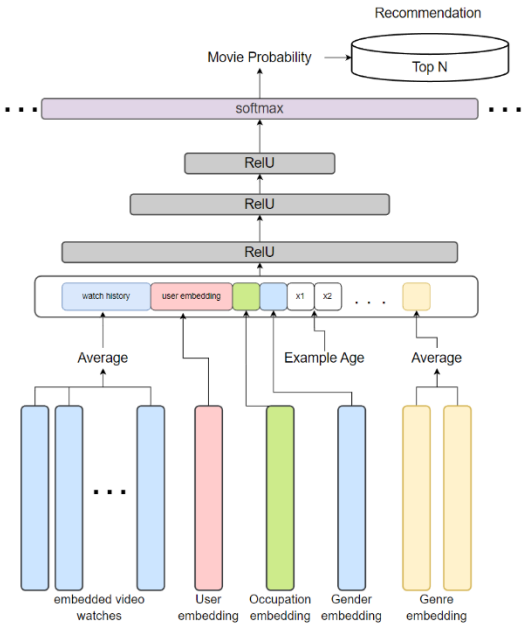


Figure 4: Costume model Architecture. All metadata in Movielens 1M enters as an inputs as well as user embedding layer to capture specific user behavior.

According to the paper, two assumptions were made:

- A good representation of the users is provided by their history of watched videos.
- The candidate generation's last layer can be used as embedding for the users (the 'surrogate problem').

In regard to the first assumption, in Movilens dataset the link between previously watched movies to the next one is much less powerful than YouTube dataset because individuals usually watch one or two movies per week, whereas, on YouTube, they may watch several videos per day. In regard to the second assumption, it is obvious that we cannot make use of this embedding. In order to represent the users, therefore we added an additional layer of embedding to our model to represent each user.

Popularity approach

As a baseline, we used a "Popularity" based approach. We evaluated it in a similar manner to our own models. Based on the dataset, we selected the top k (depending on which @k we used to evaluate) most popular movies. Based on the index of the first popular movie listed in the user's watch list, we calculated the MRR, Ndcg, and Hit-Rate for each user. The main difference in the evaluation process between our models and the popularity is that we considered all of the user's watch history for each user in the popularity evaluation. In contrast, in our model evaluation, we took only the positive items as described in the “Evaluation” chapter.

Results

We tried different hyperparameter settings including the window size for history watch, embedding size, and network size. Here are the results:

model	Embedding dimensions	Layers dimensions	Window size	MRR@10	Ndcg@10	Hit-Rate@10
popularity				0.225582164	0.455534949	0.756953642
costume	**512**	**(2048, 1024, 512)**	**1**	**0.394253456**	**0.540855736**	**0.736092715**
costume	256	(2048, 1024, 512)	1	0.390683538	0.541233112	0.741390728
costume	64	(2048, 1024, 512)	1	0.389967939	0.536277712	0.730794702
costume	1024	(2048, 1024, 512)	1	0.389583333	0.539081346	0.737913907
costume	1024	(2048, 1024, 512)	3	0.377282009	0.52842993	0.731125828
costume	256	(512, 256, 128)	1	0.374861111	0.522232429	0.717549669
costume	1024	(2048, 1024, 512)	2	0.374136576	0.523481398	0.723344371
costume	512	(512, 256, 128)	1	0.372685864	0.522271449	0.723013245
costume	16	(2048, 1024, 512)	1	0.372457558	0.521663846	0.720364238
costume	1024	(512, 256, 128)	1	0.371372595	0.521567907	0.723178808
costume	256	(2048, 1024, 512)	2	0.367893803	0.52029026	0.725
costume	512	(2048, 1024, 512)	2	0.365639717	0.517414673	0.720198675
costume	64	(512, 256, 128)	1	0.364911108	0.513126007	0.713245033
costume	512	(2048, 1024, 512)	3	0.364108851	0.515062817	0.718708609
costume	64	(2048, 1024, 512)	2	0.363730619	0.513637378	0.714238411
costume	256	(2048, 1024, 512)	3	0.363452447	0.51774534	0.725331126
costume	512	(512, 256, 128)	2	0.361553795	0.510920771	0.711754967
costume	64	(2048, 1024, 512)	3	0.360603056	0.51308991	0.717880795
costume	16	(2048, 1024, 512)	2	0.354244389	0.503755506	0.704470199
costume	256	(512, 256, 128)	2	0.353941711	0.504531778	0.706291391
costume	1024	(512, 256, 128)	2	0.353897627	0.506485815	0.710761589
costume	16	(512, 256, 128)	1	0.351262089	0.497800259	0.695860927
costume	16	(2048, 1024, 512)	3	0.348649217	0.49445823	0.691059603
yt	**512**	**(2048, 1024, 512)**	**5**	**0.347509264**	**0.493473574**	**0.689735099**
costume	1024	(2048, 1024, 512)	5	0.347189635	0.49996841	0.704966887
costume	512	(512, 256, 128)	3	0.344949806	0.495388438	0.699337748
costume	512	(2048, 1024, 512)	5	0.343625237	0.492583811	0.694205298
yt	512	(2048, 1024, 512)	3	0.343132884	0.490284363	0.688907285
costume	256	(2048, 1024, 512)	5	0.34296016	0.490886773	0.692218543

