Log In Join

Back To Module Home

Machine Learning System Design

0% completed

Search Module

Mark Module as Completed

Candidate Generation and Ranking Model

Learn about candidate generation and ranking of videos based on user preferences.

We'll cover the following

3. Multi-stage models#

There are two stages, candidate generation, and ranking. The reason for two stages is to make the system scale.

> It’s a common pattern that you will see in many ML systems.

We will explore the two stages in the section below.

- The candidate model will find the relevant videos based on user watch history and the type of videos the user has watched.
- The ranking model will optimize for the view likelihood, i.e., videos that have high a watch possibility should be ranked high. It’s a natural fit for the logistic regression algorithm.

Candidate generation model#

Feature engineering#

- Each user has a list of video watches (videos, minutes_watched).

Training data#

- For generating training data, we can make a user-video watch space. We can start by selecting a period of data like last month, last 6 months, etc. This should find a balance between training time and model accuracy.

Model#

- The candidate generation can be done by Matrix factorization. The purpose of candidate generation is to generate “somewhat” relevant content to users based on their watched history. The candidate list needs to be big enough to capture potential matches for the model to perform well with desired latency.

- One solution is to use collaborative algorithms because the inference time is fast, and it can capture the similarity between user taste in the user-video space.

> In practice, for large scale system (Facebook, Google), we don't use Collaborative Filtering and prefer low latency method to get candidate. One example is to leverage Inverted Index (commonly used in Lucene, Elastic Search). Another powerful technique can be found FAISS or Google ScaNN.

Ranking model#

During inference, the ranking model receives a list of video candidates given by the Candidate Generation model. For each candidate, the ranking model estimates the probability of that video being watched. It then sorts the video candidates based on that probability and returns the list to the upstream process.

Feature engineering#

Features	**Feature engineering**
Watched video IDs	Video embedding
Historical search query	Text embedding
Location	Geolocation embedding
User associated features: age, gender	Normalization or Standardization
Previous impression	Normalization or Standardization
Time related features	Month, week_of_year, holiday, day_of_week, hour_of_day.

Read about embedding.

Training data#

- We can use User Watched History data. Normally, the ratio between watched vs. not-watched is 2/98. So, for the majority of the time, the user does not watch a video.

Model#

At the beginning, it’s important that we started with a simple model, as we can add complexity later.

- A fully connected neural network is simple yet powerful for representing non-linear relationships, and it can handle big data.

- We start with a fully connected neural network with **sigmoid** activation at the last layer. The reason for this is that the Sigmoid function returns value in the range [0, 1]; therefore it’s a natural fit for estimating probability.

> For deep learning architecture, we can use **relu**, (Rectified Linear Unit), as an activation function for hidden layers. It’s very effective in practice.

- The loss function can be cross-entropy loss.

Model prediction

Quiz on ranking model

1

Why do we use Sigmoid activation at the last layer of our ranking model instead of relu?

Question 1 of 2

0 attempted

Problem Statement and Metrics

Next

Video Recommendation System Design

Mark as Completed

Report an Issue