myopic_measurements_10trainTimesteps_100runTimesteps_20nAttrs_10nC 0.8AttentionExp

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_69761/1535014153.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")
```

[ ]:
```
n_attrs=20
max_iter=1000
n_clusters=10
```

[ ]:
```
score_fn = None #'entropy'#'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 = 0.8
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}"
```

[ ]:
```
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.
Loaded clusters.
Loaded 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=10
model.startup_and_train(timesteps=train_timesteps)
```

```
100%|      | 10/10 [00:09<00:00,  1.10it/s]
```

[ ]:
```
run_timesteps=100
model.run(timesteps=run_timesteps)
```

```
100%|      | 100/100 [08:14<00:00,  4.94s/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])
```

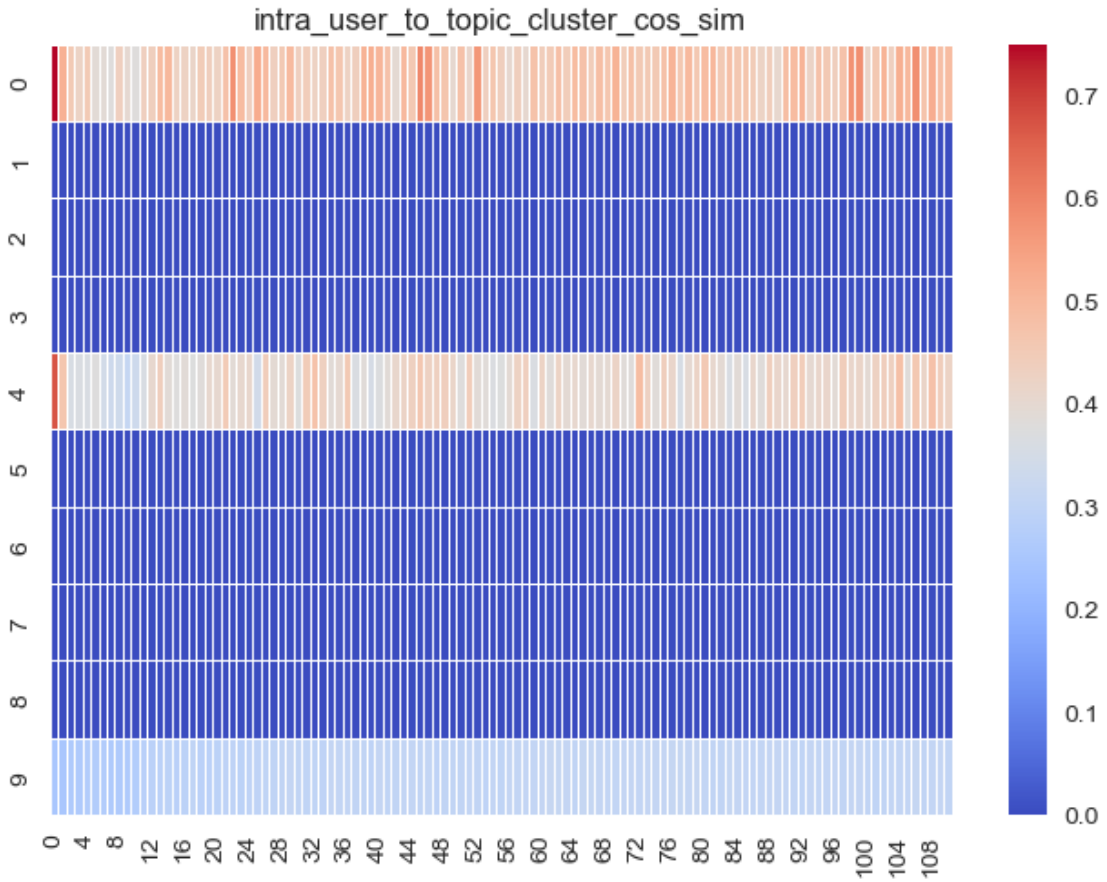
[ ]:
```
measurements_df[10:20]
```

[ ]:
```
                              interaction_histogram       mse  \
10  [3.0, 1.0, 0.0, 2.0, 3.0, 1.0, 3.0, 1.0, 1.0, …  0.142518
11  [3.0, 0.0, 3.0, 2.0, 0.0, 0.0, 2.0, 2.0, 7.0, …  1.194380
12  [1.0, 2.0, 2.0, 0.0, 0.0, 0.0, 1.0, 1.0, 9.0, …  1.249569
13  [2.0, 0.0, 3.0, 0.0, 0.0, 0.0, 2.0, 0.0, 5.0, …  1.294632
14  [3.0, 1.0, 2.0, 1.0, 0.0, 0.0, 2.0, 0.0, 4.0, …  1.323929
15  [1.0, 0.0, 3.0, 1.0, 0.0, 0.0, 3.0, 1.0, 10.0,…  1.356909
16  [1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 3.0, 0.0, 1.0, …  1.376578
17  [1.0, 0.0, 2.0, 0.0, 0.0, 1.0, 3.0, 0.0, 1.0, …  1.394039
18  [3.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 2.0, …  1.417013
19  [1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, …  1.431792

    inter_cluster_interaction_similarity  \
10                              0.006440
11                              0.007386
12                              0.008869
13                              0.010216
14                              0.011838
15                              0.013437
16                              0.014756
17                              0.015725
18                              0.016889
19                              0.018047

    intra_cluster_interaction_similarity  \
10                              0.006811
11                              0.007813
12                              0.009147
13                              0.010463
14                              0.011940
15                              0.013322
16                              0.014646
17                              0.015864
```

```
18                                0.016980
19                                0.017989

                intra_user_to_topic_cluster_cos_sim  \
10  [[0.3804936479081021], [0.0], [0.0], [0.0], [0…
11  [[0.4341003757852174], [0.0], [0.0], [0.0], [0…
12  [[0.437622662570702], [0.0], [0.0], [0.0], [0…
13  [[0.4897886595581509], [0.0], [0.0], [0.0], [0…
14  [[0.4987783069633305], [0.0], [0.0], [0.0], [0…
15  [[0.425585415899965], [0.0], [0.0], [0.0], [0…
16  [[0.42711357868650324], [0.0], [0.0], [0.0], […
17  [[0.4175277913399676], [0.0], [0.0], [0.0], [0…
18  [[0.4468750594108991], [0.0], [0.0], [0.0], [0…
19  [[0.4370575264530409], [0.0], [0.0], [0.0], [0…

                         intra_user_cluster_cos_sim  \
10  [[0.27037538800691874], [0.3680831427487129], …
11  [[0.28391769961948277], [0.35977160227359545],…
12  [[0.2905536733873685], [0.35201353664764595], …
13  [[0.28790143507123744], [0.3280343087591958], …
14  [[0.29571684140138127], [0.32236902649417604],…
15  [[0.2837633562749486], [0.33740591264598957], …
16  [[0.2915614075569197], [0.3298735595164744], […
17  [[0.3028696537133882], [0.3285279707213021], […
18  [[0.2939672222647939], [0.33642091978157035], …
19  [[0.2894341480872474], [0.3336036570485342], […

            user_cluster_avg_distance_from_centroid  timesteps  state  \
10  [[0.2975434514035181], [1.5709066528372464], […         10  train
11  [[0.2982956869530559], [1.5458014316134292], […         11    run
12  [[0.2997884899346219], [1.5111423400403403], […         12    run
13  [[0.3023164387035552], [1.5449874253492923], […         13    run
14  [[0.30303782251068134], [1.5485310207891971], …         14    run
15  [[0.30420816096782116], [1.53066165082462], [2…         15    run
16  [[0.30330678399410177], [1.5346861239786995], …         16    run
17  [[0.3024228375624952], [1.5431655629364553], […         17    run
18  [[0.30476925789066683], [1.552942981726751], […         18    run
19  [[0.3074573884179322], [1.5663824877698638], […         19    run

