trecs_exp_v2

December 17, 2022

[ ]:

```
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, '../t-recs/')
from trecs.metrics import Measurement
from trecs.metrics import MSEMeasurement, InteractionSpread, InteractionSpread,␣
 ↪InteractionSimilarity, RecSimilarity, RMSEMeasurement, InteractionMeasurement
from trecs.components import Users
import trecs.matrix_ops as mo
import src.globals as globals
import seaborn as sns

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,␣
 ↪top_k_reranking
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_27229/2185722975.py:2
```

```
7: 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")
```

1 RecommenderSystem on MovieLens

[ ]:
```
n_attrs=20
max_iter=1000
n_clusters=25
```

[ ]:
```
score_fn = 'entropy' #'content_fairness' 'top_k_reranking'
probabilistic = False
globals.ALPHA = 0.1 #0.01 # 0.1
alpha = globals.ALPHA

# User parameters
drift = 0.05
attention_exp=-0.8
```

1.1 Embeddings

[ ]:
```
binary_ratings_matrix = load_and_process_movielens(file_path='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)
```

```
Loaded embeddings.
```

[ ]:
```
# 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)
```

```
/Users/madisonthantu/miniforge3/envs/fairRS/lib/python3.8/site-
packages/sklearn/base.py:329: UserWarning: Trying to unpickle estimator KMeans
from version 1.0.2 when using version 1.1.3. This might lead to breaking code or
invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-
limitations
  warnings.warn(
```

```
Loaded clusters.
Loaded clusters.
```

```
/Users/madisonthantu/miniforge3/envs/fairRS/lib/python3.8/site-
packages/sklearn/base.py:329: UserWarning: Trying to unpickle estimator KMeans
from version 1.0.2 when using version 1.1.3. This might lead to breaking code or
invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-
limitations
  warnings.warn(
```

1.1.1 Model

[ ]:

```python
num_users = len(user_representation)
num_items = 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)
```

```
Number of items: 20
Number of users: 943
```

[ ]:

```python
# Create user_pairs by pairing users only with others that are not in the same cluster
user_item_cluster_mapping = user_topic_mapping(user_representation, item_cluster_centers) # TODO: Remove?
experiment_name = 'users_by_topic'
# 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)
```

[ ]:

```python
mse = MSEMeasurement()
measurements = [
    InteractionMeasurement(),
    MSEMeasurement(),
    InteractionSpread(),
    InteractionSimilarity(pairs=inter_cluster_user_pairs, name='inter_cluster_interaction_similarity'),
    InteractionSimilarity(pairs=intra_cluster_user_pairs, name='intra_cluster_interaction_similarity'),
    RecSimilarity(pairs=inter_cluster_user_pairs, name='inter_cluster_rec_similarity'),
    RecSimilarity(pairs=intra_cluster_user_pairs, name='intra_cluster_rec_similarity'),
    UserMSEMeasurement(),
    SerendipityMetric(),
    DiversityMetric(),
    NoveltyMetric(),
    RecallMeasurement(),
    MeanNumberOfTopics(),
]
```

[ ]:

```
# 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_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
    elif score_fn == 'top_k_reranking':
        config['score_fn'] = top_k_reranking
    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)
```

1.2 Simulation

```
[ ]: # Fair Model
     train_timesteps=10
     model.startup_and_train(timesteps=train_timesteps)
```

```
100%|          | 10/10 [00:43<00:00,  4.35s/it]
```

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

```
100%|          | 100/100 [20:31<00:00, 12.31s/it]
```

```
[ ]: import src
     reload(src.utils)
     from src.utils import *

     # 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 actual user preferences
     final_preferences_dir = 'artefacts/supplementary/final_preferences/'
     file_prefix = f'{model_name}_final_preferences'
     final_preferences_path = final_preferences_dir + file_prefix + parameters + '.
       ↪npy'
     np.save(final_preferences_path, model.users.actual_user_profiles.value,␣
       ↪allow_pickle=True)

     # Save measurements
     measurements_dir = f'artefacts/supplementary/measurements/'
     file_prefix = f'{model_name}_measurements'

     measurements_path = measurements_dir + file_prefix + parameters + '.csv'
     # np.set_printoptions(threshold=sys.maxsize)
     measurements_df = load_or_create_measurements_df(model, model_name,␣
       ↪train_timesteps, measurements_path)
     # measurements_df['interaction_histogram'] =␣
       ↪measurements_df['interaction_histogram'].tolist()
     # saving interaction histogram
     path_interaction_histogram =␣
       ↪f'{measurements_dir}{model_name}_interaction_histogram{parameters}.csv'
     interaction_hist = measurements_df['interaction_histogram'].copy()
     interaction_hist[0] = np.repeat(np.nan, interaction_hist[1].shape[0], axis=0)
     interaction_hist = np.stack(interaction_hist.values)
     test = pd.DataFrame(interaction_hist)
```

```
test.to_csv(path_interaction_histogram, index=False)
# saving use_mse histogram
path_user_mse_histogram =␣
 ↪f'{measurements_dir}{model_name}_user_mse_histogram{parameters}.csv'
user_mse = measurements_df['user_mse'].copy()
user_mse = np.stack(user_mse.values)
test = pd.DataFrame(user_mse.T)
test['clusterID'] = user_cluster_ids
test.to_csv(path_user_mse_histogram, index=False)
# saving all measurements
measurements_df.to_csv(measurements_path, index=False)
print('Measurements saved.')
```

```
Measurements saved.
```

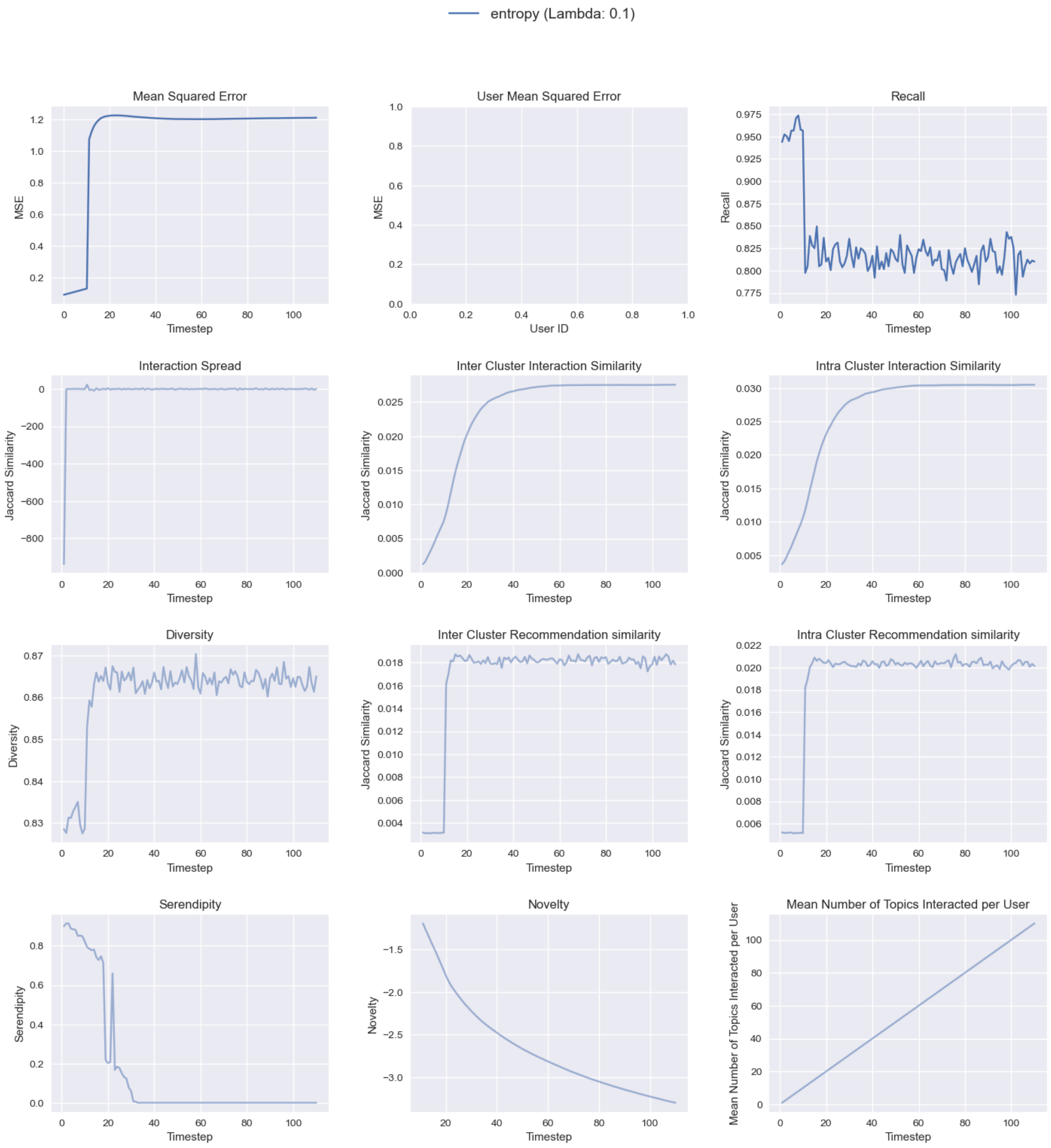
1.3 Analysis

[ ]:

```
# 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])
```

[ ]:

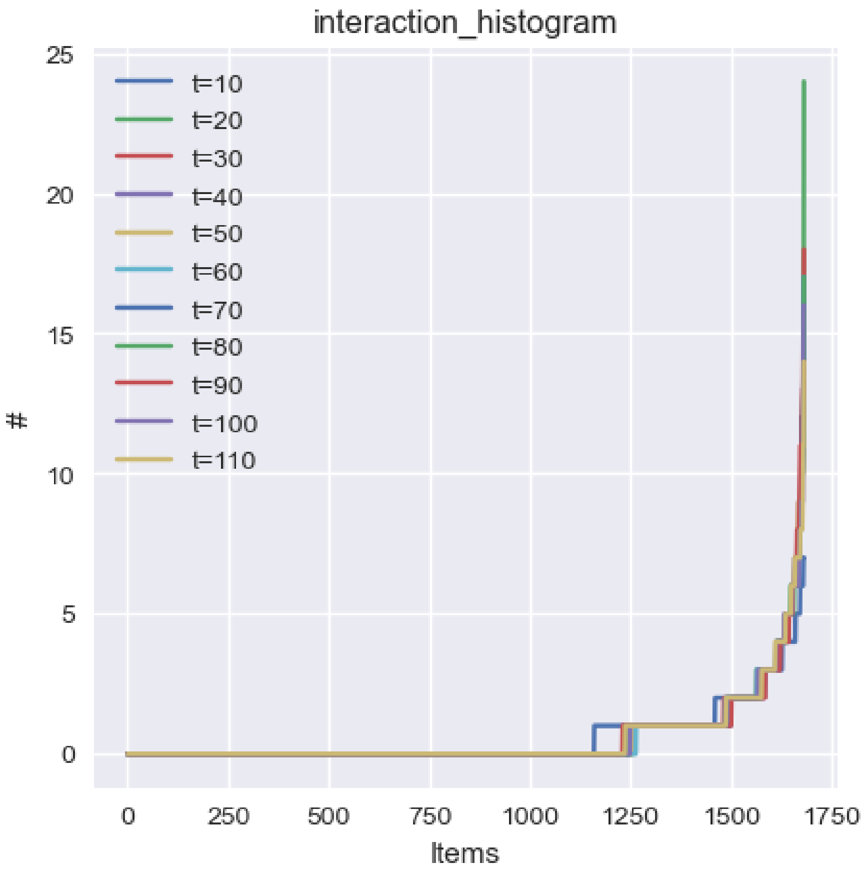
```
plot_measurements([measurements_df], parameters_df)
```



2 Experiments

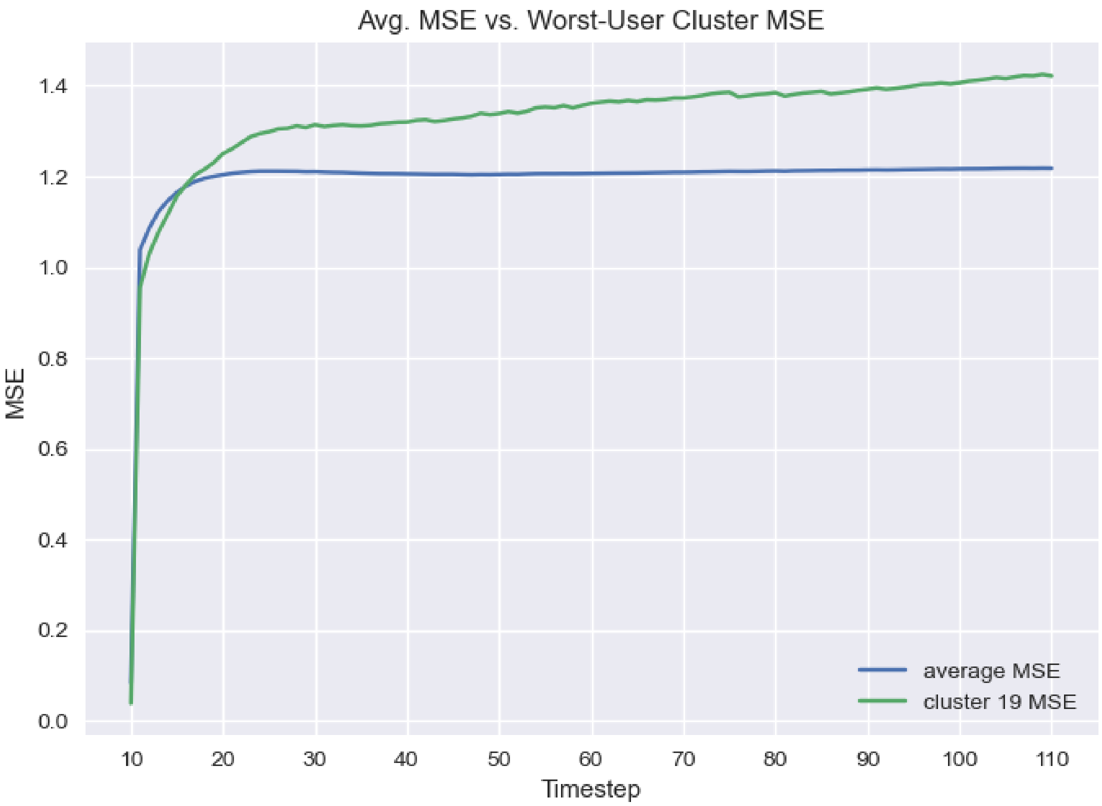
[ ]:

```
def plot_histograms(measurements_df, column_name):
    # for every 10 timesteps plot line in different color
    fig= plt.figure(figsize=(5, 5))

    timesteps = len(measurements_df)
    items = np.arange(len(measurements_df[column_name][1]))
    for i in range(10, timesteps+1, 10):
        counts = measurements_df[column_name][i]
        plt.plot(items, sorted(counts), label=f't={i}')

    plt.title(column_name)
    plt.xlabel('Items')
    plt.ylabel('#')
    plt.legend()

plot_histograms(measurements_df, 'interaction_histogram')
```



```
[ ]: # df_user_mse = pd.DataFrame(user_mse)
     # df_user_mse['clusterID'] = user_cluster_ids
     import src
     reload(src.utils)
     from src.utils import *
     df_user_mse = pd.read_csv(path_user_mse_histogram)
     analyze_user_mse(df_user_mse, train_timesteps)
```

```
[  0  10  20  30  40  50  60  70  80  90 100]
```


Avg. MSE vs. Worst-User Cluster MSE
MSE
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
10
20
30
40
50
60
70
80
90
100
110
Timestep
average MSE
cluster 19 MSE

[ ]: