test_notebook

March 27, 2023

```
[ ]: import pandas as pd
     import matplotlib.pyplot as plt
     from matplotlib import style
     import numpy as np
     import pickle
     import os
     from sklearn.decomposition import NMF, PCA
     from sklearn.cluster import KMeans
     from importlib import reload

     import sys
     sys.path.insert(1, '/Users/madisonthantu/Desktop/DREAM/t-recs')
     from trecs.metrics import MSEMeasurement, InteractionSpread, InteractionSpread,␣
      ↪InteractionSimilarity, RecSimilarity, RMSEMeasurement, InteractionMeasurement
     from trecs.components import Users
     import trecs.matrix_ops as mo
     import seaborn as sns

     sys.path.insert(1, '..')
     import src.globals as globals
     from wrapper.models.bubble import BubbleBurster
     from src.utils import *
     from src.plotting import plot_measurements
     from src.scoring_functions import cosine_sim, entropy, content_fairness,␣
      ↪cosine_sim2
     from wrapper.metrics.evaluation_metrics import *
     random_state = np.random.seed(42)
     plt.style.use("seaborn")

     # import warnings filter
     from warnings import simplefilter
     # ignore all future warnings
     simplefilter(action='ignore', category=FutureWarning)

     globals.initialize()
```

```
/var/folders/sm/hcy50x855gvf2b1qwkjstnvh0000gn/T/ipykernel_66029/1535014153.py:2
6: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are
```

```
deprecated since 3.6, as they no longer correspond to the styles shipped by
seaborn. However, they will remain available as 'seaborn-v0_8-<style>'.
Alternatively, directly use the seaborn API instead.
  plt.style.use("seaborn")
```

[ ]:

```python
n_attrs=20
max_iter=1000
n_clusters=10
```

[ ]:

```python
score_fn = 'cosine_sim'#'cosine_sim2' #'top_k_reranking'#'entropy' #'entropy' #'content_fairness'
probabilistic = False
globals.ALPHA = 0.1 #10.0 #0.01 # 0.1
alpha = globals.ALPHA

# User parameters
drift = 0.05
attention_exp=-0.8
retrain = True

if retrain:
    str_retrain = "retrainTrue"
else:
    str_retrain = "retrainFalse"

experiment_name = 'supplementary'
this_experiment = f"{score_fn}_{alpha}_{str_retrain}"
```

[ ]:

```python
binary_ratings_matrix = load_and_process_movielens(file_path='/Users/madisonthantu/Desktop/DREAM/data/ml-100k/u.data')

# Get user and item representations using NMF
user_representation, item_representation = create_embeddings(binary_ratings_matrix, n_attrs=n_attrs, max_iter=max_iter)

# Define topic clusters using NMF
item_cluster_ids, item_cluster_centers = get_clusters(item_representation.T, name='item', n_clusters=n_clusters, n_attrs=n_attrs, max_iter=max_iter)
user_cluster_ids, user_cluster_centers = get_clusters(user_representation, name='user', n_clusters=n_clusters, n_attrs=n_attrs, max_iter=max_iter)

num_users = user_representation.shape[0] #len(user_representation)
num_items = item_representation.shape[1] #len(item_representation)
print(f'Number of items: {num_items}')
print(f'Number of users: {num_users}')

users = Users(actual_user_profiles=user_representation,
              repeat_interactions=False,
              drift=drift,
              attention_exp=attention_exp)
```

```
Loaded embeddings.
Calculating clusters…
Calculated clusters.
Calculating clusters…
Calculated clusters.
Number of items: 1682
Number of users: 943
```

[ ]:

```
import math

user_item_cluster_mapping = user_topic_mapping(user_representation,␣
 ↪item_cluster_centers) # TODO: Remove?
# Create user_pairs by pairing users only with others that are not in the same␣
 ↪cluster
inter_cluster_user_pairs, intra_cluster_user_pairs =␣
 ↪create_cluster_user_pairs(user_item_cluster_mapping)

assert(len(inter_cluster_user_pairs) + len(intra_cluster_user_pairs) == (math.
 ↪factorial(num_users) / (math.factorial(2)*math.factorial(num_users-2)))),␣
 ↪"Bug with creating user pairs"
```

[ ]:

```
measurements = [
    InteractionMeasurement(),
    MSEMeasurement(),
    InteractionSimilarity(pairs=inter_cluster_user_pairs,␣
 ↪name='inter_cluster_interaction_similarity'),
    InteractionSimilarity(pairs=intra_cluster_user_pairs,␣
 ↪name='intra_cluster_interaction_similarity'),
    IntraClusterCosineSim(mapping=user_item_cluster_mapping,␣
 ↪n_clusters=n_clusters, name='intra_user_to_topic_cluster_cos_sim'),
    IntraClusterCosineSim(mapping=user_cluster_ids, n_clusters=n_clusters,␣
 ↪name='intra_user_cluster_cos_sim'),
    MeanDistanceFromCentroid(user_cluster_ids, user_cluster_centers,␣
 ↪name='user_cluster_avg_distance_from_centroid'),
]
```

[ ]:

```
# Model
config = {
    'actual_user_representation': users,
    'actual_item_representation': item_representation,
    'item_topics': item_cluster_ids,
    'num_attributes': n_attrs,
    'num_items_per_iter': 10,
    'seed': 42,
    'record_base_state': True,
}

model_name='myopic'
requires_alpha = False

if score_fn:
    if score_fn == 'cosine_sim2':
        config['score_fn'] = cosine_sim2
        requires_alpha = True
    elif score_fn == 'cosine_sim':
        config['score_fn'] = cosine_sim
        requires_alpha = True
    elif score_fn == 'entropy':
        config['score_fn'] = entropy
        requires_alpha = True
    elif score_fn == 'content_fairness':
        config['score_fn'] = content_fairness
    else:
        raise Exception('Given score function does not exist.')
    model_name = score_fn

if probabilistic:
    config['probabilistic_recommendations'] = True
    model_name += '_prob'
```

[ ]:

```
model = BubbleBurster(**config)

model.add_metrics(*measurements)
```

[ ]:

```
# Fair Model
train_timesteps=5
model.startup_and_train(timesteps=train_timesteps)
```

```
100%|          | 5/5 [00:03<00:00,  1.35it/s]
```

[ ]:

```
run_timesteps=50
model.run(timesteps=run_timesteps)
```

```
100%|          | 50/50 [03:24<00:00,  4.09s/it]
```

[ ]:

```
def create_measurements_df(model, model_name, train_timesteps, file_path):
    measurements = model.get_measurements()
    df = pd.DataFrame(measurements)
    df['state'] = 'train' # makes it easier to later understand which part was␣
 ↪training
    df.loc[df['timesteps'] > train_timesteps, 'state'] = 'run'
    df['model'] = model_name

    return df
```

[ ]:

```
import src
reload(src.utils)
from src.utils import *

if retrain:
    # Determine file name based on parameter values
    parameters =␣
 ↪f'_{train_timesteps}trainTimesteps_{run_timesteps}runTimesteps_{n_attrs}nAttrs_{n_clusters}
    if requires_alpha:
        parameters += f'_{alpha}Lambda'
    # Save measurements
    measurements_dir = f'artefacts/{experiment_name}/measurements/'
    file_prefix = f'{model_name}_measurements'
    measurements_path = measurements_dir + file_prefix + parameters + '.csv'
    # np.set_printoptions(threshold=sys.maxsize)
    measurements_df = create_measurements_df(model, model_name,␣
 ↪train_timesteps, measurements_path)
    # measurements_df['interaction_histogram'] =␣
 ↪measurements_df['interaction_histogram'].tolist()
elif not retrain:
    # Determine file name based on parameter values
    parameters =␣
 ↪f'_{train_timesteps}trainTimesteps_{run_timesteps}runTimesteps_{n_attrs}nAttrs_{n_clusters}
    if requires_alpha:
        parameters += f'_{alpha}Lambda'
    # Save measurements
    measurements_dir = f'artefacts/no_train_between_runs/{experiment_name}/
 ↪measurements/'
    file_prefix = f'{model_name}_measurements'
    measurements_path = measurements_dir + file_prefix + parameters + '.csv'
    # np.set_printoptions(threshold=sys.maxsize)
    measurements_df = create_measurements_df(model, model_name,␣
 ↪train_timesteps, measurements_path)
else:
    assert(0), "ERROR"
```

[ ]:

```
# Create df for parameters
numeric_cols = ['trainTimesteps', 'runTimesteps', 'nAttrs', 'nClusters',␣
 ↪'Lambda']
columns = ['model_name'] + numeric_cols

data = [[model_name, train_timesteps, run_timesteps, n_attrs, n_clusters, None]]
```

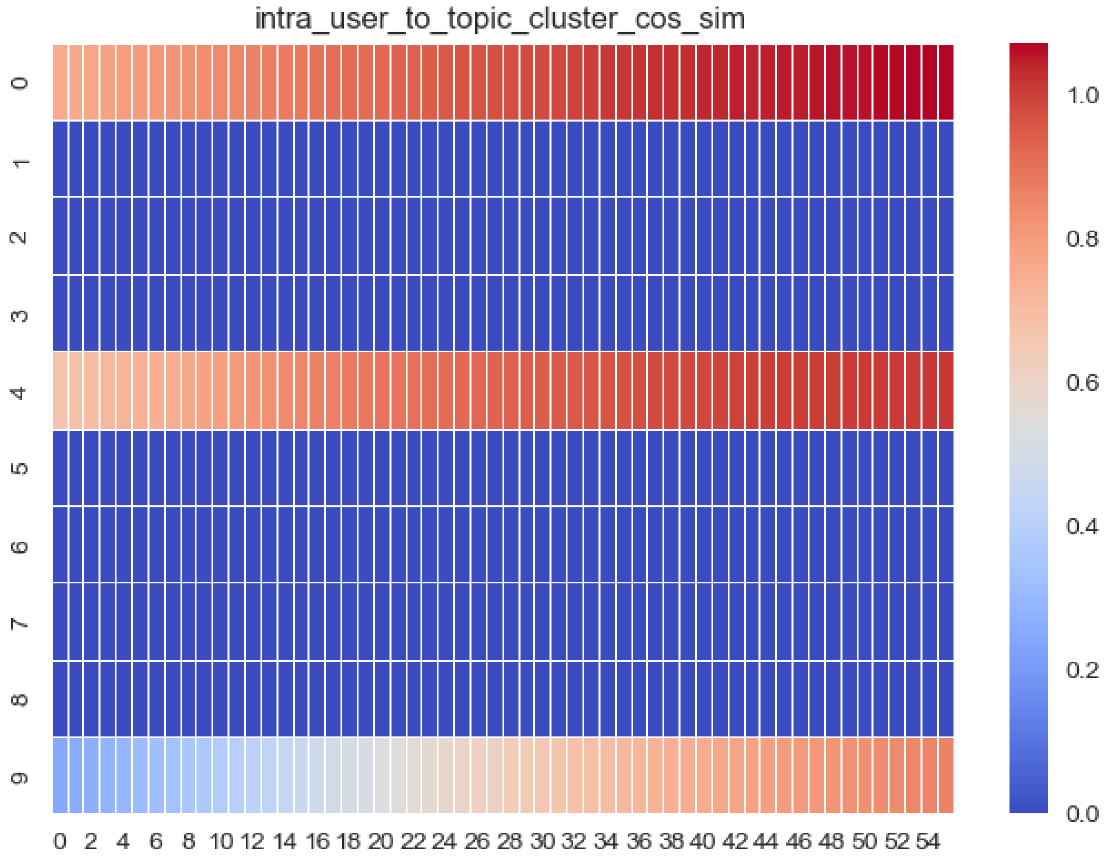
```
if requires_alpha:
    data = [[model_name, train_timesteps, run_timesteps, n_attrs, n_clusters, alpha]]

parameters_df = pd.DataFrame(data,
                             columns = columns)
for col in numeric_cols:
    parameters_df[col] = pd.to_numeric(parameters_df[col])
```

```
[ ]: measurements_df[10:20]
```

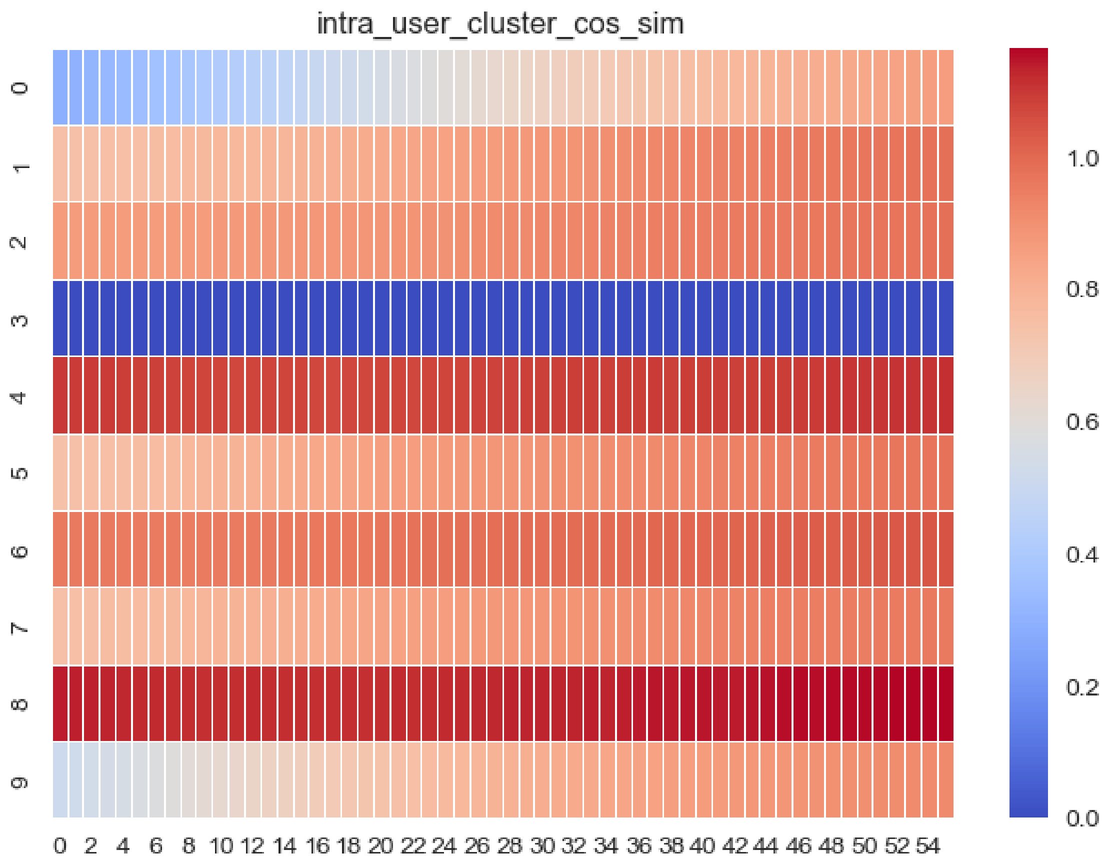
```
[ ]:                               interaction_histogram       mse  \
     10  [0.0, 0.0, 0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, …  0.130078
     11  [0.0, 0.0, 0.0, 0.0, 7.0, 0.0, 0.0, 0.0, 0.0, …  0.131707
     12  [1.0, 0.0, 0.0, 0.0, 16.0, 0.0, 0.0, 0.0, 0.0,…  0.134269
     13  [0.0, 0.0, 0.0, 1.0, 9.0, 0.0, 0.0, 0.0, 0.0, …  0.137580
     14  [0.0, 0.0, 0.0, 0.0, 6.0, 0.0, 0.0, 0.0, 0.0, …  0.140823
     15  [0.0, 0.0, 0.0, 0.0, 12.0, 0.0, 0.0, 0.0, 0.0,…  0.143469
     16  [0.0, 0.0, 0.0, 0.0, 13.0, 0.0, 0.0, 0.0, 0.0,…  0.146101
     17  [0.0, 0.0, 0.0, 0.0, 7.0, 0.0, 0.0, 0.0, 0.0, …  0.149254
     18  [0.0, 0.0, 0.0, 0.0, 6.0, 0.0, 0.0, 0.0, 0.0, …  0.152358
     19  [0.0, 0.0, 0.0, 0.0, 15.0, 0.0, 0.0, 0.0, 0.0,…  0.154906

         inter_cluster_interaction_similarity  \
     10                              0.020888
     11                              0.024067
     12                              0.031477
     13                              0.036620
     14                              0.043094
     15                              0.049023
     16                              0.054027
     17                              0.060100
     18                              0.065880
     19                              0.071610

         intra_cluster_interaction_similarity  \
     10                              0.019616
     11                              0.024369
     12                              0.029933
     13                              0.036254
     14                              0.041208
     15                              0.046259
     16                              0.051813
     17                              0.057114
     18                              0.062021
     19                              0.067680
```

```
                   intra_user_to_topic_cluster_cos_sim  \
10  [[0.8433218609343063], [0.0], [0.0], [0.0], [0…
11  [[0.8530567390346205], [0.0], [0.0], [0.0], [0…
12  [[0.8609442134307979], [0.0], [0.0], [0.0], [0…
13  [[0.8663214300558753], [0.0], [0.0], [0.0], [0…
14  [[0.8756615604179973], [0.0], [0.0], [0.0], [0…
15  [[0.8830529132377903], [0.0], [0.0], [0.0], [0…
16  [[0.8955351357207852], [0.0], [0.0], [0.0], [0…
17  [[0.9042818650481076], [0.0], [0.0], [0.0], [0…
18  [[0.9092452791355895], [0.0], [0.0], [0.0], [0…
19  [[0.9112310984650235], [0.0], [0.0], [0.0], [0…

                            intra_user_cluster_cos_sim  \
10  [[0.41290259349029734], [0.774627765096465], …
11  [[0.4267281898261442], [0.7735827272661405], […
12  [[0.44021965180452355], [0.777773448540368], …
13  [[0.45438996975281104], [0.78506245462420], […
14  [[0.46690641431131896], [0.786505051430536], …
15  [[0.4812666239981269], [0.7943462866187938], […
16  [[0.49466472035813897], [0.799817987039106], …
17  [[0.5080033204572441], [0.8045839505697426], […
18  [[0.5206649144613655], [0.8082583813839966], […
19  [[0.5340839053105585], [0.8167891262135092], […

               user_cluster_avg_distance_from_centroid  timesteps state  \
10  [[0.2516520864249138], [0.8261459366370246], […         10   run
11  [[0.25011823730119664], [0.8394056962360062], …         11   run
12  [[0.2487258074520182], [0.84801902619239], [0…        12   run
13  [[0.24735947497998306], [0.8564569207815927], …         13   run
14  [[0.24626319431406035], [0.8671825894090636], …         14   run
15  [[0.2449148934095171], [0.8754218946638347], …         15   run
16  [[0.2437915351950921], [0.8850152379298623], […         16   run
17  [[0.24273070296807822], [0.8940782565291586], …         17   run
18  [[0.2417895649262942], [0.9042088720976873], […         18   run
19  [[0.2407765508024434], [0.9125338004221754], […         19   run

         model
10  cosine_sim
11  cosine_sim
12  cosine_sim
13  cosine_sim
14  cosine_sim
15  cosine_sim
16  cosine_sim
17  cosine_sim
18  cosine_sim
19  cosine_sim
```

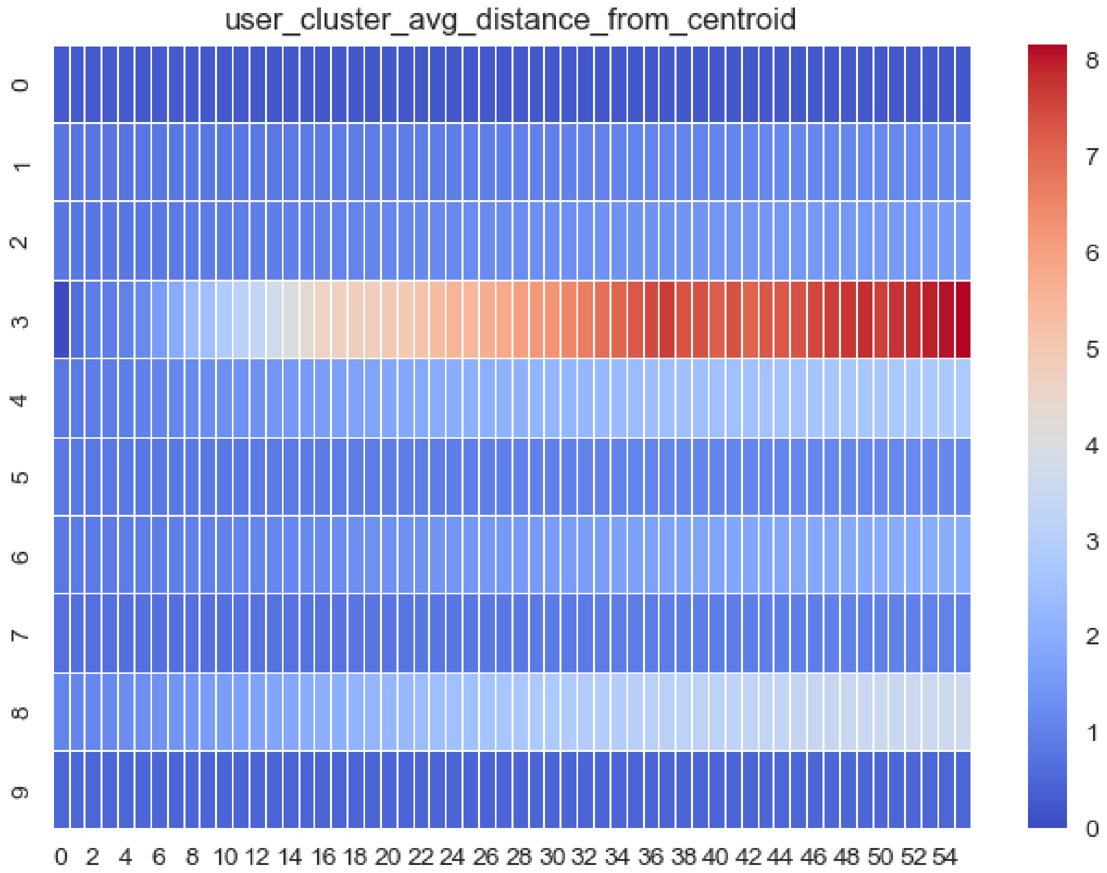
```
[ ]: cluster_degree_data = measurements_df['intra_user_to_topic_cluster_cos_sim'].
      ↪to_numpy()
     data = np.stack(cluster_degree_data, axis=0)
     data = data.reshape((data.shape[0], data.shape[1]))
     ax = sns.heatmap(data.T, linewidth = 0.5 , cmap = 'coolwarm' )

     plt.title("intra_user_to_topic_cluster_cos_sim")
     plt.show()
```



```
[ ]: cluster_degree_data = measurements_df['intra_user_cluster_cos_sim'].to_numpy()
     data = np.stack(cluster_degree_data, axis=0)
     data = data.reshape((data.shape[0], data.shape[1]))
     ax = sns.heatmap(data.T, linewidth = 0.5 , cmap = 'coolwarm' )

     plt.title("intra_user_cluster_cos_sim")
     plt.show()
```



```
[ ]: cluster_degree_data =␣
      ↪measurements_df['user_cluster_avg_distance_from_centroid'].to_numpy()
     data = np.stack(cluster_degree_data, axis=0)
     data = data.reshape((data.shape[0], data.shape[1]))
     ax = sns.heatmap(data.T, linewidth = 0.5 , cmap = 'coolwarm' )

     plt.title("user_cluster_avg_distance_from_centroid")
     plt.show()
```



```
[ ]: np.where(user_cluster_ids == 1)
```

```
[ ]: (array([ 12,  37,  48,  82,  86, 109, 124, 144, 173, 192, 215, 220, 245,
             279, 313, 319, 331, 335, 388, 393, 441, 452, 475, 495, 531, 541,
             565, 585, 647, 689, 757, 797, 886, 926, 942]),)
```

```
[ ]: val, count = np.unique(user_cluster_ids, return_counts=True)
     for i in range(len(val)):
         print(val[i],count[i])
```

```
0 559
1 35
2 43
3 1
4 12
5 35
6 20
7 58
8 10
9 170
```

[ ]:

```
# ts = measurements_df
# col_name = 'column_name'
# ignored_train_ts = train_timesteps+1
# plt.plot(ts, measurements_df[col_name][ignored_train_ts:], label=col_name)#,␣
 ↪alpha=0.5, color=colors(i))
```

[ ]:

```
# def MeanDistanceFromCentroid(recommender, user_cluster_ids, user_centroids,␣
 ↪n_clusts):
#     clusters = np.unique(user_cluster_ids)
#     avg_clust_dist = np.zeros((user_centroids.shape[0], 1))
#     for clust in clusters:
#         clust_users = np.where(user_cluster_ids == clust)[0]
#         clust_users_embed = recommender.users.actual_user_profiles.
 ↪value[clust_users,:]
#         dist = np.linalg.norm(user_centroids[clust] - clust_users_embed,␣
 ↪axis=1)
#         avg_clust_dist[clust] = np.mean(dist)
#     return avg_clust_dist

# print(MeanDistanceFromCentroid(model, user_cluster_ids, user_cluster_centers,␣
 ↪25))
# # print("\n", model.users.actual_user_profiles.value[7])
# # print(user_cluster_centers[0] - model.users.actual_user_profiles.value[7])
# # y = np.linalg.norm(user_cluster_centers[24] - model.users.
 ↪actual_user_profiles.value[[292, 342, 428, 456, 560, 652, 863, 885, 895,␣
 ↪915]], axis=1)
# # print(y)
# # print(np.mean(y))
```

```
[[0.43079131]
 [2.69412828]
 [0.28379012]
 [8.15055175]
 [1.61180713]
 [2.02779287]
 [2.87534142]
 [0.1691672 ]
 [0.5712539 ]
 [1.56233432]
 [1.06584597]
 [2.69351038]
 [1.23020428]
 [1.07379744]
 [4.05745483]
 [0.69444425]
 [1.03071026]
 [0.6469596 ]
 [0.30359675]
 [2.30333093]
 [2.57471539]
 [0.99729357]
 [1.64259691]
 [0.36358111]
 [1.99907837]]
```

```
[ ]:
```