podtapes | Personalization Service Implementation

Regarding the personalization service, the best algorithm to use is hands down 'Collaborative Filtering'. However, due to cold start and lack of both users and Items, it's may be a better idea to start using something simpler like a linear equation that relies on simple parameters instead of trying to figure out features out of nothing. Thus meaning that we're going to divide the personalization in at least two steps.

Step 1: Linear approach to address lack of Users and Items.

A good start may be to consider the number of views to be directly proportional to the number of likes against dislikes. This is good because it's based on the general trend and so if something is liked by most users, there is a high probability that you're going to like it too by the statistical 'Law of Big Numbers'.
We also want to provide personalzation based on user taste. In-fact, even though Marvel's Cinematic Universe movies are nowdays considered by a very large scale of people some of the best movies to watch, if you don't like either the Fantasy or Action genere, high chances you won't like Marvel's movies or superhero movies in general. We can add to the equation the sum of the user's views directly proportional to likes against dislikes on the selected movie genre/s. Here may become handy the use of tags to better categorize movies.
For instance, take a movie like Ready Player One. It certainly is an action and sci-fi movie, but it doesn't relate to movies like Star Wars or Inception. It certainly relates though to movies like Alita: Battle Angel or Dredd. That is because those movies have in common being in the Cyberpunk style, being Futuristic and set in a Distopian reality.
We can improve our 'User Profiling' by adding those sort of tags to our items and whenever a user interact with them, by those tags we can track easily views, likes and dislikes by tag, allowing us to better recommend movies. To have a better control on our Linear Model, we can use weights in a Deep Learning fashion

$$\hat{y} = w_1\left(views \cdot \frac{likes}{dislikes}\right) + w_2 \sum_{u\_cat=0}^{m\_cats}\left(views_{u\_cat} \cdot \frac{likes_{u\_cat}}{dislikes_{u\_cat}}\right) + w_3 \cdot \sum_{u\_tag=0}^{m\_tags}\left(views_{u\_tag} \cdot \frac{likes_{u\_tag}}{dislikes_{u\_tag}}\right)$$

The problem with this equation is that we have very little control on the value that comes out of each piece. This could cause both precision and performance issues, so we're going to use a Deep Learning trick. We're going to enclose them in logarithm functions. For simplicity let's assume:

$$a = views \cdot \frac{likes}{dislikes} \qquad b = \sum_{u\_cat=0}^{m\_cats}\left(views_{u\_cat} \cdot \frac{likes_{u\_cat}}{dislikes_{u\_cat}}\right) \qquad c = \sum_{u\_tag=0}^{m\_tags}\left(views_{u\_tag} \cdot \frac{likes_{u\_tag}}{dislikes_{u\_tag}}\right)$$

Then, the more efficient equation becomes

$$\hat{y} = w_1 \ln(a) + w_2 \ln(b) + w_3 \ln(c)$$

NOTE: Be careful when 'likes' or 'dislikes' is 0.

It may also be good a good idea to use an additional weight to each 'views' parameter.

Step 2: Switch to the more precise Collaborative Filtering.

Engineered by Ruben Viscomi