Deep Neural Networks for YouTube recommendation

Authors: Paul Covington, Jay Adams, Emre Sargin

Presenter: Omar Nada
Facilitators: Preston Engstrom, Serena McDonnell

22/04/2019

aisc

Intro

- Common Definitions used in RecSys
- Introduction and challenges
- Algorithm and Structure
- Results
- Discussion

Common Definitions for RecSys

- Implicit ratings
- Explicit ratings
- Precision
- Recall

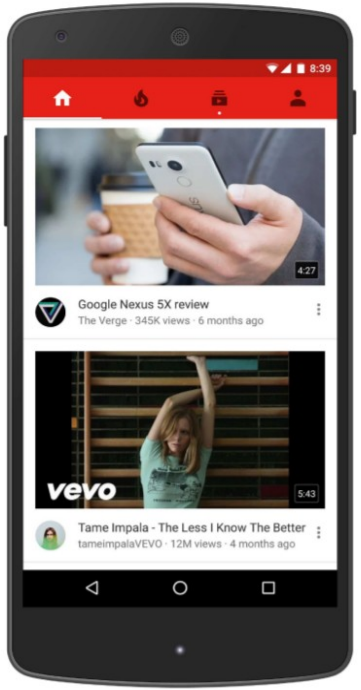


Figure 1: Recommendations displayed on YouTube mobile app home.

Introduction

- 2 stages model are used to generate top-N videos to watch NEXT
- Approximately one billion parameters are learned
- Training happens on hundreds of billion examples
- Testing is done offline using precision, recall and ranking loss as well as online using A/B testing
- Implicit feedback is used rather than explicit feedback because of the sparsity

Challenges

- Scale: Most algorithms work very well with small to medium data. Highly specialized distributed algorithms and many compute servers are required for this huge user dataset and video corpus
- Freshness: With videos being uploaded every hour (86,400 hours of videos are uploaded on daily basis). Cold Start problems? New content vs old content?
- Noise: Extremely hard to predict with the sparsity given

Structure

- As mentioned earlier, it’s a 2 stages model that brings down millions of uploaded videos to only dozens TO BE WATCHED NEXT
- The first stage is “Candidate Generation” narrowing it down from millions to hundreds focusing on precision
- The second structure is ranking where maximum scored videos are recommending first focusing on recall

