OBP Presentation

1) OBP Overview
    a. Focus on offline experimentation https://sites.google.com/cornell.edu/recsys2021tutorial
2) OBP Process(https://raw.githubusercontent.com/st-tech/zr-obp/master/images/overview.png)
    a. Data Management
        i. Datasets
        ii. Bandit Feedback
            1. Dictionary storing logged data
            2. Action_context: Context vectors characterizing actions (i.e., a vector representation or an embedding of each action).
            3. OBP Extension (Slate)
                a. Comparison of bandit feedback

    b. Policy Learning (https://colab.research.google.com/github/st-tech/zr-obp/blob/master/examples/quickstart/opl.ipynb)
        i. Class wrapper for ML model
            1. Example IPWLearner
            2. Off-policy learner based on Inverse Probability Weighting and Supervised Classification.
        ii. Outputs
            1. predictions
            2. action_probabilty distributions(where len_size = 1)
    c. Simulation
        i. Based on online policy mainly
    d. Off-Policy Estimators (OPE)
        i. Policy values (metrics) based on reward system
        ii. Return values within [0,1]
        iii. Estimates the performance of a policy based on log history
3) SCRUFF-D Integration
    a. Similarities
        i. Both reference logged data
            1. Both use feedback logs to improve predictions
        ii. Both use similar data structures
            1. UserID, Lists
    b. Differences
        i. OBP uses reward system
        ii. OBP is based on Machine Learning driven by data and SCRUFF-D is more model based
        iii. Metrics (OPB is rewards based)
    c. Implementation
        i. Partial
            1. OBP as recommender system
            2. Build Algorithm and Choice Mechanism, feedback separately

ii. Full
    1. Modifications (Similar to Slate)
        a. Evaluators/ Metrics
        b. Bandit Feedback
        c. Determine data sources/storage
    2. Rewards Problem
        a. Set all rewards equal to 1
        b. Repurpose the rewards label
            i. Could signal protected items/lists
    3. Fairness Agents
        a. Each Fariness Agent serves as an OBP policy
        b. Allocation Mechanism manages policy usage based on BanditFeedback
            i. Nested actions
    4. List Structure
        a. OBP uses three dimension (context,actions,posion)
            i. SCRUFF-D uses (userID, ItemID, pposition)
        b. Context,ListID could be used in OBP, but we would need to pull list from data source
iii. Proposed SCRUFF-D Data Structures in OBP
    1. User Recommendation List (l)
        a. Machine learning algorithm (e. g. IPWLearner)
            i. Context: User Profile (omega)
            ii. Output:
                1. Actions: itemIDs (v)
                2. Positions: 0… N-1
    2. Allocation History(H)
        a. Bandit feedback collection of agent allocations with time index
    3. Choice History(L)
        a. Bandit feedback collection of user recommendation list (l) with time index
        b. Bandit feedback collection of agent recommendation lists (l_f) with time index
        c. Bandit feedback collection of choice function output list (l_c) with time index
    4. Fairness metric for agent (m_i)
        a. Off-Policy Estimator (OPE)
            i. Context: Allocation History (H)
            ii. Context: Choice History(L)
            iii. Output: rating within [0,1]
    5. Compatibility metric for agent (c_i)
        a. Off-Policy Estimator (OPE)
            i. Context: User Profile (omega)
            ii. Output: rating within [0,1]

6. Fairness Agent Recommendation Function (R)
    a. Machine learning algorithm (e. g. IPWLearner) or hardcoded algorithm
        i. Context: User Profile (omega)
        ii. Context: ItemID (v)
        iii. Output: rating
7. Allocation Mechanism
    a. Machine learning algorithm(s) (e. g. IPWLearner)
        i. Context: Fairness metric evaluations (m_F)
        ii. Context: Compatibility metric evaluations(c_F)
        iii. Actions: Fairness Agents (f)
        iv. Output: Agent allocation (beta) as action_distribution

Bandit_feedback example form SyntheticBanditDataset in synthetic.py

```
>>> bandit_feedback
{
    'n_rounds': 10000,
    'n_actions': 5,
    'context': array([[-0.20470766,  0.47894334, -0.51943872],
            [-0.5557303 ,  1.96578057,  1.39340583],
            [ 0.09290788,  0.28174615,  0.76902257],
            ...,
            [ 0.42468038,  0.48214752, -0.57647866],
            [-0.51595888, -1.58196174, -1.39237837],
            [-0.74213546, -0.93858948,  0.03919589]]),
    'action_context': array([[1, 0, 0, 0, 0],
            [0, 1, 0, 0, 0],
            [0, 0, 1, 0, 0],
            [0, 0, 0, 1, 0],
            [0, 0, 0, 0, 1]]),
    'action': array([4, 2, 0, ..., 0, 0, 3]),
    'position': None,
    'reward': array([1, 0, 1, ..., 1, 1, 1]),
    'expected_reward': array([[0.58447584, 0.42261239, 0.28884131,
0.40610288, 0.59416389],
            [0.13543381, 0.06309101, 0.3696813 , 0.69883145, 0.19717306],
            [0.52369136, 0.30840555, 0.45036116, 0.59873096, 0.4294134 ],
            ...,
            [0.68953133, 0.55893616, 0.34955984, 0.45572919, 0.67187002],
            [0.88247154, 0.76355595, 0.25545932, 0.19939877, 0.78578675],
            [0.67637136, 0.42096732, 0.33178027, 0.36439361, 0.52300522]]),
    'pi_b': array([[[0.27454777],
            [0.16342857],
            [0.12506266],
            [0.13791739],
            [0.22195834]]]),
    'pscore': array([0.28264435, 0.19326617, 0.23079467, ..., 0.28729378,
0.36637549,
            0.13791739])
}
```

Bandit_feedback example form SyntheticSlateBanditDataset in synthetic_slate.py

```
>>> bandit_feedback
{
    'n_rounds': 5,
    'n_unique_action': 10,
    'slate_id': array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4]),
    'context': array([[-0.20470766,  0.47894334, -0.51943872, -0.5557303 ,
1.96578057],
        [ 1.39340583,  0.09290788,  0.28174615,  0.76902257,  1.24643474],
        [ 1.00718936, -1.29622111,  0.27499163,  0.22891288,  1.35291684],
        [ 0.88642934, -2.00163731, -0.37184254,  1.66902531, -0.43856974],
        [-0.53974145,  0.47698501,  3.24894392, -1.02122752, -0.5770873 ]]),
    'action_context': array([[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]]),
    'action': array([8, 6, 5, 4, 7, 0, 1, 3, 5, 4, 6, 1, 4, 1, 7]),
    'position': array([0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2]),
    'reward': array([1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0]),
    'expected_reward_factual': array([0.5       , 0.73105858, 0.5       ,
0.88079708, 0.88079708,
        0.88079708, 0.5       , 0.73105858, 0.5       , 0.5       ,
        0.26894142, 0.5       , 0.73105858, 0.73105858, 0.5       ]),
    'pscore_cascade': array([0.05982646, 0.00895036, 0.00127176, 0.10339675,
0.00625482,
        0.00072447, 0.14110696, 0.01868618, 0.00284884, 0.10339675,
        0.01622041, 0.00302774, 0.10339675, 0.01627253, 0.00116824]),
    'pscore': array([0.00127176, 0.00127176, 0.00127176, 0.00072447,
0.00072447,
        0.00072447, 0.00284884, 0.00284884, 0.00284884, 0.00302774,
        0.00302774, 0.00302774, 0.00116824, 0.00116824, 0.00116824]),
    'pscore_item_position': array([0.19068462, 0.40385939, 0.33855573,
0.31231088, 0.40385939,
        0.2969341 , 0.40489767, 0.31220474, 0.3388982 , 0.31231088,
        0.33855573, 0.40489767, 0.31231088, 0.40489767, 0.33855573])
}
```

Position 1

	item_1	item_2	item_3	item_4
user_1	0	1	0	0
user_2	0	0	1	0
user_3	1	0	0	0
user_4	0	0	0	1
user_5	1	0	0	0

Position 2

	item_1	item_2	item_3	item_4
user_1	0	0	1	0
user_2	1	0	0	0
user_3	0	1	0	0
user_4	1	0	0	0
user_5	0	0	0	1

Position 3

	item_1	item_2	item_3	item_4
user_1	1	0	0	0
user_2	0	0	0	1
user_3	0	0	0	1
user_4	0	0	1	0
user_5	0	1	0	0

```
Item Positions for user 1:

array([[0, 1, 0, 0],
       [0, 0, 1, 0],
       [1, 0, 0, 0]])

[1, 2, 0]
```

Out[95]:
```
array(['item_2', 'item_3', 'item_1'], dtype='<U6')
```