     model
10  myopic
11  myopic
12  myopic
13  myopic
14  myopic
15  myopic
16  myopic
```

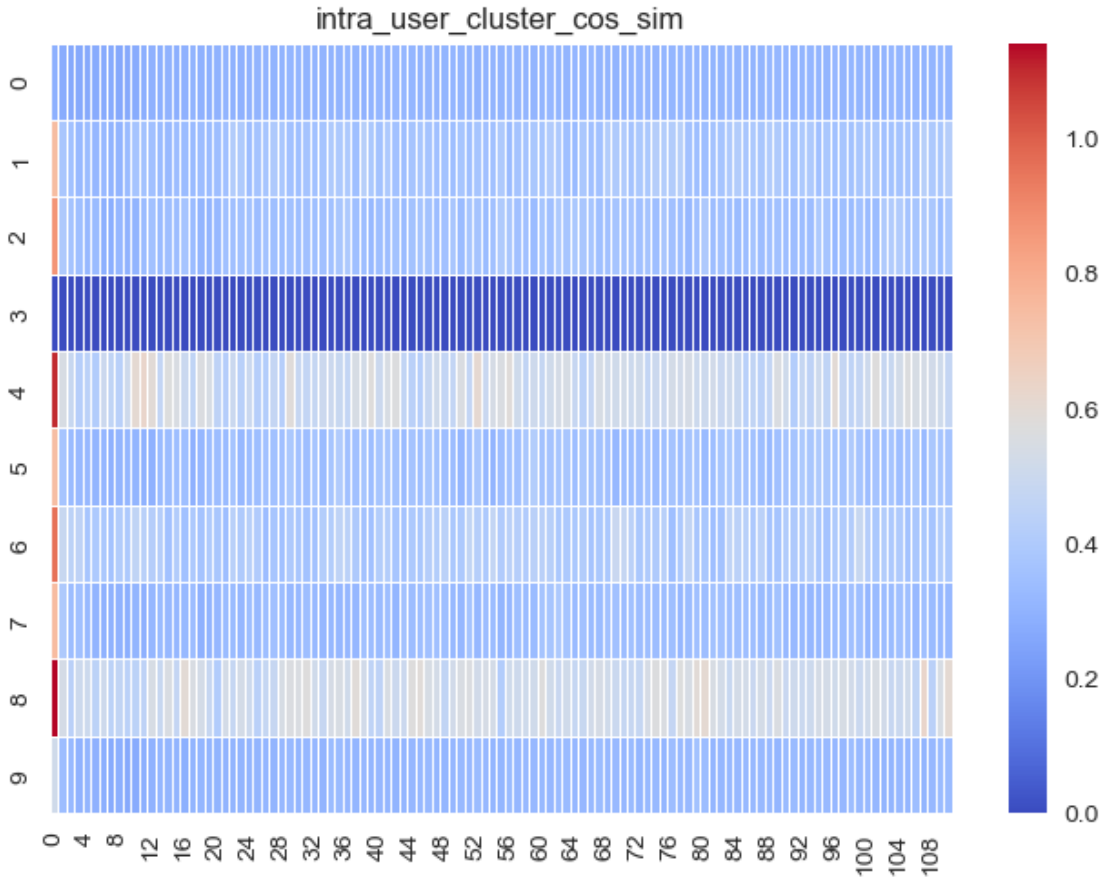
```
17  myopic
18  myopic
19  myopic
```

[ ]:
```
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()
```

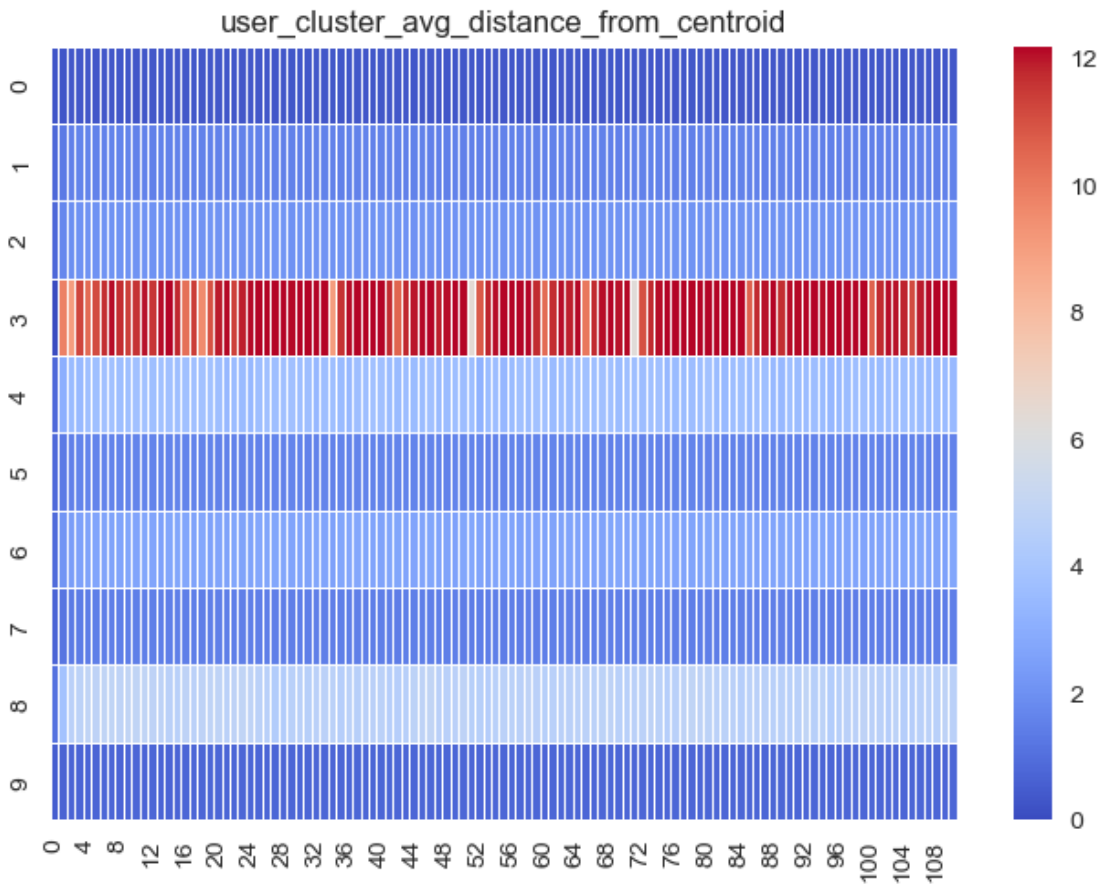


[ ]:
```
cluster_degree_data = measurements_df['intra_user_cluster_cos_sim'].to_numpy()
data = np.stack(cluster_degree_data, axis=0)
data = data.reshape((data.shape[0], data.shape[1]))
ax = sns.heatmap(data.T, linewidth = 0.5 , cmap = 'coolwarm' )

