Deep Neural Networks for YouTube Recommendations

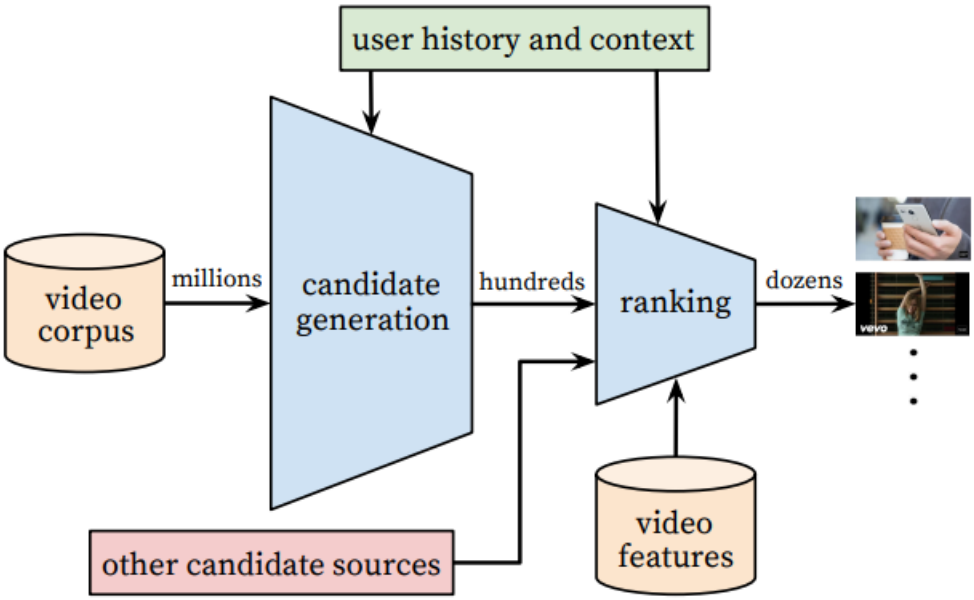


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

Note: In this document, red colored texts are the features that need to be implemented for model's success.

System Overview

The system is comprised of two neural networks: one for candidate generation and one for ranking.

The **candidate generation** network takes events from the user's YouTube activity history as input and retrieves a small subset (hundreds) of videos from a large corpus. The similarity between users is expressed in terms of coarse features such as IDs of video watches, search query tokens and demographics.

The **ranking** network assigns a score to each video according to a desired objective function using a rich set of features describing the video and user. The highest scoring videos are presented to the user, ranked by their score.

This design enables blending candidates generated by other sources.

For the final determination of the effectiveness of an algorithm or model, we rely on A/B testing via live experiments.

Candidate Generation

We pose recommendation as extreme multiclass classification where the prediction problem becomes accurately classifying a specific video watch w_t at time t among millions of videos i (classes) from a corpus V based on a user U and context C,

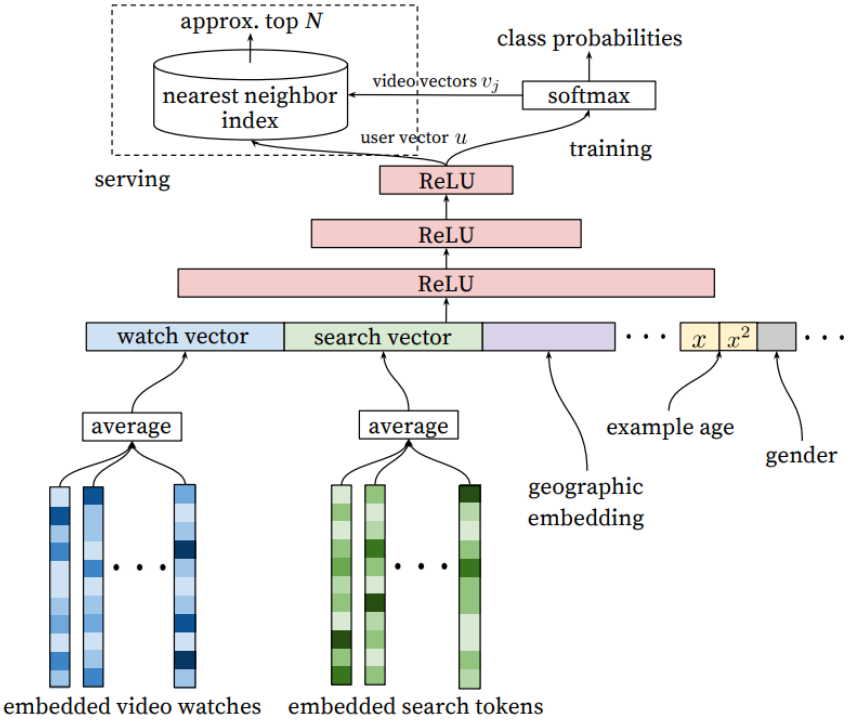
$$P(w_t = i|U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}}$$

