Creator-Assisted Contextual Ranking

Archisman Biswas

August 2023

1 Introduction

Say we have a creator who has uploaded many videos on YouTube. What he just does is that, he submits a ranking of the videos, according to his preference. So, he would like the higher ranked videos to be watched more than the lower ranked ones. But, we cannot really present the same ranking while recommending to a particular user. This is a contextual problem, since it depends on the present context with which we are dealing with. That is, a particular user will have a particular choice at a particular time. Hence, we have to take care of that and recommend a ranking which balances the user as well as creator satisfaction.
We have a pool of videos which a particular creator has uploaded and his preference sorted accordingly. We also have a particular user for whom we shall present a ranking of recommendations.

2 Outline

The idea is that, we have first shortlisted the top N options (arms). This is because, a user has to be provided with a fixed number of recommendations. Then, we use a Disjoint Payoff Model to calculate the expected scores of a particular user. The options shall be ranked accordingly then, which is assumed to be the ideal ranking for the user. At the same time, the creator will also have his preference, which is another ranking. So, we define a measure of regret, and then take a weighted mean of the regret of both the user and the creator. This is the regret which has to be minimized as we reach the optimal ranking.
Currently, one fundamental assumption is that, this is a stationary environment.

2.1 Shortlisting the Options

Choose a number N. Shortlist the top N options using info from the log file which is available. The log file stores the Past-Student-Behaviours (PSBs). Those are data from the learning paths of previous students.

For example, if someone has the list of actions as:

$$a \to b \to a \to b \to d$$

then it will store the frequency of each transition in the matrix. That is,

$$|a \to b| = 2,\ |b \to a| = 1$$

	b_1	b_2	b_3	$\dots$	$\dots$	b_k
a_1	f_{11}	f_{12}	f_{13}	$\dots$	$\dots$	f_{1k}
a_1	f_{21}	f_{22}	f_{23}	$\dots$	$\dots$	f_{2k}
a_3	f_{31}	.				
.	.	.				
.	.	.				
.	.	.				

Table 1: Correlation Frequency Matrix

These data will be stored in a matrix. The transitions occur from a_i to b_j, where $1 \leq j \leq k$ and $i \in N$, where b_j s are the arms (videos) of the particular creator.

Definition : *Correlation Measurement*
Let L be a log file over A, where A be a set of actions and $a, b \in A$. The correlation of action a and b is defined by:

$$\text{cor}(a, b) = \frac{|a \to b|}{N(a)} \tag{1}$$

Here $N(a)$ is the number of times action a appear as a prefix in a transition
After correlation analysis, the Current Student State (CSS) is used as the contextual information to predict the next action. We determine the top N highest value of correlation given the current state, where N is the number of desired candidate actions.
The top-N actions, $top(A_t)$, is a set of actions that are generated as the candidates to be recommended to the target user at time step t where $(a_1, a_2, \dots, a_N) \in A$, but not in the previously taken actions of the target student, $top(A_t) \neq (a_{t=1}, a_{t=2}, \dots, a_{t-1})$, and N is the number of desired top actions for recommended $top(A_t)$ to the target student at time t. The agent does not take into account the duplicated actions that have already been taken by the target student.
The agent only determines the learning path and assumes that the target student already achieves the learning objectives from the previous LOs.

So after the end of this step, we have our top-N arms (videos), which can be recommended.

2.2 Computing the User ranking

Here, we shall compute the expected payoffs and rank the arms accordingly. This ranking would be the ideal ranking for the user.
Given a user feature vector x of size n, N feature vectors of each arm y_i $(1 \leq i \leq N)$ of size m. We also have a preference matrix W of size $n \times m$.
W_{ij} stores the preference of the ith user feature towards the jth arm feature. The expected reward r_i is calculated as:

$$E[r_i | x, W, y_i] = x^T W y_i, \text{ for } 1 \leq i \leq N \tag{2}$$

2.3 Optimizing the ranking

Let p be a permutation (or a ranking) of n options, with the first element having the highest priority. Now consider another permutation p'. Define f and $R(p, p')$ (regret of p' w.r.t. p) as:

$$f(i, p, p') := \begin{cases} 0, & \text{if, } pos(i \ in \ p) \leq pos(i \ in \ p') \\ pos(i \ in \ p) - pos(i \ in \ p'), & \text{Otherwise} \end{cases} \tag{3}$$

$$R(p,\ p') := \sum_{i=1}^{n} ((n + 1 - i) \cdot f(p[i],\ p,\ p')) \tag{4}$$

1-based indexing have been used above.

Now, in the final step of the problem, we shall have to minimize a weighted sum of the regrets of both the user and the creator.
Given, the user ranking obtained using 2.2 (p), the ranking submitted by the creator (q), we can can choose two weights w_1, w_2 and compute:

$$\pi = \arg\min_x \ (w_1 \cdot R(p,\ x) + w_2 \cdot R(q,\ x)) \tag{5}$$

2.4 Conclusion

π is the ranking which we return.