homogenization_analysis

December 19, 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_69904/2185722975.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")
```

[ ]:
```
retrain_paths = {
    'cosine_sim (=0.01)':'artefacts/supplementary/measurements/
 ↪cosine_sim_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp_0.01Lambda.csv',
    'cosine_sim (=0.1)':'artefacts/supplementary/measurements/
 ↪cosine_sim_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp_0.1Lambda.csv',
    'entropy (=0.1)':'artefacts/supplementary/measurements/
 ↪entropy_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp_0.1Lambda.csv',
    'entropy (=1.0)':'artefacts/supplementary/measurements/
 ↪entropy_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp_1.0Lambda.csv',
    # 'entropy (=10.0)':'artefacts/supplementary/measurements/
 ↪entropy_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp_10.0Lambda.csv',
    'top_k_reranking':'artefacts/supplementary/measurements/
 ↪top_k_reranking_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp.csv',
    'myopic':'artefacts/supplementary/measurements/
 ↪myopic_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp.csv'
}

no_retrain_paths = {
    'cosine_sim (=0.01)':'artefacts/no_train_between_runs/supplementary/
 ↪measurements/
 ↪cosine_sim_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp_0.01Lambda.csv',
    'cosine_sim (=0.1)':'artefacts/no_train_between_runs/supplementary/
 ↪measurements/
 ↪cosine_sim_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp_0.1Lambda.csv',
    'entropy (=0.1)':'artefacts/no_train_between_runs/supplementary/
 ↪measurements/
 ↪entropy_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp_0.1Lambda.csv',
    'entropy (=1.0)':'artefacts/no_train_between_runs/supplementary/
 ↪measurements/
 ↪entropy_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
 ↪05Drift_-0.8AttentionExp_1.0Lambda.csv',
```

```
    # 'entropy (=10.0)':'artefacts/no_train_between_runs/supplementary/
↪measurements/
↪entropy_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
↪05Drift_-0.8AttentionExp_10.0Lambda.csv',
    'top_k_reranking':'artefacts/no_train_between_runs/supplementary/
↪measurements/
↪top_k_reranking_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
↪05Drift_-0.8AttentionExp.csv',
    'myopic':'artefacts/no_train_between_runs/supplementary/measurements/
↪myopic_measurements_10trainTimesteps_100runTimesteps_20nAttrs_25nClusters_0.
↪05Drift_-0.8AttentionExp.csv'
}
```

[ ]:
```
def plot_homogenization(dfs, parameters, training_between):
    fig, ax = plt.subplots(2, 4, figsize=(20, 10))
    fig.tight_layout(pad=5.0)
    colors = plt.get_cmap('tab10')

    # plot rec_similarity with timesteps on x axis
    # idxs = [(0,0), (0,1), (1,0), (1,1), (2,0), (2,1), (3,0), (3,1)]
    idxs = [(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3)]
    legend_lines, legend_names = [], []
    for i, df in enumerate(dfs):
        ts = df['timesteps']
        name = parameters[i][0]
        # if not np.isnan(parameters_df.loc[i, 'Lambda']):
        #     name += f" (Lambda: {parameters_df.loc[i, 'Lambda']})"
        legend_names.append(name)

        line, = ax[idxs[0]].plot(ts,␣
↪df['inter_cluster_interaction_similarity'], label=name, alpha=0.5,␣
↪color=colors(i))
        if 'intra_cluster_interaction_similarity' in df.columns:
            ax[idxs[1]].plot(ts, df['intra_cluster_interaction_similarity'],␣
↪label=name, alpha=0.5, color=colors(i))
        if 'inter_cluster_rec_similarity' in df.columns:
            ax[idxs[2]].plot(ts, df['inter_cluster_rec_similarity'],␣
↪label=name, alpha=0.5, color=colors(i))
        if 'intra_cluster_rec_similarity' in df.columns:
            ax[idxs[3]].plot(ts, df['intra_cluster_rec_similarity'],␣
↪label=name, alpha=0.5, color=colors(i))
        if 'interaction_spread' in df.columns:
            ax[idxs[4]].plot(ts, df['interaction_spread'], label=name, alpha=0.
↪5, color=colors(i))
        if 'mean_num_topics' in df.columns:
```

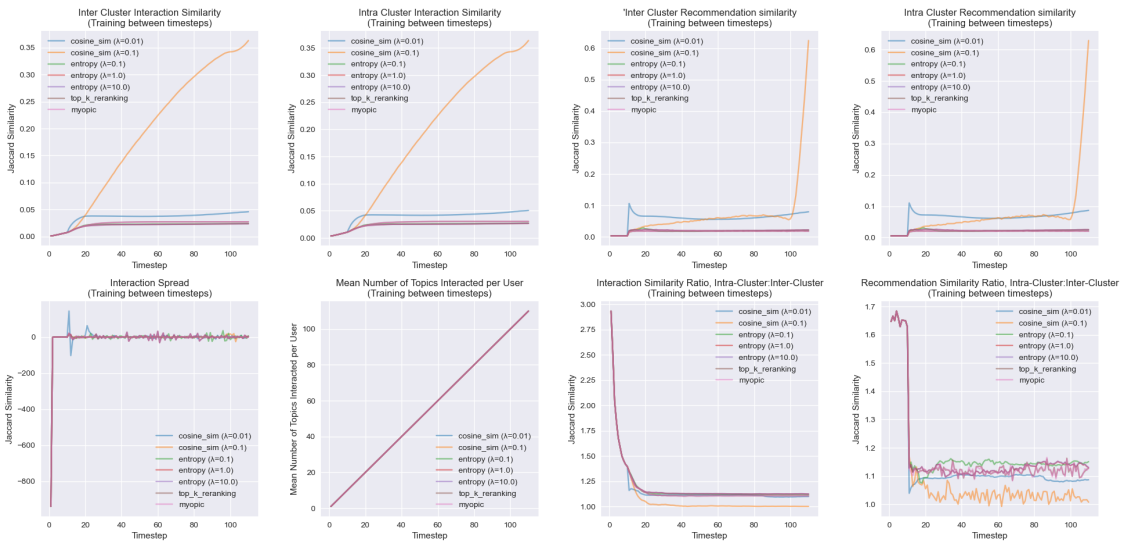
```
            ax[idxs[5]].plot(ts, df['mean_num_topics'], label=name, alpha=0.5,␣
↪color=colors(i))

        ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
        ax[idxs[6]].plot(ts, ratio_intrxn_sim, label=name, alpha=0.5,␣
↪color=colors(i))

        ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
        ax[idxs[7]].plot(ts, ratio_rec_sim, label=name, alpha=0.5,␣
↪color=colors(i))

        legend_lines.append(line)

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

    title_suffix = ''

    if training_between:
        title_suffix = '(Training between timesteps)'
    else:
        title_suffix = '(No training between timesteps)'

    ax[idxs[0]].set_title(f"Inter Cluster Interaction␣
↪Similarity\n{title_suffix}")
    ax[idxs[0]].set_ylabel('Jaccard Similarity')

    ax[idxs[1]].set_title(f"Intra Cluster Interaction␣
↪Similarity\n{title_suffix}")
    ax[idxs[1]].set_ylabel('Jaccard Similarity')

    ax[idxs[2]].set_title(f"'Inter Cluster Recommendation␣
↪similarity\n{title_suffix}")
    ax[idxs[2]].set_ylabel('Jaccard Similarity')

    ax[idxs[3]].set_title(f"Intra Cluster Recommendation␣
↪similarity\n{title_suffix}")
    ax[idxs[3]].set_ylabel('Jaccard Similarity')

    ax[idxs[4]].set_title(f"Interaction Spread\n{title_suffix}")
    ax[idxs[4]].set_ylabel('Jaccard Similarity')
```

```
    ax[idxs[5]].set_title(f"Mean Number of Topics Interacted per␣
↪User\n{title_suffix}")
    ax[idxs[5]].set_ylabel('Mean Number of Topics Interacted per User')

    ax[idxs[6]].set_title(f"Interaction Similarity Ratio, Intra-Cluster:
↪Inter-Cluster\n{title_suffix}")
    ax[idxs[6]].set_ylabel('Jaccard Similarity')

    ax[idxs[7]].set_title(f"Recommendation Similarity Ratio, Intra-Cluster:
↪Inter-Cluster\n{title_suffix}")
    ax[idxs[7]].set_ylabel('Jaccard Similarity')

    # fig.legend(legend_lines,
    #            legend_names,
    #            loc='upper center',
    #            fontsize=12,
    #            frameon=False,
    #            ncol=5,
    #            bbox_to_anchor=(.5, 1.02))

    for i in idxs:
        if training_between:
            ax[i].legend(loc='upper left')
        else:
            ax[i].legend(loc='lower right')

    ax[idxs[4]].legend(loc='lower right')
    ax[idxs[5]].legend(loc='lower right')

    ax[idxs[-1]].legend(loc='upper right')
    ax[idxs[-2]].legend(loc='upper right')

    if training_between:
        plt.savefig('figures/YES_train_between_homogen_analysis.png')
    else:
        plt.savefig('figures/NO_train_between_homogen_analysis.png')

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

1 Train between timesteps

[ ]:
```
paths = [
    ('cosine_sim ( =0.01)', retrain_paths['cosine_sim ( =0.01)']),
    ('cosine_sim ( =0.1)', retrain_paths['cosine_sim ( =0.1)']),
    ('entropy ( =0.1)', retrain_paths['entropy ( =0.1)']),
    ('entropy ( =1.0)', retrain_paths['entropy ( =1.0)']),
    # ('entropy ( =10.0)', retrain_paths['entropy ( =10.0)']),
    ('top_k_reranking', retrain_paths['top_k_reranking']),
    ('myopic', retrain_paths['myopic'])
]

# paths
dfs_homogen = []
for i in paths:
    df = pd.read_csv(i[1])
    dfs_homogen.append(df)
```

[ ]:
```
plot_homogenization(dfs_homogen, paths, training_between=True)
```

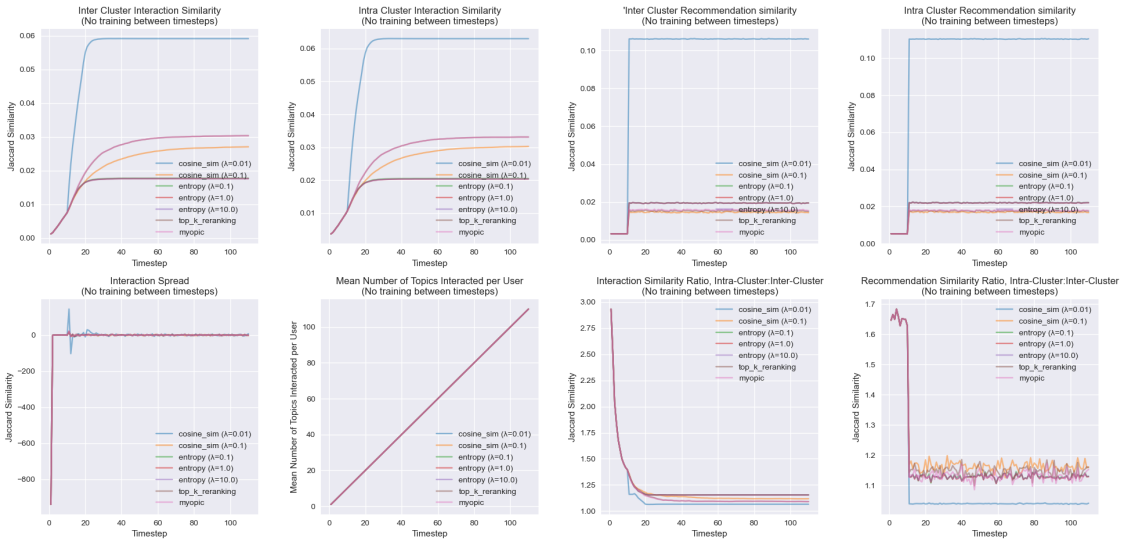


2 Testing NO training between steps

[ ]:
```
paths = [
    ('cosine_sim ( =0.01)', no_retrain_paths['cosine_sim ( =0.01)']),
    ('cosine_sim ( =0.1)', no_retrain_paths['cosine_sim ( =0.1)']),
    ('entropy ( =0.1)', no_retrain_paths['entropy ( =0.1)']),
    ('entropy ( =1.0)', no_retrain_paths['entropy ( =1.0)']),
    # ('entropy ( =10.0)', no_retrain_paths['entropy ( =10.0)']),
```

```
    ('top_k_reranking', no_retrain_paths['top_k_reranking']),
    ('myopic', no_retrain_paths['myopic'])
]

# paths
dfs_homogen = []
for i in paths:
    df = pd.read_csv(i[1])
    dfs_homogen.append(df)
```

[ ]:
```
plot_homogenization(dfs_homogen, paths, training_between=False)
```



3 Plotting just ratios

[ ]:
```
def plot_ratios1(dfs, parameters):
    fig, ax = plt.subplots(1, 4, figsize=(20, 5))
    fig.tight_layout(pad=5.0)
    colors = plt.get_cmap('tab10')
    alpha = 0.75

    # plot rec_similarity with timesteps on x axis
    # idxs = [(0,0), (0,1), (1,0), (1,1), (2,0), (2,1), (3,0), (3,1)]
    idxs = [0,1,2,3]
    legend_lines, legend_names = [], []
    for i, df in enumerate(dfs):
        ts = df['timesteps']
        name = parameters[i][0]
        # if not np.isnan(parameters_df.loc[i, 'Lambda']):
```

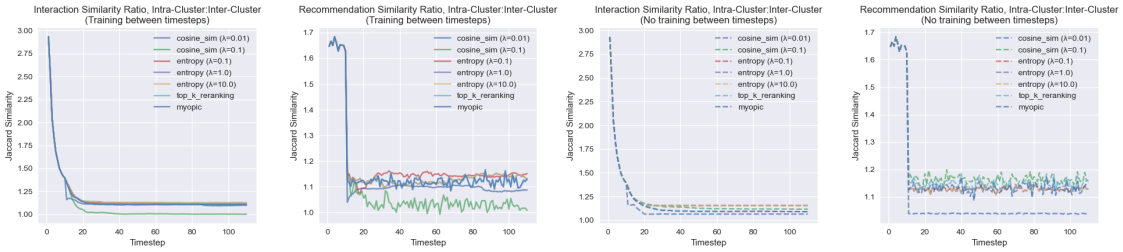
```
        #     name += f" (Lambda: {parameters_df.loc[i, 'Lambda']})"
        legend_names.append(name)

        if i < len(parameters)/2:

            ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
            line, = ax[idxs[0]].plot(ts, ratio_intrxn_sim, label=name,␣
↪alpha=alpha)#, color=colors(i))

            ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
            ax[idxs[1]].plot(ts, ratio_rec_sim, label=name, alpha=alpha)#,␣
↪color=colors(i))

        else:
            ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
            line, = ax[idxs[2]].plot(ts, ratio_intrxn_sim, label=name,␣
↪alpha=alpha, linestyle='--')#, color=colors(i))

            ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
            ax[idxs[3]].plot(ts, ratio_rec_sim, label=name, alpha=alpha,␣
↪linestyle='--')#, color=colors(i))

        legend_lines.append(line)

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

    title_suffix = '(Training between timesteps)'

    ax[idxs[0]].set_title(f"Interaction Similarity Ratio, Intra-Cluster:
↪Inter-Cluster\n{title_suffix}")
    ax[idxs[0]].set_ylabel('Jaccard Similarity')

    ax[idxs[1]].set_title(f"Recommendation Similarity Ratio, Intra-Cluster:
↪Inter-Cluster\n{title_suffix}")
    ax[idxs[1]].set_ylabel('Jaccard Similarity')

    title_suffix = '(No training between timesteps)'
```

```
    ax[idxs[2]].set_title(f"Interaction Similarity Ratio, Intra-Cluster:
↪Inter-Cluster\n{title_suffix}")
    ax[idxs[2]].set_ylabel('Jaccard Similarity')

    ax[idxs[3]].set_title(f"Recommendation Similarity Ratio, Intra-Cluster:
↪Inter-Cluster\n{title_suffix}")
    ax[idxs[3]].set_ylabel('Jaccard Similarity')

    # fig.legend(legend_lines,
    #            legend_names,
    #            loc='upper center',
    #            fontsize=12,
    #            frameon=False,
    #            ncol=5,
    #            bbox_to_anchor=(.5, 1.02))

    for i in idxs:
        ax[i].legend(loc='upper right')

    plt.savefig('figures/train_vs_no_train_homogen_analysis.png')

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

[ ]:
```
paths = [
    ('cosine_sim (=0.01)', retrain_paths['cosine_sim (=0.01)']),
    ('cosine_sim (=0.1)', retrain_paths['cosine_sim (=0.1)']),
    ('entropy (=0.1)', retrain_paths['entropy (=0.1)']),
    ('entropy (=1.0)', retrain_paths['entropy (=1.0)']),
    # ('entropy (=10.0)', retrain_paths['entropy (=10.0)']),
    ('top_k_reranking', retrain_paths['top_k_reranking']),
    ('myopic', retrain_paths['myopic']),

    ('cosine_sim (=0.01)', no_retrain_paths['cosine_sim (=0.01)']),
    ('cosine_sim (=0.1)', no_retrain_paths['cosine_sim (=0.1)']),
    ('entropy (=0.1)', no_retrain_paths['entropy (=0.1)']),
    ('entropy (=1.0)', no_retrain_paths['entropy (=1.0)']),
    # ('entropy (=10.0)', no_retrain_paths['entropy (=10.0)']),
    ('top_k_reranking', no_retrain_paths['top_k_reranking']),
    ('myopic', no_retrain_paths['myopic'])
]

# paths
dfs_homogen = []
for i in paths:
    df = pd.read_csv(i[1])
```

```
    dfs_homogen.append(df)
```

[ ]:
```
plot_ratios1(dfs_homogen, paths)
```



[ ]:
```
def plot_ratios2(dfs, parameters):
    fig, ax = plt.subplots(2, 4, figsize=(25, 10))
    fig.tight_layout(pad=5.0)
    colors = plt.get_cmap('tab10')
    alpha = 0.75

    # plot rec_similarity with timesteps on x axis
    idxs = [(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3)]
    # idxs = [0,1,2,3]
    legend_lines, legend_names = [], []

    for i, df in enumerate(dfs):
        ts = df['timesteps']
        name = parameters[i][0]
        # if not np.isnan(parameters_df.loc[i, 'Lambda']):
        #     name += f" (Lambda: {parameters_df.loc[i, 'Lambda']})"
        legend_names.append(name)
        # , linestyle='--'
        if i < 2:
            ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
            line, = ax[idxs[0]].plot(ts, ratio_intrxn_sim, label=name,␣
↪alpha=alpha)#, color=colors(i))

            ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
            ax[idxs[4]].plot(ts, ratio_rec_sim, label=name, alpha=alpha)#,␣
↪color=colors(i))
        elif i < 4:
```

```
            ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
            line, = ax[idxs[0]].plot(ts, ratio_intrxn_sim, label=name,␣
↪alpha=alpha, linestyle='--')#, color=colors(i))

            ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
            ax[idxs[4]].plot(ts, ratio_rec_sim, label=name, alpha=alpha,␣
↪linestyle='--')#, color=colors(i))

        elif i < 6:
            ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
            line, = ax[idxs[1]].plot(ts, ratio_intrxn_sim, label=name,␣
↪alpha=alpha)#, color=colors(i))

            ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
            ax[idxs[5]].plot(ts, ratio_rec_sim, label=name, alpha=alpha)#,␣
↪color=colors(i))
        elif i < 8:
            ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
            line, = ax[idxs[1]].plot(ts, ratio_intrxn_sim, label=name,␣
↪alpha=alpha, linestyle='--')#, color=colors(i))

            ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
            ax[idxs[5]].plot(ts, ratio_rec_sim, label=name, alpha=alpha,␣
↪linestyle='--')#, color=colors(i))

        elif i < 9:
            ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
            line, = ax[idxs[2]].plot(ts, ratio_intrxn_sim, label=name,␣
↪alpha=alpha)#, color=colors(i))

            ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
            ax[idxs[6]].plot(ts, ratio_rec_sim, label=name, alpha=alpha)#,␣
↪color=colors(i))
        elif i < 10:
```

```
            ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
            line, = ax[idxs[2]].plot(ts, ratio_intrxn_sim, label=name,␣
↪alpha=alpha, linestyle='--')#, color=colors(i))

            ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
            ax[idxs[6]].plot(ts, ratio_rec_sim, label=name, alpha=alpha,␣
↪linestyle='--')#, color=colors(i))

        elif i < 11:
            ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
            line, = ax[idxs[3]].plot(ts, ratio_intrxn_sim, label=name,␣
↪alpha=alpha)#, color=colors(i))

            ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
            ax[idxs[7]].plot(ts, ratio_rec_sim, label=name, alpha=alpha)#,␣
↪color=colors(i))
        else:
            ratio_intrxn_sim = np.
↪divide(df['intra_cluster_interaction_similarity'],␣
↪df['inter_cluster_interaction_similarity'])
            line, = ax[idxs[3]].plot(ts, ratio_intrxn_sim, label=name,␣
↪alpha=alpha, linestyle='--')#, color=colors(i))

            ratio_rec_sim = np.divide(df['intra_cluster_rec_similarity'],␣
↪df['inter_cluster_rec_similarity'])
            ax[idxs[7]].plot(ts, ratio_rec_sim, label=name, alpha=alpha,␣
↪linestyle='--')#, color=colors(i))

        print(i, "-", name)

        legend_lines.append(line)

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

    prefix = "cosine_sim"
    ax[idxs[0]].set_title(f"{prefix}\nInteraction Similarity Ratio,␣
↪Intra-Cluster:Inter-Cluster")
    ax[idxs[0]].set_ylabel('Jaccard Similarity')
```

```
    ax[idxs[4]].set_title(f"{prefix}\nRecommendation Similarity Ratio,␣
↪Intra-Cluster:Inter-Cluster")
    ax[idxs[4]].set_ylabel('Jaccard Similarity')

    prefix = "entropy"
    ax[idxs[1]].set_title(f"{prefix}\nInteraction Similarity Ratio,␣
↪Intra-Cluster:Inter-Cluster")
    ax[idxs[1]].set_ylabel('Jaccard Similarity')

    ax[idxs[5]].set_title(f"{prefix}\nRecommendation Similarity Ratio,␣
↪Intra-Cluster:Inter-Cluster")
    ax[idxs[5]].set_ylabel('Jaccard Similarity')

    prefix = "top_k_reranking"
    ax[idxs[2]].set_title(f"{prefix}\nInteraction Similarity Ratio,␣
↪Intra-Cluster:Inter-Cluster")
    ax[idxs[2]].set_ylabel('Jaccard Similarity')

    ax[idxs[6]].set_title(f"{prefix}\nRecommendation Similarity Ratio,␣
↪Intra-Cluster:Inter-Cluster")
    ax[idxs[6]].set_ylabel('Jaccard Similarity')

    prefix = "myopic"
    ax[idxs[3]].set_title(f"{prefix}\nInteraction Similarity Ratio,␣
↪Intra-Cluster:Inter-Cluster")
    ax[idxs[3]].set_ylabel('Jaccard Similarity')

    ax[idxs[7]].set_title(f"{prefix}\nRecommendation Similarity Ratio,␣
↪Intra-Cluster:Inter-Cluster")
    ax[idxs[7]].set_ylabel('Jaccard Similarity')

    # fig.legend(legend_lines,
    #            legend_names,
    #            loc='upper center',
    #            fontsize=12,
    #            frameon=False,
    #            ncol=5,
    #            bbox_to_anchor=(.5, 1.02))

    for i in idxs:
        ax[i].legend(loc='upper right')

    plt.savefig('figures/train_vs_no_train_homogen_analysis.png')
```

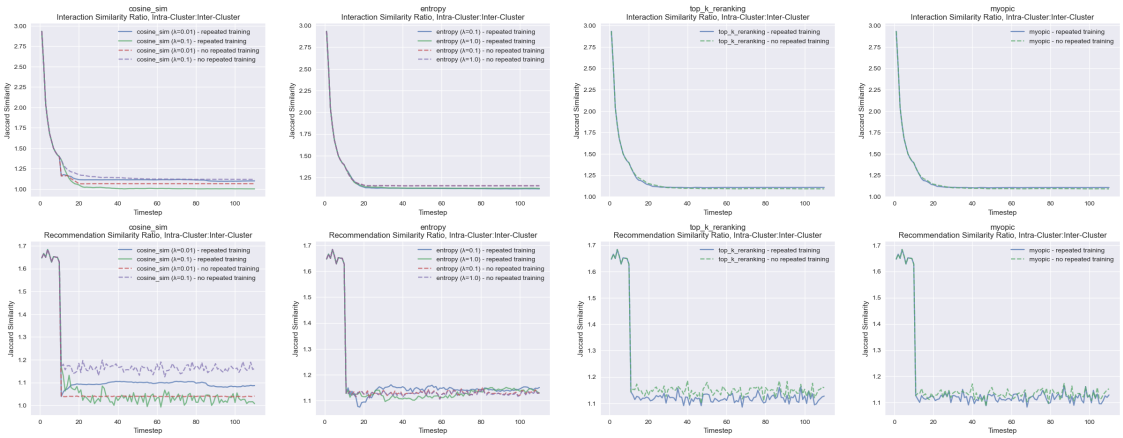
```
    # fig.legend(legend_lines, legend_names)#, loc='upper center', fontsize=14,␣
↪frameon=False, ncol=5, bbox_to_anchor=(.5, 1.05))
    # plt.legend()
```

[ ]:
```
paths = [
    ('cosine_sim (=0.01) - repeated training', retrain_paths['cosine_sim (=0.
↪01)']),
    ('cosine_sim (=0.1) - repeated training', retrain_paths['cosine_sim (=0.
↪1)']),
    ('cosine_sim (=0.01) - no repeated training', no_retrain_paths['cosine_sim␣
↪(=0.01)']),
    ('cosine_sim (=0.1) - no repeated training', no_retrain_paths['cosine_sim␣
↪(=0.1)']),

    ('entropy (=0.1) - repeated training', retrain_paths['entropy (=0.1)']),
    ('entropy (=1.0) - repeated training', retrain_paths['entropy (=1.0)']),
    # ('entropy (=10.0) - repeated training', retrain_paths['entropy (=10.
↪0)']),
    ('entropy (=0.1) - no repeated training', no_retrain_paths['entropy (=0.
↪1)']),
    ('entropy (=1.0) - no repeated training', no_retrain_paths['entropy (=1.
↪0)']),
    # ('entropy (=10.0) - no repeated training', no_retrain_paths['entropy␣
↪(=10.0)']),

    ('top_k_reranking - repeated training', retrain_paths['top_k_reranking']),
    ('top_k_reranking - no repeated training',␣
↪no_retrain_paths['top_k_reranking']),

    ('myopic - repeated training', retrain_paths['myopic']),
    ('myopic - no repeated training', no_retrain_paths['myopic'])
]

# paths
dfs_homogen = []
for i in paths:
    df = pd.read_csv(i[1])
    dfs_homogen.append(df)
```

[ ]:
```
plot_ratios2(dfs_homogen, paths)
```

```
0 - cosine_sim (=0.01) - repeated training
1 - cosine_sim (=0.1) - repeated training
2 - cosine_sim (=0.01) - no repeated training
3 - cosine_sim (=0.1) - no repeated training
4 - entropy (=0.1) - repeated training
5 - entropy (=1.0) - repeated training
```

```
6 - entropy ( =0.1) - no repeated training
7 - entropy ( =1.0) - no repeated training
8 - top_k_reranking - repeated training
9 - top_k_reranking - no repeated training
10 - myopic - repeated training
11 - myopic - no repeated training
```



4 Plotting difference in ratios

[ ]:
```python
def plot_ratio_diff(dfs, parameters):
    fig, ax = plt.subplots(1, 2, figsize=(20, 10))
    fig.tight_layout(pad=5.0)
    colors = plt.get_cmap('tab10')
    alpha = 0.75

    # plot rec_similarity with timesteps on x axis
    # idxs = [(0,0), (0,2)]
    # idxs = [0,1,2,3]
    legend_lines, legend_names = [], []

    # names = ['cosine_sim ( =0.01)', 'cosine_sim ( =0.1)', 'entropy ( =0.1)',␣
↪'entropy ( =1.0)', 'top_k_reranking', 'myopic']
    name_idx = 0
    # Ploting 'cosine_sim ( =0.01)'

    ts = dfs[0]['timesteps']

    for i in range(0, len(dfs), 2):

        print(i)
        offset = 1
```

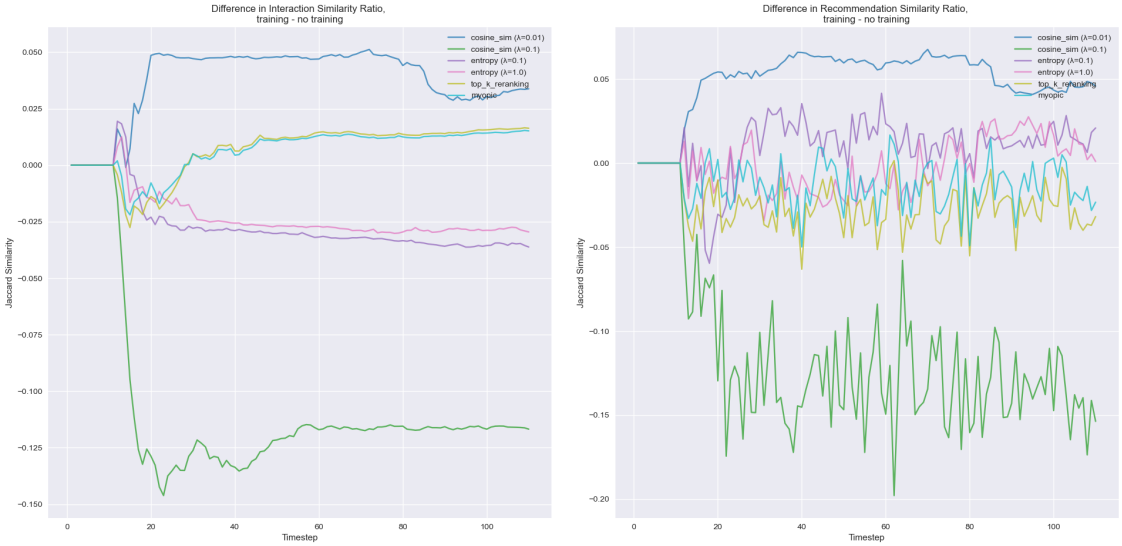
```
        print(dfs[i]['model'][0])
        print(dfs[i+offset]['model'][0])

        train_ratio = (np.
↪divide(dfs_homogen[i]['intra_cluster_interaction_similarity'],␣
↪dfs_homogen[i]['inter_cluster_interaction_similarity']))
        no_train_ratio = (np.
↪divide(dfs_homogen[i+offset]['intra_cluster_interaction_similarity'],␣
↪dfs_homogen[i+offset]['inter_cluster_interaction_similarity']))
        ratio_diff = train_ratio - no_train_ratio

        line, = ax[0].plot(ts, ratio_diff, label=parameters[name_idx],␣
↪alpha=alpha, color=colors(i))

        train_ratio = (np.
↪divide(dfs_homogen[i]['intra_cluster_rec_similarity'],␣
↪dfs_homogen[i]['inter_cluster_rec_similarity']))
        no_train_ratio = (np.
↪divide(dfs_homogen[i+offset]['intra_cluster_rec_similarity'],␣
↪dfs_homogen[i+offset]['inter_cluster_rec_similarity']))
        ratio_diff = train_ratio - no_train_ratio

        ax[1].plot(ts, ratio_diff, label=parameters[name_idx], alpha=alpha,␣
↪color=colors(i))

        name_idx += 1

        legend_lines.append(line)

    for a in ax:
            a.set_xlabel('Timestep')
            a.legend(loc='upper right')

    ax[0].set_title(f"Difference in Interaction Similarity Ratio, \ntraining -␣
↪no training")#\n{title_suffix}")
    ax[0].set_ylabel('Jaccard Similarity')

    ax[1].set_title(f"Difference in Recommendation Similarity Ratio, \ntraining␣
↪- no training")#\n{title_suffix}")
    ax[1].set_ylabel('Jaccard Similarity')

    plt.savefig('figures/ratio_difference_homogen_analysis.png')
```

[ ]:
```
paths = [
    ('cosine_sim ( =0.01) - repeated training', retrain_paths['cosine_sim ( =0.
 ↪01)']),
    ('cosine_sim ( =0.01) - no repeated training', no_retrain_paths['cosine_sim␣
 ↪( =0.01)']),
    ('cosine_sim ( =0.1) - repeated training', retrain_paths['cosine_sim ( =0.
 ↪1)']),
    ('cosine_sim ( =0.1) - no repeated training', no_retrain_paths['cosine_sim␣
 ↪( =0.1)']),

    ('entropy ( =0.1) - repeated training', retrain_paths['entropy ( =0.1)']),
    ('entropy ( =0.1) - no repeated training', no_retrain_paths['entropy ( =0.
 ↪1)']),
    ('entropy ( =1.0) - repeated training', retrain_paths['entropy ( =1.0)']),
    ('entropy ( =1.0) - no repeated training', no_retrain_paths['entropy ( =1.
 ↪0)']),

    ('top_k_reranking - repeated training', retrain_paths['top_k_reranking']),
    ('top_k_reranking - no repeated training',␣
 ↪no_retrain_paths['top_k_reranking']),

    ('myopic - repeated training', retrain_paths['myopic']),
    ('myopic - no repeated training', no_retrain_paths['myopic'])
]

# paths
dfs_homogen = []
for i in paths:
    df = pd.read_csv(i[1])
    dfs_homogen.append(df)
```

[ ]:
```
names = ['cosine_sim ( =0.01)', 'cosine_sim ( =0.1)', 'entropy ( =0.1)',␣
 ↪'entropy ( =1.0)', 'top_k_reranking', 'myopic']

plot_ratio_diff(dfs_homogen, names)
```

```
0
cosine_sim
cosine_sim
2
cosine_sim
cosine_sim
4
entropy
entropy
6
entropy
```

```
entropy
8
top_k_reranking
top_k_reranking
10
myopic
myopic
```



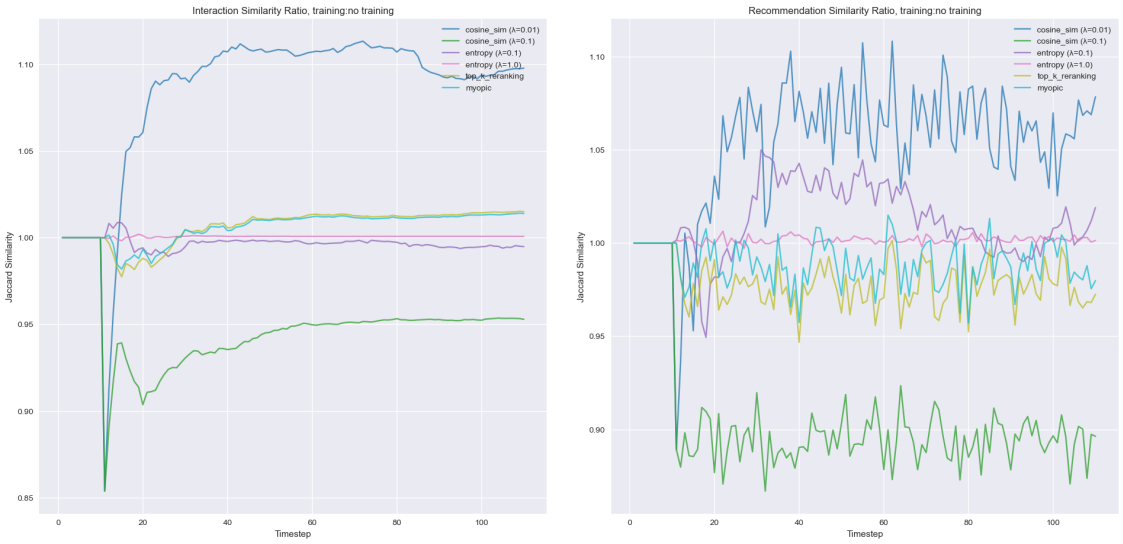
[ ]:

[ ]:
```
# list = []

# # fig, ax = plt.subplots(1, 2, figsize=(20, 10))
# # fig.tight_layout(pad=5.0)
# # colors = plt.get_cmap('tab10')

# # plot rec_similarity with timesteps on x axis
# # idxs = [(0,0), (0,1), (1,0), (1,1), (2,0), (2,1), (3,0), (3,1)]
# idxs = [0,1]
# legend_lines, legend_names = [], []

# ts = dfs_homogen[0]['timesteps']

# alpha = 0.75

# names = ['cosine_sim (=0.01)', 'cosine_sim (=0.1)', 'entropy (=0.1)',␣
  ↪'entropy (=1.0)', 'top_k_reranking', 'myopic']
# name_count = 0
```

```
# for i in range(0, len(paths), 2):

#     # print(i)
#     # print(paths[i])
#     # if i < 7:
#     #     offset = 2
#     # else:
#     #     break
#     # print(offset, "\n")
#     print(i)
#     offset = 1
#     print(dfs_homogen[i]['model'][0])
#     print(dfs_homogen[i+offset]['model'][0])

#     ratio1 = (np.
 ↪divide(dfs_homogen[i]['intra_cluster_interaction_similarity'],␣
 ↪dfs_homogen[i]['inter_cluster_interaction_similarity']))
#     ratio2 = (np.
 ↪divide(dfs_homogen[i+offset]['intra_cluster_interaction_similarity'],␣
 ↪dfs_homogen[i+offset]['inter_cluster_interaction_similarity']))
#     ratio_ratios = np.divide(ratio1, ratio2)

#     line, = ax[idxs[0]].plot(ts, ratio_ratios, label=names[name_count],␣
 ↪alpha=alpha, color=colors(i))

#     ratio1 = (np.divide(dfs_homogen[i]['intra_cluster_rec_similarity'],␣
 ↪dfs_homogen[i]['inter_cluster_rec_similarity']))
#     ratio2 = (np.
 ↪divide(dfs_homogen[i+offset]['intra_cluster_rec_similarity'],␣
 ↪dfs_homogen[i+offset]['inter_cluster_rec_similarity']))
#     ratio_ratios = np.divide(ratio1, ratio2)

#     ax[idxs[1]].plot(ts, ratio_ratios, label=names[name_count], alpha=alpha,␣
 ↪color=colors(i))

#     name_count += 1

#     legend_lines.append(line)

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

# title_suffix = '(Training between timesteps)'

# ax[idxs[0]].set_title(f"Interaction Similarity Ratio, training:no␣
 ↪training")#\n{title_suffix}")
```

```
# ax[idxs[0]].set_ylabel('Jaccard Similarity')

# ax[idxs[1]].set_title(f"Recommendation Similarity Ratio, training:no␣
  ↪training")#\n{title_suffix}")
# ax[idxs[1]].set_ylabel('Jaccard Similarity')

# for i in idxs:
#     ax[i].legend(loc='upper right')

# plt.savefig('figures/ratio_of_ratios_homogen_analysis.png')
```

```
0
cosine_sim
cosine_sim
2
cosine_sim
cosine_sim
4
entropy
entropy
6
entropy
entropy
8
top_k_reranking
top_k_reranking
10
myopic
myopic
```



[ ]:
```
# list = []

# fig, ax = plt.subplots(1, 2, figsize=(20, 10))
# fig.tight_layout(pad=5.0)
# colors = plt.get_cmap('tab10')

# # plot rec_similarity with timesteps on x axis
# # idxs = [(0,0), (0,1), (1,0), (1,1), (2,0), (2,1), (3,0), (3,1)]
# idxs = [0,1]
# legend_lines, legend_names = [], []

# ts = dfs_homogen[0]['timesteps']

# alpha = 0.75

# names = ['cosine_sim (=0.01)', 'cosine_sim (=0.1)', 'entropy (=0.1)',␣
 ↪'entropy (=1.0)', 'top_k_reranking', 'myopic']
# name_count = 0

# for i in range(0, len(paths), 2):

#     # print(i)
#     # print(paths[i])
#     # if i < 7:
#     #     offset = 2
#     # else:
#     #     break
#     # print(offset, "\n")
#     print(i)
#     offset = 1
#     print(dfs_homogen[i]['model'][0])
#     print(dfs_homogen[i+offset]['model'][0])

#     train_ratio = (np.
 ↪divide(dfs_homogen[i]['intra_cluster_interaction_similarity'],␣
 ↪dfs_homogen[i]['inter_cluster_interaction_similarity']))
#     no_train_ratio = (np.
 ↪divide(dfs_homogen[i+offset]['intra_cluster_interaction_similarity'],␣
 ↪dfs_homogen[i+offset]['inter_cluster_interaction_similarity']))
#     ratio_ratios = np.divide(ratio1, ratio2)

#     line, = ax[idxs[0]].plot(ts, ratio_ratios, label=names[name_count],␣
 ↪alpha=alpha, color=colors(i))

#     ratio1 = (np.divide(dfs_homogen[i]['intra_cluster_rec_similarity'],␣
 ↪dfs_homogen[i]['inter_cluster_rec_similarity']))
```

```
#     ratio2 = (np.
 ↪divide(dfs_homogen[i+offset]['intra_cluster_rec_similarity'],␣
 ↪dfs_homogen[i+offset]['inter_cluster_rec_similarity']))
#     ratio_ratios = np.divide(ratio1, ratio2)

#     ax[idxs[1]].plot(ts, ratio_ratios, label=names[name_count], alpha=alpha,␣
 ↪color=colors(i))

#     name_count += 1

#     legend_lines.append(line)

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

# title_suffix = '(Training between timesteps)'

# ax[idxs[0]].set_title(f"Interaction Similarity Ratio, training:no␣
 ↪training")#\n{title_suffix}")
# ax[idxs[0]].set_ylabel('Jaccard Similarity')

# ax[idxs[1]].set_title(f"Recommendation Similarity Ratio, training:no␣
 ↪training")#\n{title_suffix}")
# ax[idxs[1]].set_ylabel('Jaccard Similarity')

# for i in idxs:
#     ax[i].legend(loc='upper right')

# plt.savefig('figures/ratio_of_ratios_homogen_analysis.png')
```

[ ]:

[ ]: