Ranking

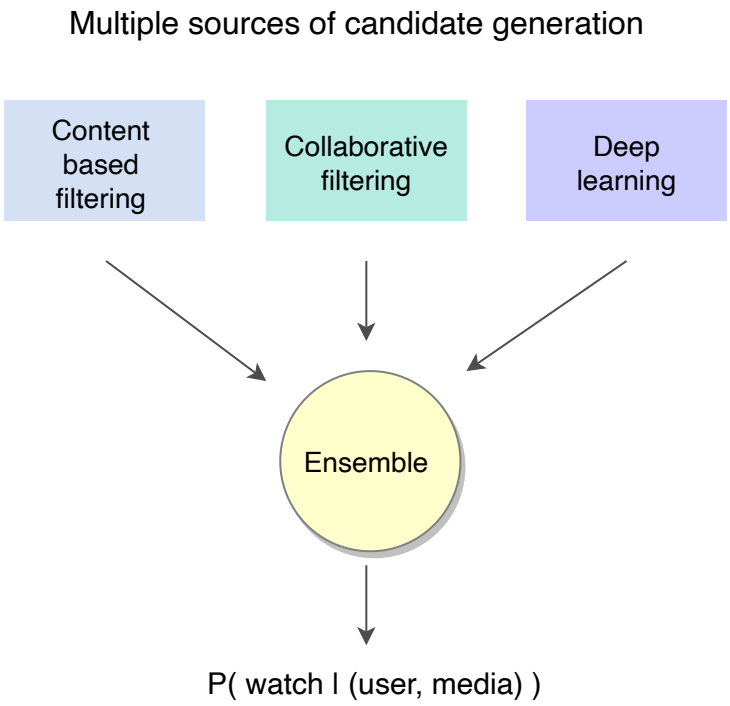
Let's look at modeling options for the recommendation system.

We'll cover the following

- Approach 1: Logistic regression or random forest
- Approach 2: Deep NN with sparse and dense features
- Network structure
- Re-ranking

The ranking model takes the top candidates from multiple sources of candidate generation that we have discussed. Then, an ensemble of all of these candidates is created, and the candidates are ranked with respect to the chance of the user watching that video content.

Here, your goal is to rank the content based on the probability of a user watching a media given a user and a candidate media, i.e., P(watch | (User, Media)).



Ensemble of candidate generation models

There are a few ways in which you can try to predict the probability of *watch*. It would make sense to first try a simplistic approach to see how far you can go and then apply complex modelling approaches to further optimize the system.

First, we will discuss some approaches using logistic regression or tree ensemble methods and then a deep learning model with dense and sparse features.

Deep learning should be able to learn through sparse features and outperform simplistic approach. Still, as we discussed in earlier problems, generalization with a deep NN model needs an order of magnitude more data (order of 100 of millions of training examples) and training capacity/time (100x more CPU cycles).

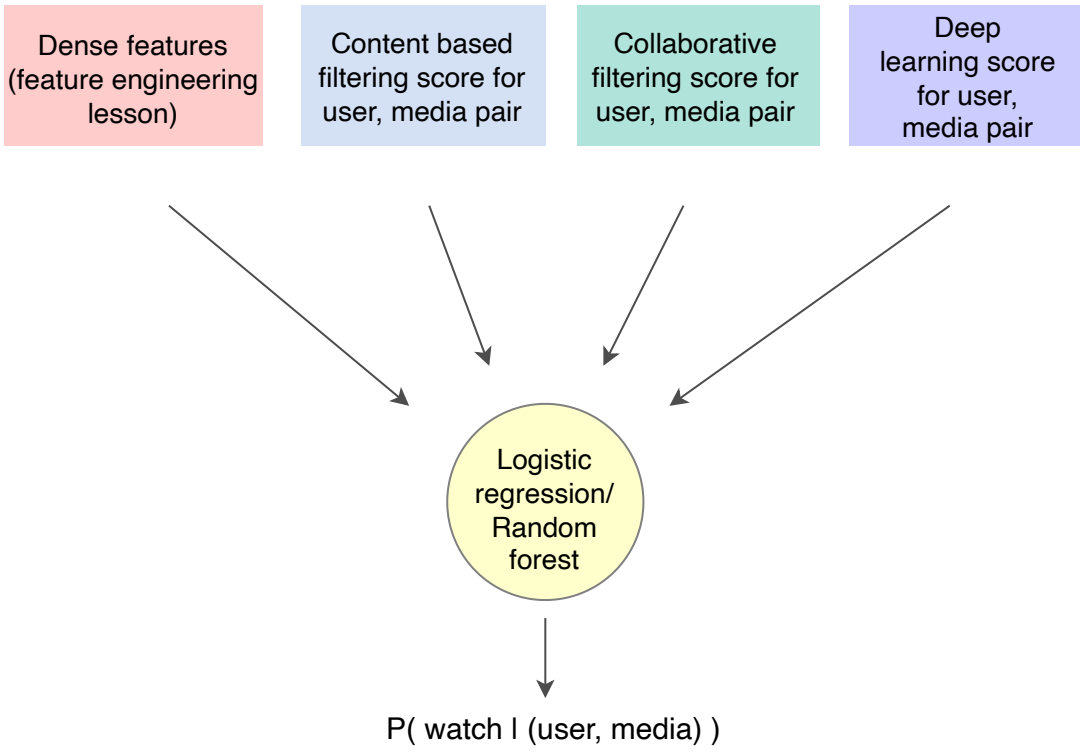
Approach 1: Logistic regression or

random forest#

There are multiple reasons that training a simplistic model might be the way to go. They are as follows:

- Training data is limited
- You have limited training and model evaluation capacity
- You want model explainability to really understand how the ML model is making its decision and show that to the end-user
- You require an initial baseline to see how far you can go in reducing our test set loss before you try more complex approaches

Along with other important features we discussed in the feature engineering section, output scores from different candidate selection algorithms are also a fairly important input consumed by the ranking models.



Approach 1

It is critical to minimize the test error and choose hyperparameters for training and regularization that gives us the best result on the test data.

Approach 2: Deep NN with sparse and dense features#

Another way to model this problem is to set up a deep NN. Some of the factors that were discussed in Approach 1 are now key requirements for training this deep NN model. They are as follows:

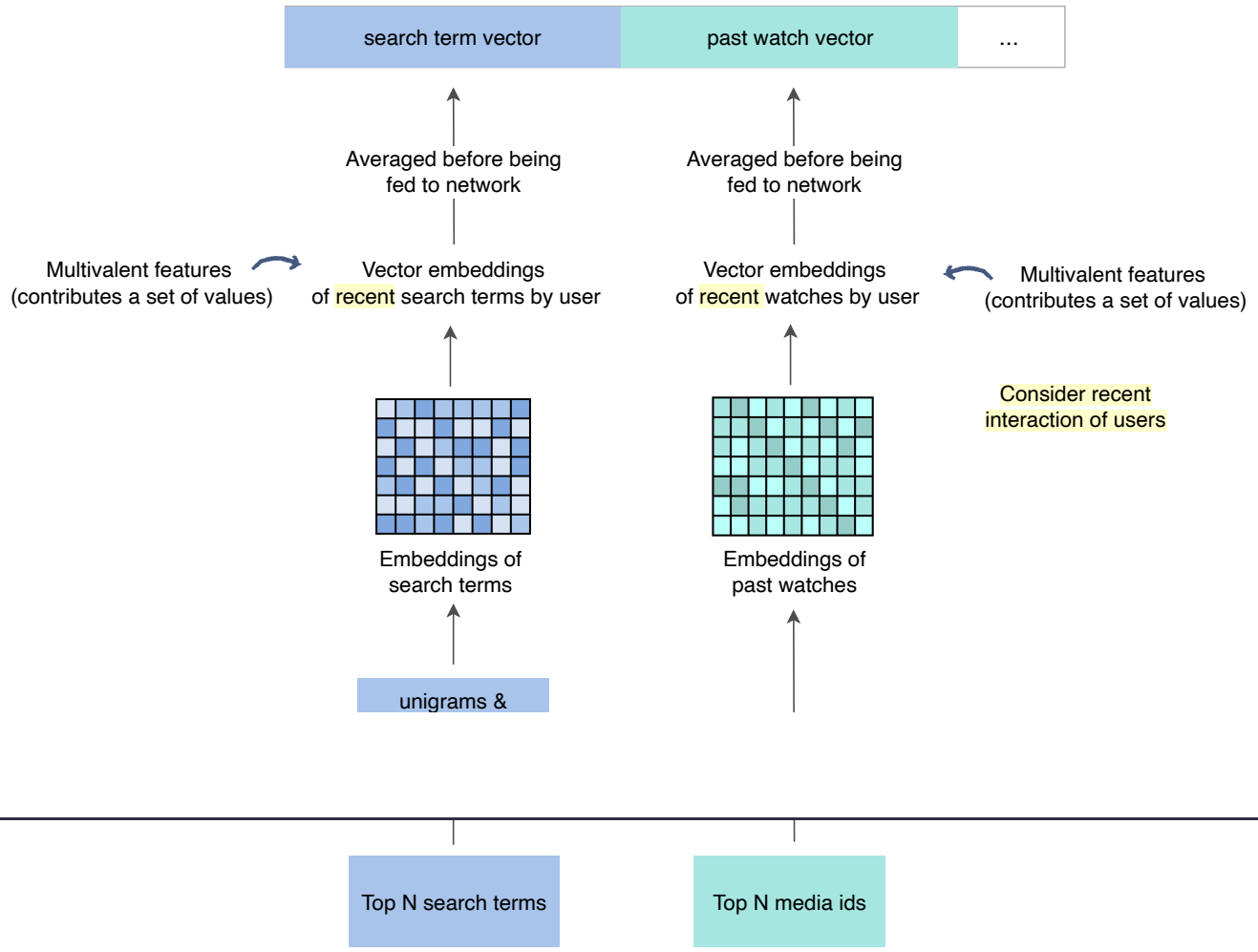
- Hundreds of millions of training examples should be available
- Having the capacity to evaluate these models in terms of capacity and model interpretability is not that critical.

It's important to call out that given the scale of Netflix, fulfilling the above requirements should not be a problem. Utilizing large scale data will definitely be able to outperform simplistic approaches discussed earlier.

Since the idea is that you want to predict whether the user will watch the media or not, you train a deep NN with sparse and dense features for this learning task. Two extremely powerful sparse features fed into such a network can be videos that the user has previously watched and the user's search terms. For these sparse features, you can set up the network to also learn media and search term embeddings as part of the learning task. These specialized embeddings for historical watches and search terms can be very powerful in predicting the *next watch idea* for a user. They will allow the model to personalize the recommendation ranking based on the user's recent interaction with media content on the platform.

An important aspect here is that both search terms and historical watched content are list-wise features. You need to think about how to feed them in the network given that the size of the layers is fixed. You can use an approach similar to pooling layers in CNN (convolution neural networks) and simply average the historical watch id and search text term embeddings

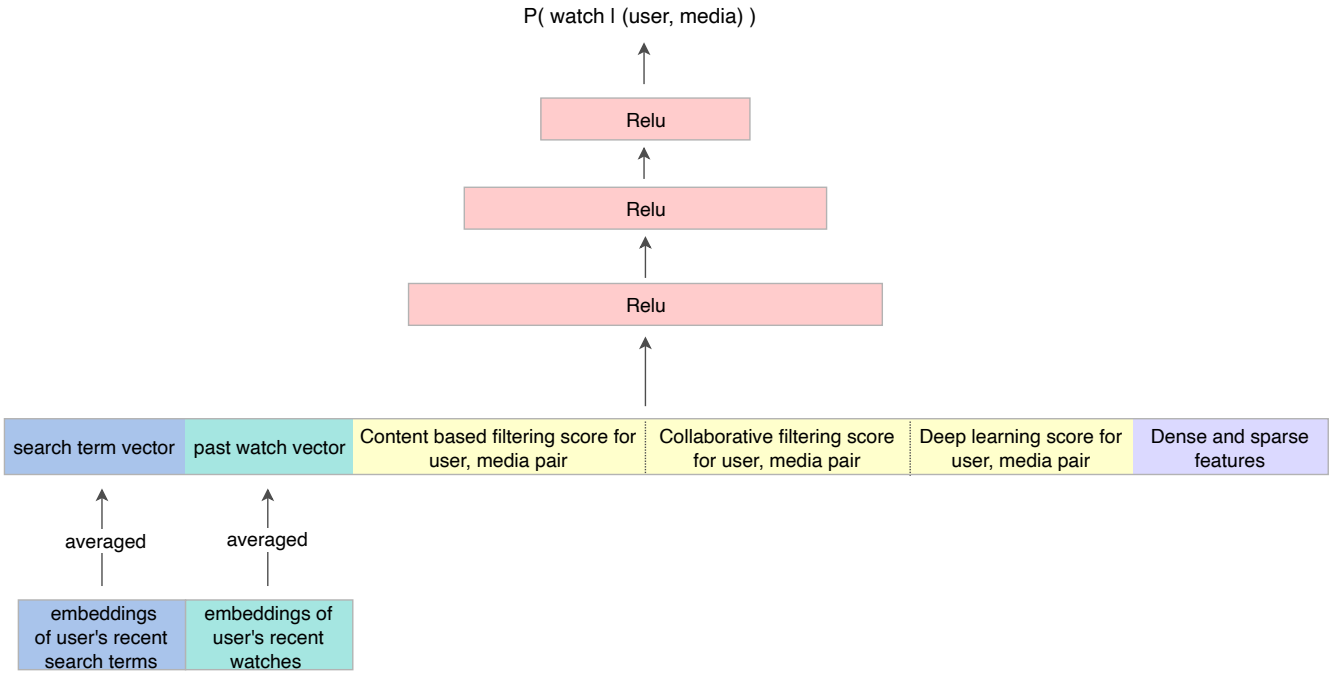
before feeding it into the network.



Generating features from historical watches and search terms

Click to expand: out of vocabulary searches / watches

One way to set up the network is shown below utilizing these sparse data embeddings and feeding them into the Neural network.



Training a deep neural network with embedding and other dense features

As shown above, you can set it up as multiple RELU units layered on top of the embeddings that you learned along with other features. The top layer will predict whether the media will be watched or not using logistic loss.

Network structure#

How many layers should you setup? How many activation units should be used in each layer? The best answer to these questions is that you should start with 2-3 hidden layers with a RELU based activation unit and then play around with the numbers to see how this helps us reduce the test error. Generally, adding more layers and units helps initially, but its usefulness tapers off quickly. The computation and time cost would be higher relative to the drop in error rate.

Re-ranking#

The top ten recommendations on the user's page are of great importance.

After your system has given the watch probabilities and you ra t

results accordingly, you may re-rank the results.

Re-ranking is done for various reasons, such as bringing diversity to the recommendations. Consider a scenario where all the top ten recommended movies are comedy. You might decide to keep only two of each genre in the top ten recommendations. This way, you would have five different genres for the user in the top recommendations.

If you are also considering past watches for the media recommendations, then re-ranking can help you. It prevents the recommendation list from being overwhelmed by previous watches by moving some previously watched media down the list of recommendations.

Back

Training Data Generation

Next

Problem Statement

Mark as Completed

Report an Issue