Report Experiment ClayRS Framework

SWAP research group UniBa

09-01-2024

This LaTex document was generated automatically from yaml files for the purpose of replicability of experiments done with ClayRS, it contains information about the experiment that has been conducted and the results obtained. The report is divided in 3 principal section dedicated each one for the 3 principal module of the ClayRS framework and a conclusion section to highlights what have been achieved from the experiment.

1 Content Analyzer Module

The content analyzer module will deal with raw source document or more in general data which could be video or audio data and give a representation of these data which will be used by the other two module. The text data source could be represented with exogenous technique or with a specified representation and each field given could be treated with preprocessing techniques and postprocessing technique. In this experiment the following techniques have been used on specific field in order to achieve the representation wanted:

plot0 has been represented with the following techniques:

- `OriginalData` used as content representation technique with following parameters:
  - `dtype: <class 'str'>`

No preprocessing techniques have been used to preprocess data plot0 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data plot0 in this experiment.

plot1 has been represented with the following techniques:

- `WhooshTfIdf` used as content representation technique with following parameters:

- `Spacy` used as preprocessing technique with following settings:
  - `lemmatization: True`
  - `model: en_core_web_sm`
  - `named_entity_recognition: False`
  - `new_stopwords: None`
  - `not_stopwords: None`
  - `remove_punctuation: False`
  - `stopwords_removal: True`
  - `strip_multiple_whitespace: True`
  - `url_tagging: False`

No postprocessing techniques have been used on the data plot1 in this experiment.

plot2 has been represented with the following techniques:

- `SkLearnTfIdf` used as content representation technique with following parameters:
  - `binary: False`
  - `dtype: <class 'numpy.float64'>`
  - `max_df: 1.0`
  - `max_features: None`
  - `min_df: 1`
  - `norm: l2`
  - `smooth_idf: True`
  - `sublinear_tf: False`
  - `use_idf: True`
  - `vocabulary: None`

- `Ekphrasis` used as preprocessing technique with following settings:
  - `all_caps_tag: None`
  - `annotate: None`
  - `corrector: None`
  - `dicts: None`
  - `normalize: None`
  - `omit: None`
  - `segmentation: False`
  - `segmenter: None`

  - spell_correct_elong: True
  - spell_correction: True
  - tokenizer: <bound method SocialTokenizer.tokenize of <ekphrasis.classes.tok
  - unpack_contractions: False
  - unpack_hashtags: False

- NLTK used as preprocessing technique with following settings:
  - lang: english
  - lemmatization: True
  - pos_tag: False
  - remove_punctuation: False
  - stemming: False
  - stopwords_removal: True
  - strip_multiple_whitespace: True
  - url_tagging: False

No postprocessing techniques have been used on the data plot2 in this experiment.

plot3 has been represented with the following techniques:

- WordEmbeddingTechnique used as content representation technique with following parameters:
  - embedding_source: Gensim glove-twitter-25

No preprocessing techniques have been used to preprocess data plot3 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data plot3 in this experiment.

plot4 has been represented with the following techniques:

- `SentenceEmbeddingTechnique` used as content representation technique with following parameters:
  - `embedding_source: Sbert paraphrase-distilroberta-base-v1`

No preprocessing techniques have been used to preprocess data plot4 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data plot4 in this experiment.

plot5 has been represented with the following techniques:

- `DocumentEmbeddingTechnique` used as content representation technique with following parameters:
  - `embedding_source: Sbert paraphrase-distilroberta-base-v1`

No preprocessing techniques have been used to preprocess data plot5 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data plot5 in this experiment.

plot6 has been represented with the following techniques:

- `Word2SentenceEmbedding` used as content representation technique with following parameters:
  - `combining_technique: Centroid`
  - `embedding_source: Gensim glove-twitter-25`

No preprocessing techniques have been used to preprocess data plot6 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data plot6 in this experiment.

plot7 has been represented with the following techniques:

- `Word2DocEmbedding` used as content representation technique with following parameters:
  - `combining_technique: Centroid`
  - `embedding_source: Gensim glove-twitter-25`

No preprocessing techniques have been used to preprocess data plot7 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data plot7 in this experiment.

plot8 has been represented with the following techniques:

- `Sentence2DocEmbedding` used as content representation technique with following parameters:
  - `combining_technique: Centroid`
  - `embedding_source: Sbert paraphrase-distilroberta-base-v1`

No preprocessing techniques have been used to preprocess data plot8 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data plot8 in this experiment.

plot9 has been represented with the following techniques:

- `PyWSDSynsetDocumentFrequency` used as content representation technique with following parameters:

No preprocessing techniques have been used to preprocess data plot9 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data plot9 in this experiment.

plot10 has been represented with the following techniques:

- `OriginalData` used as content representation technique with following parameters:
  - `dtype: <class 'str'>`

No preprocessing techniques have been used to preprocess data plot10 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data plot10 in this experiment.

plot11 has been represented with the following techniques:

- `OriginalData` used as content representation technique with following parameters:
  - `dtype: <class 'str'>`

No preprocessing techniques have been used to preprocess data plot11 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data plot11 in this experiment.

idx0 has been represented with the following techniques:

- `FromNPY` used as content representation technique with following parameters:
  - `npy_file_path: report_stuff\report.npy`

No preprocessing techniques have been used to preprocess data idx0 has been represented with the following techniques: during the experiment.

No postprocessing techniques have been used on the data idx0 in this experiment.

video_path0 has been represented with the following techniques:

- `SkImageHogDescriptor` used as content representation technique with following parameters:
  - `block_norm: L2-Hys`
  - `cells_per_block: [3, 3]`
  - `contents_dirs: contents_dirs`

  - flatten: False
  - max_retries: 5
  - max_timeout: 2
  - max_workers: 0
  - orientations: 9
  - pixels_per_cell: [8, 8]
  - time_tuple: [0, None]
  - transform_sqrt: False

- TorchUniformTemporalSubSampler used as preprocessing technique with following settings:
  - num_samples: 8

No postprocessing techniques have been used on the data video_path0 in this experiment.

video_path1 has been represented with the following techniques:

- MFCC used as content representation technique with following parameters:
  - contents_dirs: contents_dirs
  - dct_type: 2
  - flatten: False
  - log_mels: False
  - max_retries: 5
  - max_timeout: 2

  - max_workers: 0
  - mean: False
  - melkwargs: None
  - n_mfcc: 40
  - norm: ortho
  - time_tuple: [0, None]

- ConvertToMono used as preprocessing technique with following settings:
- TorchResample used as preprocessing technique with following settings:
  - beta: None
  - lowpass_filter_width: 6
  - new_freq: 16000
  - resampling_method: sinc_interp_hann
  - rolloff: 0.99

On th field video_path1 have been applied the following postprocessing techniques:

- 0 :
- 1 :
  - FVGMM:
    * alpha: 0.5
    * covariance_type: diag
    * improved: True
    * init_params: kmeans
    * max_iter: 100

        * means_init: None
        * n_components: 2
        * n_init: 1
        * precisions_init: None
        * random_state: None
        * reg_covar: 1e-06
        * tol: 0.001
        * verbose: 0
        * verbose_interval: 10
        * warm_start: False
        * weights_init: None
        * with_mean: False
        * with_std: False

- 2 :
    - VLADGMM:
        * alpha: 0.5
        * covariance_type: diag
        * improved: False
        * init_params: kmeans
        * max_iter: 100
        * means_init: None
        * n_components: 2
        * n_init: 1
        * precisions_init: None
        * random_state: None
        * reg_covar: 1e-06
        * tol: 0.001
        * verbose: 0
        * verbose_interval: 10
        * warm_start: False
        * weights_init: None
        * with_mean: False
        * with_std: False

- 3 :

  - SkLearnPCA:
    * copy: True
    * iterated_power: auto
    * n_components: 1
    * random_state: None
    * svd_solver: auto
    * tol: 0.0
    * whiten: False

video_path2 has been represented with the following techniques:

