Slate Recommendation Systems: A Reinforcement Learning Problem

Introduction to Problem Statement:

The PS here deals with building a Recommender System that optimally chooses and places C=9 items on a grid out of a total dataset of T=350 items. This should consider some basic features of the users, expected to be single-use, as well as interaction and correlation amongst the chosen items. We also wish to incorporate features such as seasonality and trends.

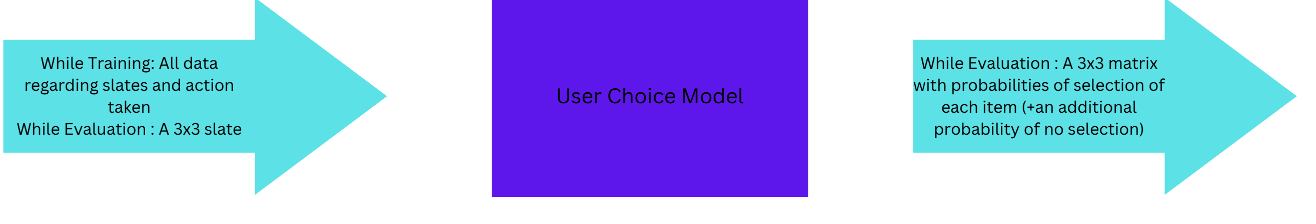
What a Multi Arm Bandit Approach would look like:

A simple bandit approach would treat every slate as one item, i.e. one action set. This would create 350P9, i.e. 7*10^22 items. Finding enough data to exploit the best actions from this set would be near impossible. This makes it intractable to use on such a dataset. This leads to the need to look at alternate methods of solving this problem

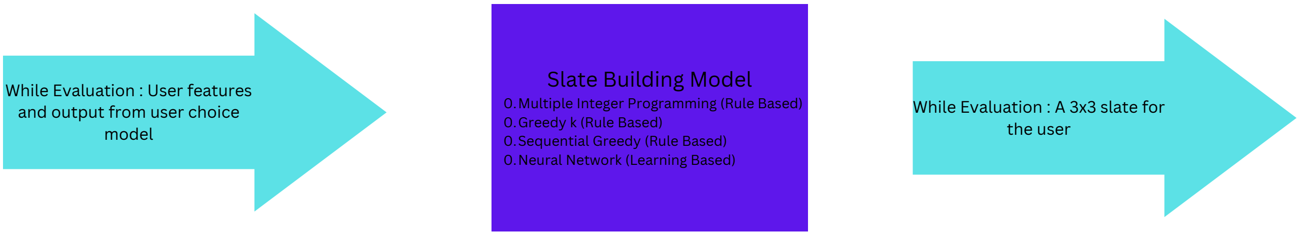
SlateQ:

A recent approach to the slate recommendation problem is an MDP model called SlateQ. SlateQ looks at building a two-part solution to this problem as follows:

1. A User Choice Model:
    a. Input:
       Interaction Data from users
    b. Output:
       The user choice model outputs a relative probability of selecting each item for every grid. For example, this probability would be 1/10 for each item before training. (The 10th item being a null item)
       After that, it would then train to calculate relative probability.
    c. How:
       SlateQ does not explain this model. Here we are looking to ideate and create an alternate model.
    d. Diagram:



2. A Slate Building Model:
    a. Input:
    The relative probabilities of selection, and the rewards associated with each item
    b. Output:
    The best slate out of the 7*10^22 items.
    c. How:
    SlateQ uses mixed integer programming to solve a combinatorial problem in linear time.
    d. Diagram:



Approach to the problem:

This problem can be approached in two stages:

1. Stage 1: Looking for alternate slate recommendation solutions
    a. This includes looking at SlateQ and exploring alternatives such as off-policy evaluation, SEO models, and CMAB models.
2. Stage 2: Proceeding with SlateQ and looking at possible User Choice models
    a. If we proceed with SlateQ, it still leaves uncertainty about how the user choice will be modelled.
    b. This could include looking at Multinomial Logit models, CTR models or building a model from scratch.

Stages of solution:

1. Literature Review:
    a. This is the current step of the process that I am at, which involves reading both about possible alternate models as well as possible options for user choice and understanding their pros and cons
    b. I have reviewed the papers listed below and am still trying to find evidence that interaction data is modelled at any stage of this process.
    c. I have found papers that take into account the positioning of the items
2. Database selection:
    a. We worked initially n setting up the Finn.no database, however, since it could not be fixed up, we moved to the Google RecSim database.
    b. The Recsim database has now been installed and set up.
3. Proposed Alternative Models:
    a. Proposed Model Type 1

        i. The first proposal is to consider 350C2 interactions. This would involve learning all 350C2 interactions, then choosing the k items with the highest correlation magnitude will be factored in, while creating the final slate.
        ii. This reduces complexity by eliminating all interaction terms above 350C3, for which we need significantly more data. It also brings us down from 350C2 to 350*k interaction terms to consider. k could be 1,2,5,10, depending on the data.
        iii. The alternative could be to initially reduce ur action space from 350 to around 50 based on the popularity of the singular items (i.e level 1 terms) itself
    b. Proposed Model Type 2
        i. We could look at non-linearising the action space based on suggestions from Prof.Ravi. This could include building the following.
            1. A neural network-based Contextual Bandit
            2. Relational Boosted Bandits
            3. GP-UCB (Gaussian Process optimizer with upper confidence bounds)

References:

1. SlateQ:
    a. Paper: https://www.ijcai.org/proceedings/2019/0360.pdf
    b. Paper 2: https://arxiv.org/pdf/1905.12767.pdf
    c. Implementation Code (Raylib): https://docs.ray.io/en/latest/rllib/rllib-algorithms.html
    d. Source Code (Raylib): https://docs.ray.io/en/latest/_modules/ray/rllib/algorithms/slateq/slateq.html#SlateQConfig
    e. Implementations: https://github.com/collinprather/SlateQ/blob/master/notebooks
    f. Detailed Paper: https://www.arxiv-vanity.com/papers/1905.12767/#S5
2. RecSim platform
    a. Paper: https://arxiv.org/pdf/1909.04847.pdf
    b. Implementation: https://github.com/google-research/ recsim/ (Pasted as text due to format issues)
3. Off-policy evaluation: https://proceedings.neurips.cc/paper/2017/file/5352696a9ca3397beb79f116f3a33991-Paper.pdf
4. Combinatorial Multi Arm Bandit: http://proceedings.mlr.press/v28/chen13a.pdf
5. Clickthrough Data Optimizer: https://www.cs.cornell.edu/people/tj/publications/joachims_02c.pdf
6. MNL Model: https://www.public.asu.edu/~hongminl/mnl_attributes_price%20v5t_final.pdf
7. Cascading Behaviours:
    a. Cascading Bandits: https://arxiv.org/pdf/1502.02763.pdf
    b. Multiple Cascading Behaviours: https://arxiv.org/pdf/1809.08161.pdf
8. General Survey - RL in Recommender Systems: https://arxiv.org/pdf/2109.10665.pdf
9. Gaussian Process Optimization in the Bandit Set - https://arxiv.org/pdf/0912.3995.pdf
10. Neural Bandits
    a. Neural Contextual Bandits: https://arxiv.org/pdf/2107.03144.pdf
    b. Neural Contextual Bandits with UCB: https://arxiv.org/pdf/1911.04462.pdf
11. RecSim NG: https://arxiv.org/pdf/2103.08057.pdf
12. Cluster Bandits: https://github.com/google-research/recsim/tree/master/recsim /colab (Pasted as text due to format issues)