Structure

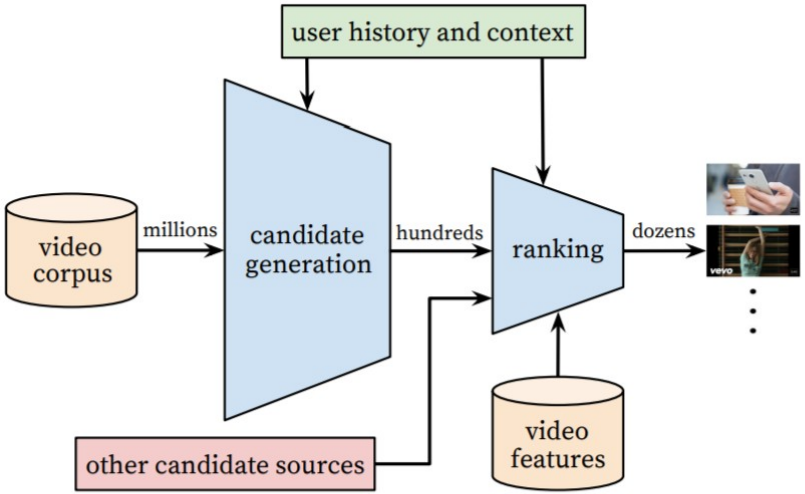


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

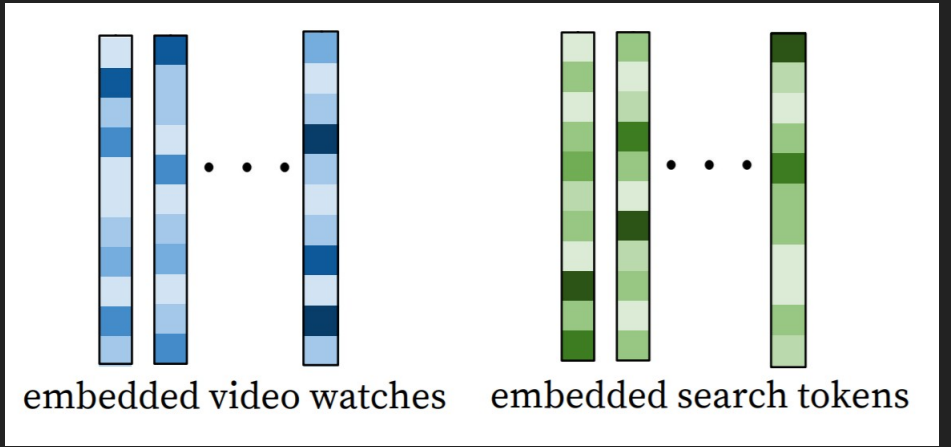
Candidate Generation

- In this step the main goal is to extract the most relevant videos for a user given the user's context and history
- The author's claim that this could be considered as non-linear matrix factorization (which makes sense)
- This step can be viewed as extreme multi-class classification where the goal is to correctly classifying a specific video to be watched from the corpus of videos using Softmax classifier
- u is high-dimensional embedding of the user and v is the embeddings for videos

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

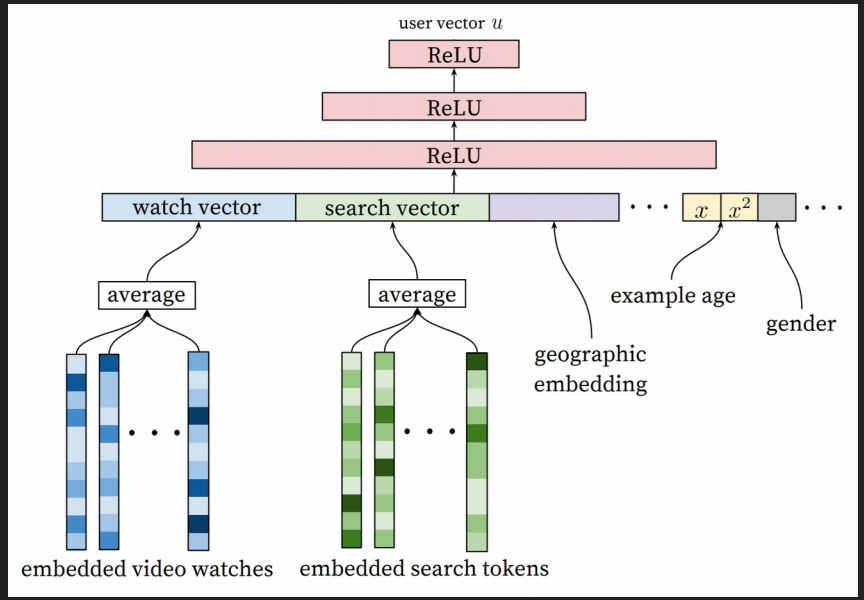
Candidate Generation Contd.

- Given there are millions of videos (classes) positive and negative sampling techniques are used then weightings are corrected by minimizing the cross-entropy loss
- User’s watch history represented with sparse variable-length sequence of IDs which is embedded into fixed size of dense inputs fed to the network



Candidate Generation Contd.

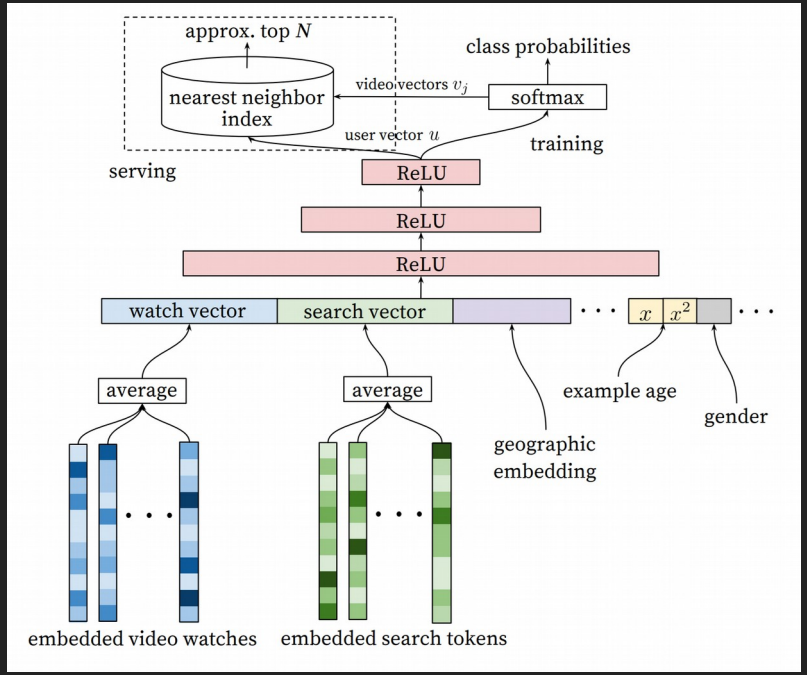
- These embeddings are then averaged, concatenated along with additional features as an input for the hidden layers
- The hidden layers are fully connected



Candidate Generation Contd.

- In training, a cross-entropy loss is minimized with gradient descent on the output of the sampled softmax

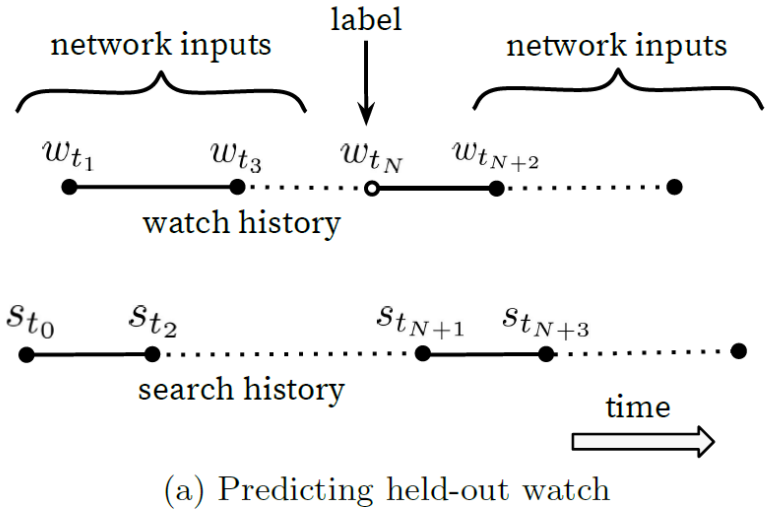
- At serving, an approximate nearest neighbor lookup is performed to generate hundreds of candidate video recommendations



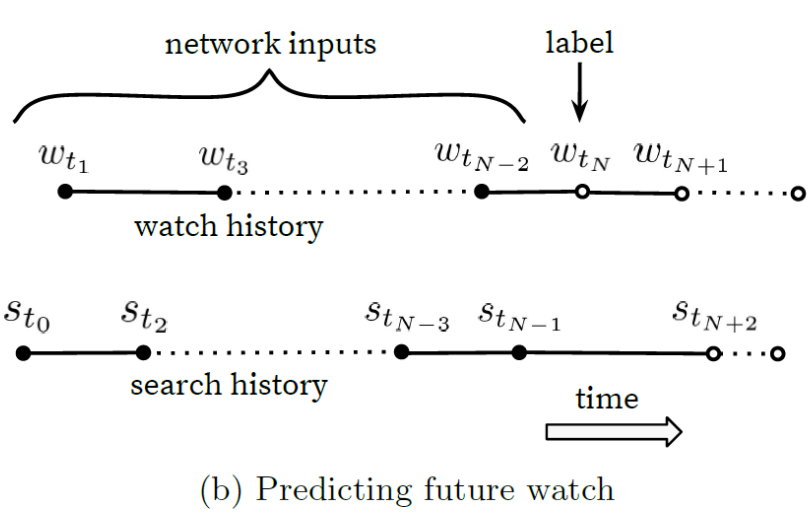
Labels and Context Selection

- All recommendations are generated from all YouTube watches (Eg: videos embedded on other sites) rather than just the recommendations produced earlier
- Fixed number of training examples are generated per user to allow fairness between frequent users and occasional users
- Withholding information from the classifier is done careful to prevent the model from exploiting the structure of the site (overfitting)
- The network’s input is everything PRIOR to the label (to prevent any bias)

Labels and Context Selection



(a) Predicting held-out watch



(b) Predicting future watch

Figure 5: Choosing labels and input context to the model is challenging to evaluate offline but has a large impact on live performance. Here, solid events • are input features to the network while hollow events ∘ are excluded. We found predicting a future watch (5b) performed better in A/B testing. In (5b), the example age is expressed as $t_{\max} - t_N$ where $t_{\max}$ is the maximum observed time in the training data.

Ranking

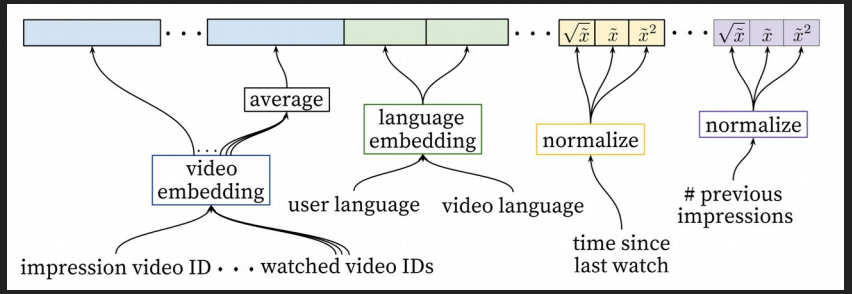
- The primary task of this is to use the impression data to rank the videos generated from candidate generation network by assigning independent score for each video (since there are only hundreds now) for a given user using logistic regression
- The final ranking is constantly being tuned based on live A/B testing results. It is a simple function of predicting the watch time per impression (instead of click rate as some videos are started and never ended)
- Features could be: Categorical, continuous, ordinal. It could be univariate or multivariate

Ranking Contd.

- Hundreds of features are used in the ranking model roughly split evenly between categorical and continuous features
- It’s observed that it is important to propagate information as features e.g. source nominated the video? The score assigned?
- Embeddings are used to map sparse categorical features into dense representations
- If the cardinality of the video IDs are too large, the last N is taken into consideration

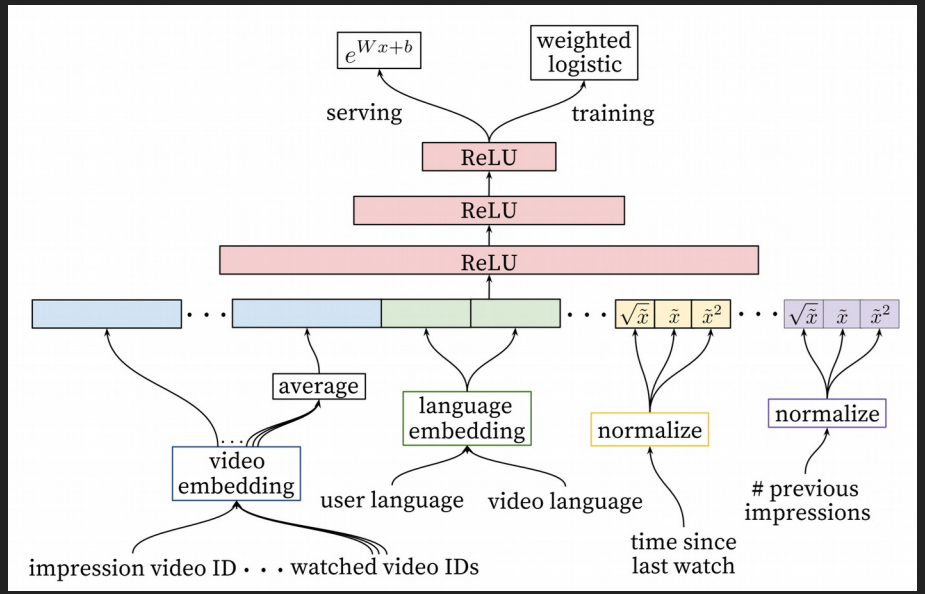
Ranking Contd.