- VGGISH used as content representation technique with following parameters:
  - apply_on_output: None
  - batch_size: 1
  - contents_dirs: contents_dirs
  - device: cuda:0
  - feature_layer: embedding_network.5
  - flatten: True
  - max_retries: 5
  - max_timeout: 2
  - max_workers: 0
  - time_tuple: [0, None]

- ConvertToMono used as preprocessing technique with following settings:

- TorchResample used as preprocessing technique with following settings:
  - beta: None
  - lowpass_filter_width: 6
  - new_freq: 16000
  - resampling_method: sinc_interp_hann
  - rolloff: 0.99

On th field video_path2 have been applied the following postprocessing techniques:

- 0 :
- 1 :
  - FVGMM:
    * alpha: 0.5
    * covariance_type: diag
    * improved: True
    * init_params: kmeans
    * max_iter: 100
    * means_init: None
    * n_components: 2
    * n_init: 1
    * precisions_init: None
    * random_state: None
    * reg_covar: 1e-06
    * tol: 0.001
    * verbose: 0
    * verbose_interval: 10
    * warm_start: False
    * weights_init: None
    * with_mean: False

        * with_std: False

- 2 :
    - VLADGMM:
        * alpha: 0.5
        * covariance_type: diag
        * improved: False
        * init_params: kmeans
        * max_iter: 100
        * means_init: None
        * n_components: 2
        * n_init: 1
        * precisions_init: None
        * random_state: None
        * reg_covar: 1e-06
        * tol: 0.001
        * verbose: 0
        * verbose_interval: 10
        * warm_start: False
        * weights_init: None
        * with_mean: False
        * with_std: False
- 3 :
    - SkLearnPCA:
        * copy: True
        * iterated_power: auto
        * n_components: 1
        * random_state: None
        * svd_solver: auto
        * tol: 0.0
        * whiten: False

video_path3 has been represented with the following techniques:

- `PytorchImageModels` used as content representation technique with following parameters:
  - `apply_on_output: None`
  - `batch_size: 4`
  - `contents_dirs: contents_dirs`
  - `custom_weights_path: None`
  - `device: cuda:0`
  - `feature_layer: head_drop`
  - `flatten: True`
  - `max_retries: 5`
  - `max_timeout: 2`
  - `max_workers: 0`
  - `model_name: densenet161`
  - `num_classes: None`
  - `time_tuple: [0, 10]`
  - `use_default_transforms: False`

- `CenterCrop` used as preprocessing technique with following settings:
  - `size: [224, 224]`
- `Lambda` used as preprocessing technique with following settings:
- `Normalize` used as preprocessing technique with following settings:
  - `mean: [0.485, 0.456, 0.406]`
  - `std: [0.229, 0.224, 0.225]`
- `Resize` used as preprocessing technique with following settings:

  - antialias: None
  - interpolation: bilinear
  - max_size: None
  - size: [256, 256]

- TorchUniformTemporalSubSampler used as preprocessing technique with following settings:
  - num_samples: 8

On th field video_path3 have been applied the following postprocessing techniques:

- 0 :
- 1 :
  - FVGMM:
    * alpha: 0.5
    * covariance_type: diag
    * improved: True
    * init_params: kmeans
    * max_iter: 100
    * means_init: None
    * n_components: 2
    * n_init: 1
    * precisions_init: None
    * random_state: None
    * reg_covar: 1e-06
    * tol: 0.001
    * verbose: 0
    * verbose_interval: 10
    * warm_start: False
    * weights_init: None
    * with_mean: False

        * with_std: False

- 2 :
    - VLADGMM:
        * alpha: 0.5
        * covariance_type: diag
        * improved: False
        * init_params: kmeans
        * max_iter: 100
        * means_init: None
        * n_components: 2
        * n_init: 1
        * precisions_init: None
        * random_state: None
        * reg_covar: 1e-06
        * tol: 0.001
        * verbose: 0
        * verbose_interval: 10
        * warm_start: False
        * weights_init: None
        * with_mean: False
        * with_std: False
- 3 :
    - SkLearnPCA:
        * copy: True
        * iterated_power: auto
        * n_components: 1
        * random_state: None
        * svd_solver: auto
        * tol: 0.0
        * whiten: False

video_path4 has been represented with the following techniques:

- `TorchVisionVideoModels` used as content representation technique with following parameters:
  - `apply_on_output: None`
  - `batch_size: 4`
  - `contents_dirs: contents_dirs`
  - `device: cuda:0`
  - `feature_layer: avgpool`
  - `flatten: True`
  - `max_retries: 5`
  - `max_timeout: 2`
  - `max_workers: 0`
  - `mini_batch_size: 10`
  - `model_name: r2plus1d_18`
  - `time_tuple: [0, 10]`
  - `weights: DEFAULT`

- `CenterCrop` used as preprocessing technique with following settings:
  - `size: [112, 112]`
- `ClipSampler` used as preprocessing technique with following settings:
  - `num_clips: 5`
  - `num_frames_for_clip: 16`
  - `selection_strategy: sequential`
- `Lambda` used as preprocessing technique with following settings:
- `Normalize` used as preprocessing technique with following settings:

  - mean: [0.43216, 0.394666, 0.37645]
  - std: [0.22803, 0.22145, 0.216989]

- Resize used as preprocessing technique with following settings:
  - antialias: None
  - interpolation: bilinear
  - max_size: None
  - size: [128, 171]

On th field video_path4 have been applied the following postprocessing techniques:

- 0 :
- 1 :
  - FVGMM:
    * alpha: 0.5
    * covariance_type: diag
    * improved: True
    * init_params: kmeans
    * max_iter: 100
    * means_init: None
    * n_components: 2
    * n_init: 1
    * precisions_init: None
    * random_state: None
    * reg_covar: 1e-06
    * tol: 0.001
    * verbose: 0
    * verbose_interval: 10
    * warm_start: False
    * weights_init: None
    * with_mean: False

        * with_std: False

- 2 :
    - VLADGMM:
        * alpha: 0.5
        * covariance_type: diag
        * improved: False
        * init_params: kmeans
        * max_iter: 100
        * means_init: None
        * n_components: 2
        * n_init: 1
        * precisions_init: None
        * random_state: None
        * reg_covar: 1e-06
        * tol: 0.001
        * verbose: 0
        * verbose_interval: 10
        * warm_start: False
        * weights_init: None
        * with_mean: False
        * with_std: False
- 3 :
    - SkLearnPCA:
        * copy: True
        * iterated_power: auto
        * n_components: 1
        * random_state: None
        * svd_solver: auto
        * tol: 0.0
        * whiten: False

In this experiment the data used have been represented with exogenous representation, in particular the following techniques have been used:

- `DBPediaMappingTechnique` used as exogenous technique with following settings:
  - `label_field: dbpedia_label`
  - `max_timeout: 5`
  - `mode: only_retrieved_evaluated, entity type=dbo:Film`
  - `prop_as_uri: False`

2 Recommender System module: RecSys

The **RecSys module** allows to instantiate a recommender system and make it work on items and users serialized by the Content Analyzer module, despite this is also possible using other serialization made with other framework and give them as input to the recommender system and obtain score prediction or recommend items for the active user(s). In particular this module allows has to get some general statistics on the data used, the scheme used to split the data and train the recommender system and the settings belonging to the algorithm chosen.

2.1 Statistics on data used

In this experiment the statistics of the dataset used are reported in the following table: 1:

2.2 Partitioning techinque used

K-fold cross-validation is a technique used in machine learning to assess the performance of a predictive model. The basic idea is to divide the dataset into K subsets, or folds. The model is then trained K times, each time using

Parameter	Value
n_users	2
n_items	6
total_interactions	6
min_score	1.0
max_score	4.0
mean_score	2.66667
sparsity	0.5

Table 1: Table of the Interactions

K-1 folds for training and the remaining fold for validation. This process is repeated K times, with a different fold used as the validation set in each iteration.

The KFoldPartitioning has been used with the following setting:

The data has been shuffled before being split into batches. The partitioning technique has been executed with the following settings:

- number of splits: 2
- shuffle: True
- random state: None
- skip user error: True

2.3 Algorithm used for the recommender system

The framework of ClayRs allows to instantiate a recommender system in order to make list of recommendation, to achieve this target the system need to be based on a chosen algorithm that will work with the representation of data that we have. In this section we will analyse which algorithm has been used for the experiment and what are the settings given.

