Re-ranking

In the final stage of a recommendation system, the system can re-rank the candidates to consider additional criteria or constraints. One re-ranking approach is to use filters that remove some candidates.

le: You can implement re-ranking on a video recommender by doing the following:

Training a separate model that detects whether a video is click-bait.

Running this model on the candidate list.

Removing the videos that the model classifies as click-bait.

Another re-ranking approach is to manually transform the score returned by the ranker.

le: The system re-ranks videos by modifying the score as a function of:

video age (perhaps to promote fresher content)

video length

This section briefly discusses freshness, diversity, and fairness. These factors are among many that can help improve your recommendation system. Some of these factors often require modifying different stages of the process. Each section offers solutions that you might apply individually or collectively.

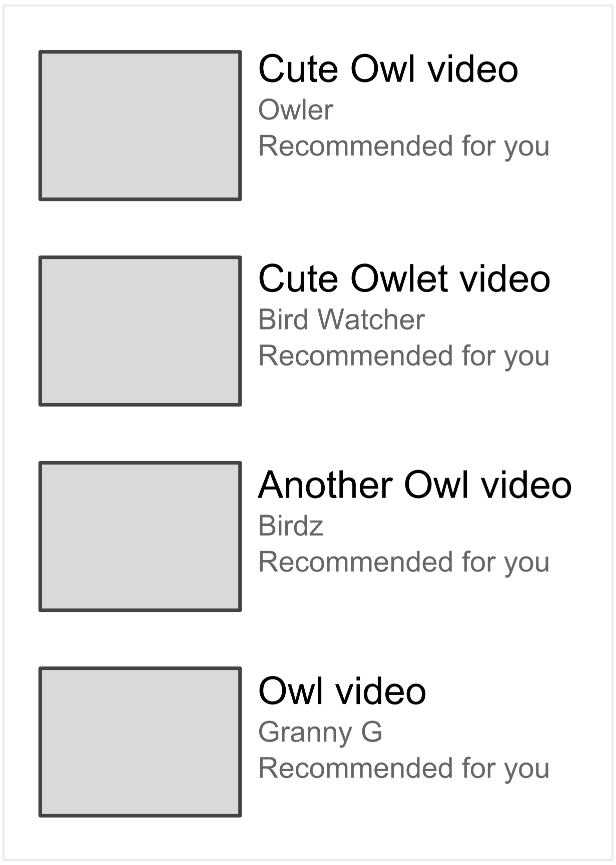
Freshness

Most recommendation systems aim to incorporate the latest usage information, such as current user history and the newest items. Keeping the model fresh helps the model make good recommendations.

Solutions

- Re-run training as often as possible to learn on the latest training data. We recommend warm-starting the training so that the model does not have to re-learn from scratch. Warm-starting can significantly reduce training time. For example, in matrix factorization, warm-start the embeddings for items that were present in the previous instance of the model.

- Create an "average" user to represent new users in matrix factorization models. You don't need the same embedding for each user—you can create clusters of users based on user features.
- Use a DNN such as a softmax model or two-tower model. Since the model takes feature vectors as input, it can be run on a query or item that was not seen during training.
- Add document age as a feature. For example, YouTube can add a video's age or the time of its last viewing as a feature.



Diversity

If the system always recommend items that are "closest" to the query embedding, the candidates tend to be very similar to each other. This lack of diversity can cause a bad or boring user experience. For example, if YouTube just recommends videos very similar to the video the user is currently watching, such as nothing but owl videos (as shown in the illustration), the user will likely lose interest quickly.

Solutions

- Train multiple candidate generators using different sources.
- Train multiple rankers using different objective functions.
- Re-rank items based on genre or other metadata to ensure diversity.

Fairness

Your model should treat all users fairly. Therefore, make sure your model isn't learning unconscious biases from the training data.

Solutions

- Include diverse perspectives in design and development.
- Train ML models on comprehensive data sets. Add auxiliary data when your data is too sparse (for example, when certain categories are under-represented).
- Track metrics (for example, accuracy and absolute error) on each demographic to watch for biases.
- Make separate models for underserved groups.

← Previous
Scoring (/machine-learning/recommendation/dnn/scoring)

Next
Softmax Exercise →
(/machine-learning/recommendation/labs/movie-rec-softmax-programming-exercise)