- All the watched videos are embedded and averaged. The impression video is embedded and concatenated as well along with other features



Ranking Contd.

- The concatenated features are all feed to the hidden layers (again fully connected)
- In training, the parameters are learnt by backpropagation by minimizing the loss of the logistic regression
- In serving, the video ID with the highest score gets recommended



Ranking Contd.

- It’s observed that it is important to propagate information as features e.g. source nominated the video? The score assigned?
- Embeddings are used to map sparse categorical features into dense representations
- If the cardinality of the video IDs are too large, the last N is taken into consideration

5 Minute Break

Experimenting with feature and depth (Candidate)

- Adding features and depth improves precision on holdout data
- A vocabulary of 1M videos and 1M search tokens were embedded with 256 float to a maximum bag size of 50 recent watches and 50 searches
- The model is trained until convergence over all YouTube users, corresponding to several epochs over the data

Experimenting with feature and depth (Candidate)

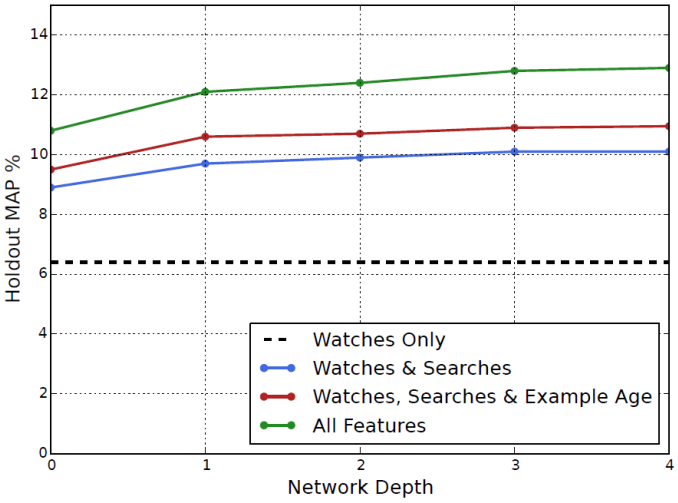


Figure 6: Features beyond video embeddings improve holdout Mean Average Precision (MAP) and layers of depth add expressiveness so that the model can effectively use these additional features by modeling their interaction.

Experimenting with hidden layers and results (Ranking)

- Results show that increasing the width and depth of hidden layers improve the results (CPU time needed increases as well)
- The most optimum results were using 1024 RELu then 512, then 256

Hidden layers	weighted, per-user loss
None	41.6%
256 ReLU	36.9%
512 ReLU	36.7%
1024 ReLU	35.8%
512 ReLU → 256 ReLU	35.2%
1024 ReLU → 512 ReLU	34.7%
1024 ReLU → 512 ReLU → 256 ReLU	34.6%

Table 1: Effects of wider and deeper hidden ReLU layers on watch time-weighted pairwise loss computed on next-day holdout data.

Discussion Points

- What could be changed to generalize the recommendations instead of only focusing for next video?
- Cold start problem ?
- Use attention instead of averaging