As shown in the table above, different learning results were obtained for different parameters. According to our evaluation it safe to say that the user’s embeddings including additional metadata its crucial for achieving more accurate predictions .The results of both the Costume (our) model and the YouTube model were best when the dimensions of the layers were $(2048, 1024, 512)$ and the embedding size was 512. Moreover, in the Costume model, the best results were obtained with a window size of 1 which is the shortest if them all, while in the YouTube model, the best results were obtained with a window size of 5 which is the largest. This further supports the

hypothesis presented in the article that the user's representation is determined by the history of his viewing. Although the measurements were differed largely between the YouTube model and the costume one, most of the hyper parameters were the same – embedding size and network layers dimensions.

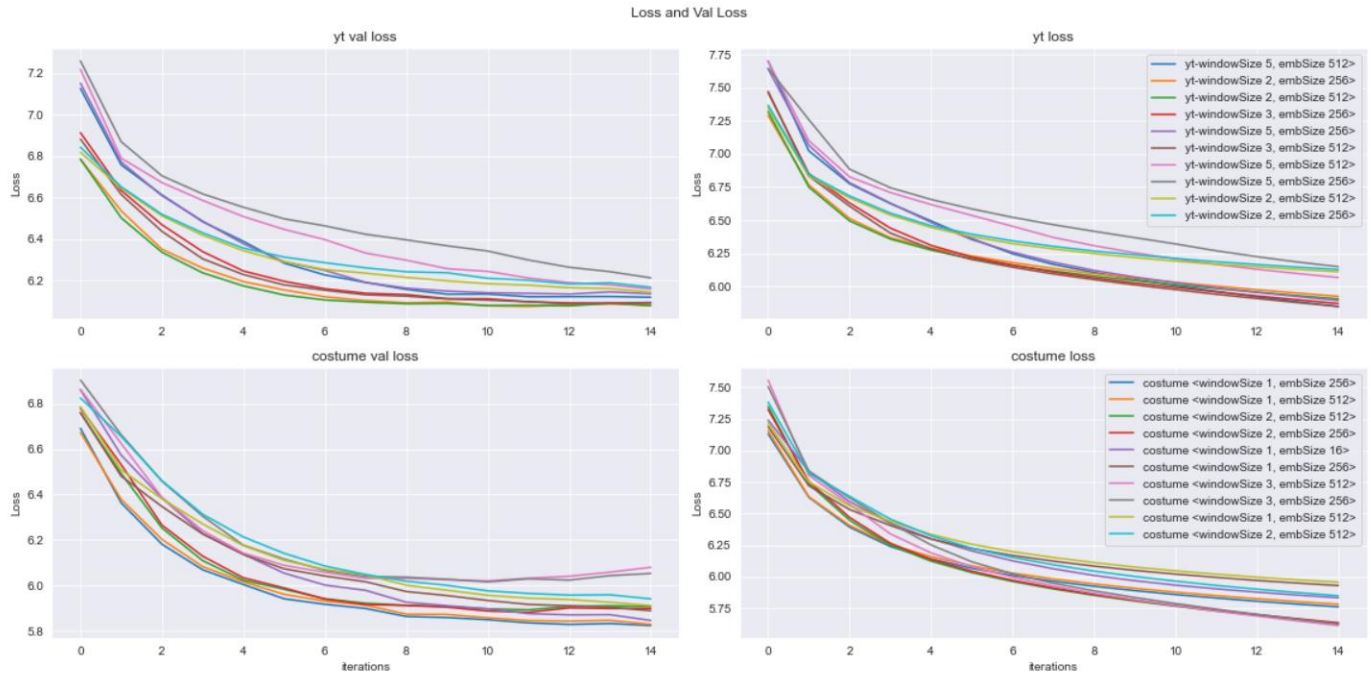


Figure 5 - Validation and training loss vs training iteration. Top 2 plots correspond to YouTube model whole the lower two plot are for our costume model. It shows that our model is more sensitive to overfitting.

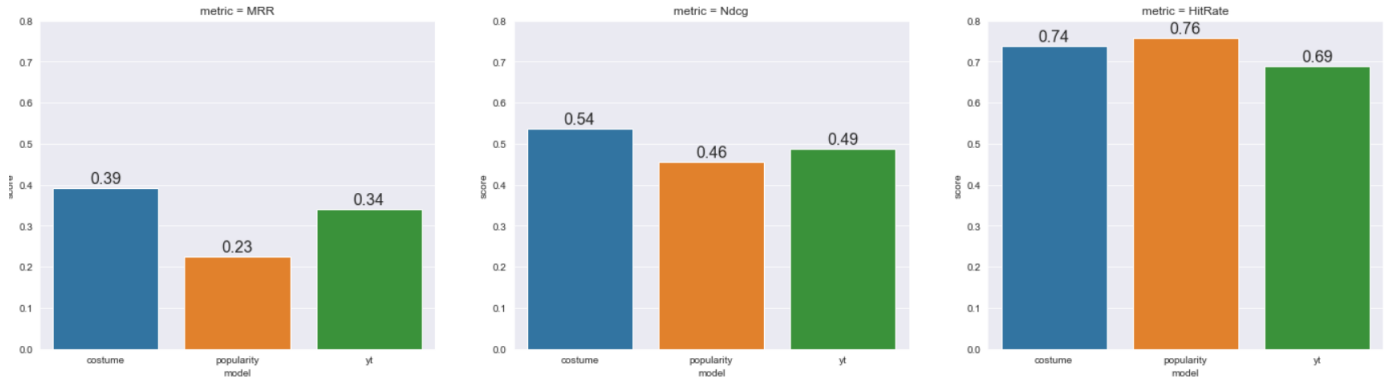


Figure 6: Dislpay top results in each metric between our YouTube model, costume model, and popularity based.

In Figure 5, we compare the performance of the three models, Costume, YouTube, and Popularity. Regarding MRR and NDCG, the Costume model outperforms the popularity and YouTube models, although the Hit-Rate of the popularity model is higher than the others. One possible explanation is that the evaluation method for the popularity approach was generally more generous as described in “Comparison” chapter. Additionally, those findings may indicate that the popularity model was able to recommend “somewhat” relevant items for more users than our models, but on the other hand, for users that our models could recommend relevant item our recommendation was much higher ranked which could lead to more successful recommendations overall.

In conclusion, we found that a lot of the paper's insights were appropriate, especially for YouTube data and their interaction types. But, there is some key component from their research that could be transferred to different dataset or domains, for example: Treating the data as sequential data, concatenation whole different type of features, categorical and continuous, attaching to embedding and lastly defining the problem as multiclass classification is appropriate in a different setup. We found out that adding an embedding layer to represent the users are an important feature that improved the performance by a big margin.