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
```

[ ]:
```
plot_df = measurements_df[['timesteps', 'mse',␣
↪'inter_cluster_interaction_similarity','intra_cluster_interaction_similarity']].
↪copy()
print(plot_df.shape)

mean_cos_sim = [np.mean(x) for x in␣
↪measurements_df['intra_user_cluster_cos_sim'].to_numpy()]
plot_df['avg_intra_user_cluster_cos_sim'] = mean_cos_sim

mean_centroid_dist = [np.mean(x) for x in␣
↪measurements_df['user_cluster_avg_distance_from_centroid'].to_numpy()]
plot_df['avg_distance_from_centroid'] = mean_centroid_dist
```

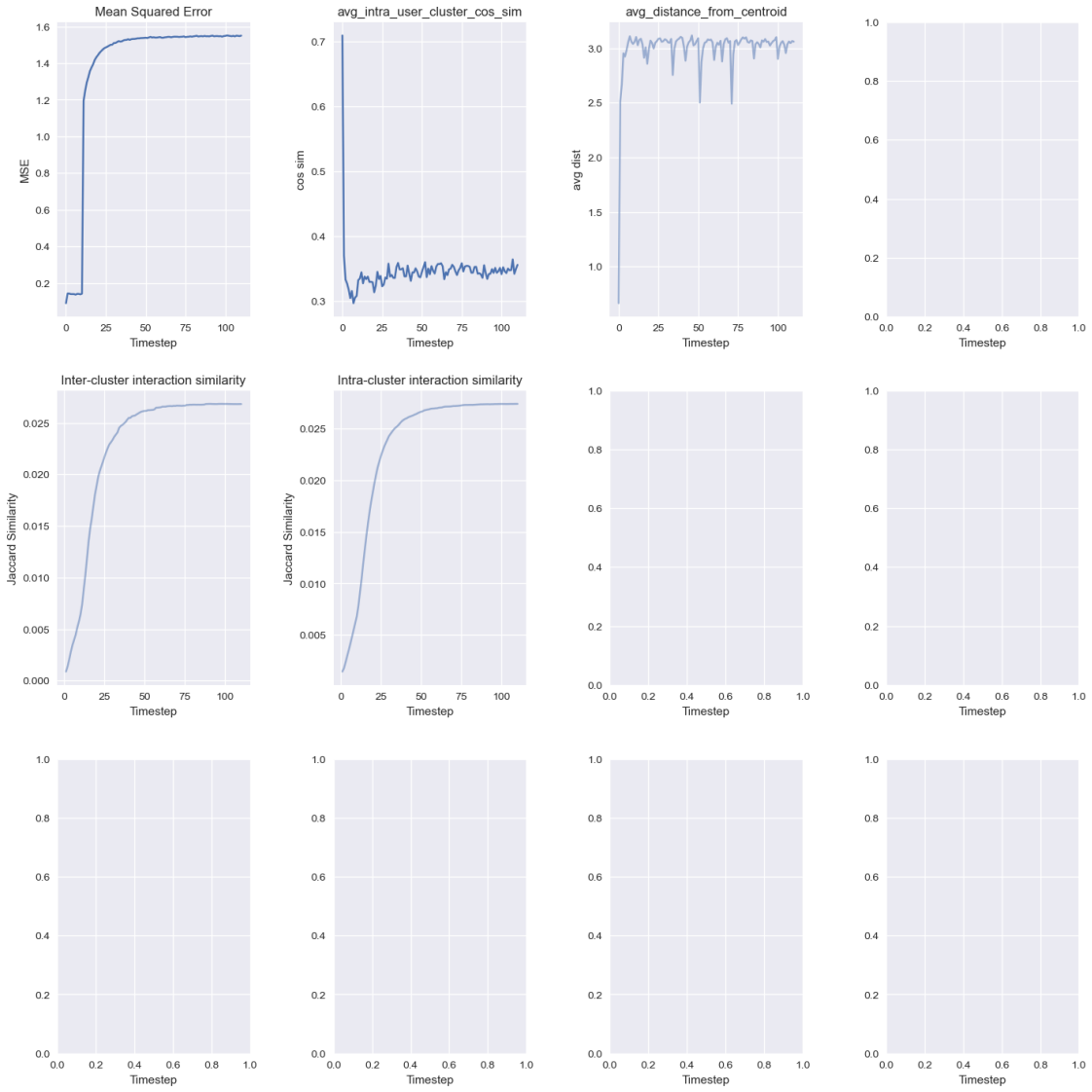
```
(111, 4)
```

[ ]:
```
def plot_measurements_1_df(df, model_name):
    fig, ax = plt.subplots(3, 4, figsize=(15, 15))
    fig.tight_layout(pad=5.0)

    ts = df['timesteps'].values

    cols_exclude = ['timesteps', 'interaction_histogram', 'user_mse']

    # if not requires_alpha:
    #     alpha = np.nan
    alpha=0.5

    # mse
    ax[0,0].plot(ts, df['mse'], label=model_name)
    ax[0,0].set_title('Mean Squared Error')
    ax[0,0].set_ylabel('MSE')
    # # recall_at_k
    # ax[0,1].plot(ts, df['recall_at_k'], label=model_name)
    # ax[0,1].set_title('Recall at k')
    # ax[0,1].set_ylabel('Recall')
    ax[0,1].plot(ts, df['avg_intra_user_cluster_cos_sim'], label=model_name)
    ax[0,1].set_title('avg_intra_user_cluster_cos_sim')
    ax[0,1].set_ylabel('cos sim')
    # # interaction_spread
    # ax[0,2].plot(ts, df['interaction_spread'], label=model_name, alpha=alpha)
    # ax[0,2].set_title('Interaction Spread')
    # ax[0,2].set_ylabel('Jaccard Similarity')
    ax[0,2].plot(ts, df['avg_distance_from_centroid'], label=model_name,␣
↪alpha=alpha)
    ax[0,2].set_title('avg_distance_from_centroid')
    ax[0,2].set_ylabel('avg dist')
```

```
    # ax[0,3].axis('off')

    # inter_cluster_interaction_similarity
    ax[1,0].plot(ts, df['inter_cluster_interaction_similarity'],␣
↪label=model_name, alpha=alpha)
    ax[1,0].set_title('Inter-cluster interaction similarity')
    ax[1,0].set_ylabel('Jaccard Similarity')
    # intra_cluster_interaction_similarity
    ax[1,1].plot(ts, df['intra_cluster_interaction_similarity'],␣
↪label=model_name, alpha=alpha)
    ax[1,1].set_title('Intra-cluster interaction similarity')
    ax[1,1].set_ylabel('Jaccard Similarity')
    # # inter_cluster_rec_similarity
    # ax[1,2].plot(ts, df['inter_cluster_rec_similarity'], label=model_name,␣
↪alpha=alpha)
    # ax[1,2].set_title('Inter-cluster interaction similarity')
    # ax[1,2].set_ylabel('Jaccard Similarity')
    # # intra_cluster_rec_similarity
    # ax[1,3].plot(ts, df['intra_cluster_rec_similarity'], label=model_name,␣
↪alpha=alpha)
    # ax[1,3].set_title('Intra-cluster interaction similarity')
    # ax[1,3].set_ylabel('Jaccard Similarity')

    # # mean_num_topics
    # ax[2,0].plot(ts, df['mean_num_topics'], label=model_name, alpha=alpha)
    # ax[2,0].set_title('Mean number of topics interacted with per timestep')
    # ax[2,0].set_ylabel('Avg. number of topics')
    # # inter_cluster_mean_interaction_distance
    # ax[2,1].plot(ts, df['inter_cluster_mean_interaction_distance'],␣
↪label=model_name, alpha=alpha)
    # ax[2,1].set_title('Inter-cluster interaction similarity')
    # ax[2,1].set_ylabel('Avg. Interaction Distance')
    # # intra_cluster_mean_interaction_distance
    # ax[2,2].plot(ts, df['intra_cluster_mean_interaction_distance'],␣
↪label=model_name, alpha=alpha)
    # ax[2,2].set_title('Intra-cluster interaction similarity')
    # ax[2,2].set_ylabel('Avg. Interaction Distance')
    # ax[2,3].axis('off')

    for a in ax:
        for b in a:
            b.set_xlabel('Timestep')

    # fig.legend(legend_lines, legend_names, loc='upper center', fontsize=14,␣
↪frameon=False, ncol=5, bbox_to_anchor=(.5, 1.05))
```

```
print(measurements_path)
plot_measurements_1_df(plot_df, model_name)
```

```
artefacts/supplementary/measurements/myopic_measurements_10trainTimesteps_100run
Timesteps_20nAttrs_10nClusters_0.8Drift_-0.8AttentionExp.csv
```



[ ]:
```
# Get user and item representations using NMF
# user_representation, item_representation

# Define topic clusters using NMF
print('item_cluster_ids: ', item_cluster_ids.shape)
print('item_cluster_centers:', item_cluster_centers.shape)
print('user_cluster_ids: ', user_cluster_ids.shape)
```

```
print('user_cluster_centers: ', user_cluster_centers.shape)
print('user_item_cluster_mapping: ', user_item_cluster_mapping.shape)
```

```
item_cluster_ids:  (1682,)
item_cluster_centers: (10, 20)
user_cluster_ids:  (943,)
user_cluster_centers:  (10, 20)
user_item_cluster_mapping:  (943,)
```

[ ]: