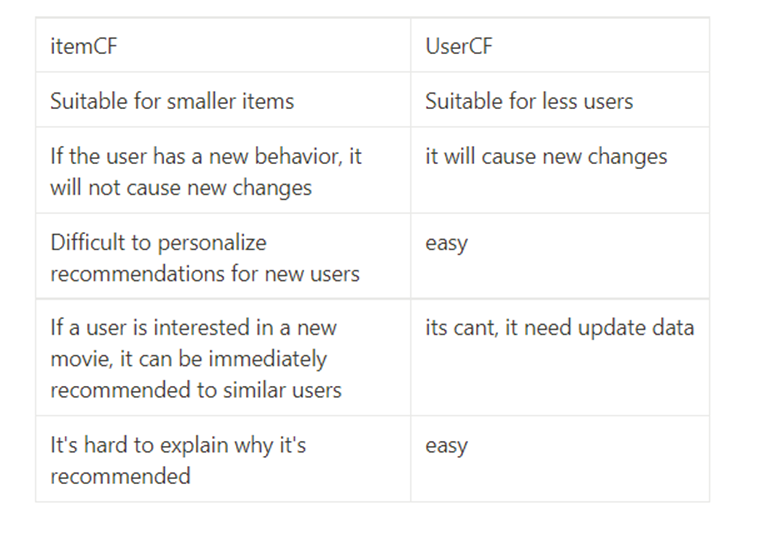
Pengxu Zhu - *Recommendation algorithms*

●       Initial design and model

The very first models used collaborative filtering, of which there are user-CF and item-CF respectively. item-CF recommendation mechanism is an improved strategy of Amazon on a user-based mechanism, where the number of items is much smaller than the number of users in most of the web, and the number of items and similarity are more stable. The item-based mechanism is also a little better than the user-based mechanism in real time. Regarding user-cf, it is generally used when the number of items is greater than the number of users. The 2 types of collaborative filtering algorithms have obvious disadvantages and advantages for both types of recommendations. The initial plan to choose the model was because I found the dataset because I could use collaborative filtering to calculate the similarity between user and item (Pearson's correlation coefficient or cosine similarity) and the algorithm could recommend similar movies if the user searched for a new movie.



Improved model CNN

By looking at the types of fields in the dataset, we found that some are category fields, the usual processing is to convert these fields into one hot code, but fields like UserID, MovieID become very sparse and the dimensionality of the input expands dramatically, which we don't want to see, after all, my little laptop is not like the big factories that can handle hundreds of millions of dimensions of input at any one time :)So in pre-processing the data these fields were converted to numbers, which we used as an index for the embedding matrix, using the embedding layer in the first layer of the network, with dimensions (N, 32) and (N, 16).The processing of film types takes one more step, sometimes a film has more than one film type, so that the index from the embedding matrix is an (n, 32) matrix, because there are more than one type well, we have to sum this matrix into a (1, 32) vector. The movie title is a special case, as instead of using a recurrent neural network, a text convolutional network is used, as will be explained below. After indexing the features from the embedding layer, each feature is passed into the fully-connected layer and the output is passed again into the fully-connected layer, resulting in two feature vectors of (1, 200) user features and movie features respectively. Our aim is to train the user features and movie features to be used in the implementation of the recommendation function. Once these two features are obtained, it is possible to choose any way to fit the ratings. I have used two approaches, one is to do a vector multiplication of the two features as drawn above and regress the results against the true ratings, using MSE to optimize the loss. Because essentially this is a regression problem, the other way is to take the two features as input, pass them into the fully connected layer again, output a value and regress the output value on the true score, using MSE optimized loss.

**CNN model performance and idea**

1.Recommended films in the same genre

2.recommending movies that users like

3. What other movies have people who have seen this movie watched (or liked)

1 Idea is: By calculating the movie similarity, select top\_k similar movies, and then add some random movies

2 Idea is Compute ratings for all movies using the user feature vector and movie feature matrix. Take the top\_k ones with the highest score, and also add some random movies.

3 Idea is Select top\_k people who like this movie to get features, top\_k people score all movies, push the movie with the highest score to the user, and add some random movies