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.

The task of the deep neural network is to learn user embeddings u as a function of the user's history and context that are useful for discriminating among videos with a softmax classifier.

Efficient Extreme Multiclass

To efficiently train such a model with millions of classes, we rely on techniques such as negative sampling or hierarchical softmax (first one works better).
At serving time we need to compute the most likely N classes (videos) between millions of items in order to choose the top N to present to the user, hence, we need an approximate scoring scheme in sublinear time. Since calibrated likelihoods from the softmax output layer are not needed at serving time, the scoring problem reduces to a nearest neighbor search in the dot product space. For this, we use hashing combined with general purpose nearest neighbor libraries.



Embeddings

The embeddings for user's watched videos (based on video data) and search tokens (based on search text) are learned during training. These embeddings are then averaged and fed to a neural network along with other inputs such as demographical information and example age (how old the item is, used for freshness purposes).

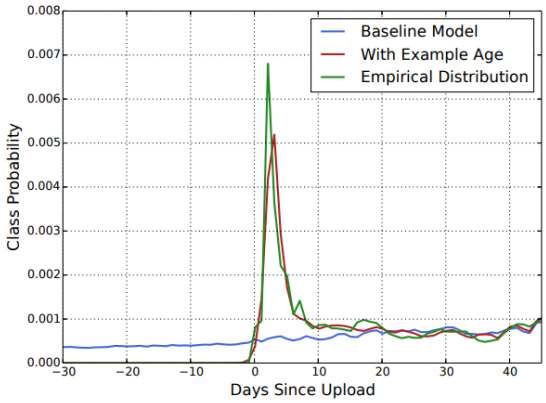


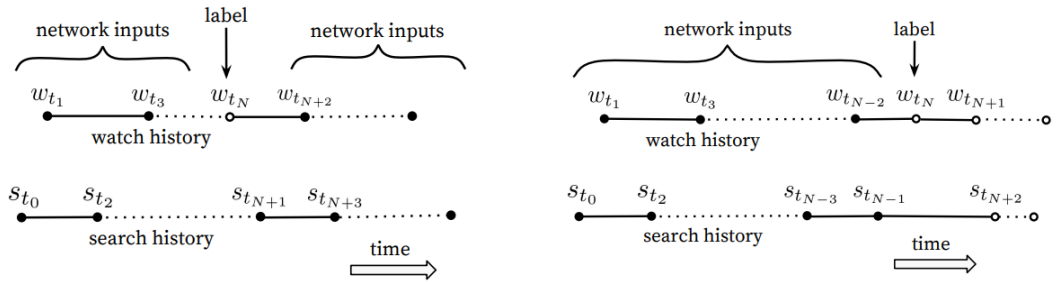
Figure 4: For a given video [26], the model trained with example age as a feature is able to accurately represent the upload time and time-dependant popularity observed in the data. Without the feature, the model would predict approximately the average likelihood over the training window.

Label and Context Selection

Training examples are generated from all YouTube watches (even those embedded on other sites) rather than just watches on the recommendations we produce. Otherwise, it would be very difficult for new content to surface and the recommender would be overly biased towards exploitation.

Another key insight that improved live metrics was to generate a fixed number of training examples per user, effectively weighting our users equally in the loss function. This prevented a small cohort of highly active users from dominating the loss.

For a good A/B testing performance, the model's access to some information must be constrained. For example, the model can use the user's last search and suggest the same items as in the search results, which causes a low training loss but poor live performance. By discarding sequence information and representing search queries with an unordered bag of tokens, the classifier is no longer directly aware of the origin of the label.



(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

During ranking, we have access to many more features describing the video and the user's relationship to the video because of the lower number of videos.