The recommender system is based on content, the algorithm used is `ClassifierRecommender` and the following are its settings:

- `classifier: SkKNN`
- `embedding_combiner: Centroid`
- `item_field: {'plot': ['tfidf_sk']}`
- `threshold: None`

The mode used is rank and the number of recommendations given is 10. The methodology used:

- `TestRatingsMethodology:`
  - `only_greater_eq: None`

3 Evaluation Module

The **EvalModel** which is the abbreviation for Evaluation Model has the task of evaluating a recommender system, using several state-of-the-art metrics, this allows to compare different recommender system and different algorithm of recommendation and find out which are the strength points and which the weak ones.

3.1 Metrics

During the experiment a bunch of formal metrics have been performed on the recommendation produced in order to evaluate the performance of the system. The metrics used are the followings:

3.1.1 Precision

In ClayRS, the Precision metric is calculated as such for the **single user**:

$$Precision_u = \frac{tp_u}{tp_u + fp_u}$$

Where:

- tp_u is the number of items which are in the recommendation list of the user and have a $\geq$ **2.6666666666666665**.
- fp_u is the number of items which are in the recommendation list of the user and have a rating $<$ **no relevant threshold used**.

In ClayRS, Precision needs those parameters: the **relevant_threshold**, is a parameter needed to discern relevant items and non-relevant items for every user. If not specified, the mean rating score of every user will be used, in this experiment it has been set to **None**.

3.1.2 Recall

The Recall metric is calculated as such for the **single user**:

$$Recall_u = \frac{tp_u}{tp_u + fn_u}$$

Where:

- tp_u is the number of items which are in the recommendation list of the user and have a rating >= **2.6666666666666665**.
- fn_u is the number of items which are not in the recommendation list of the user and have a rating >= **no relevant threshold used**.

In ClayRS, Recall needs those parameters: the **relevant_threshold**, is a parameter needed to discern relevant items and non-relevant items for every user. If not specified, the mean rating score of every user will be used, in this experiment it has been set to **None**.

3.1.3 FMeasure

The FMeasure metric combines Precision and Recall into a single metric. It is calculated as such for the **single user**:

$$FMeasure_u = (1 + \beta^2) \cdot \frac{P_u \cdot R_u}{(\beta^2 \cdot P_u) + R_u}$$

Where:

- P_u is the Precision calculated for the user **u**.
- R_u is the Recall calculated for the user **u**.
- β is a real factor which could weight differently Recall or Precision based on its value:
  - $\beta = 1$: Equally weight Precision and Recall.
  - $\beta > 1$: Weight Recall more.
  - $\beta < 1$: Weight Precision more.

A famous FMeasure is the F1 Metric, where $\beta = 1$, which basically is the harmonic mean of recall and precision:

$$F1_u = \frac{2 \cdot P_u \cdot R_u}{P_u + R_u}$$

The FMeasure metric is calculated as such for the entire system, depending on if **macro** average or **micro** average has been chosen:

$$FMeasure_{sys} - micro = (1 + \beta^2) \cdot \frac{P_u \cdot R_u}{(\beta^2 \cdot P_u) + R_u}$$

$$FMeasure_{sys} - macro = \frac{\sum_{u \in U} FMeasure_u}{|U|}$$

During the experiment the FMeasure has been calculated with $\beta = 1$ and the relevant threshold is **None**.

3.1.4 Precision@K

The Precision@K metric is calculated as such for the **single user**:

$$Precision@K_u = \frac{tp@K_u}{tp@K_u + fp@K_u}$$

Where:

- $tp@K_u$ is the number of items which are in the recommendation list of the user cutoff to the first K items and have a rating >= relevant threshold in its ground truth.
- $tp@K_u$ is the number of items which are in the recommendation list of the user cutoff to the first K items** and have a rating < relevant threshold in its ground truth.

And it is calculated as such for the entire system, depending on if **macro** average or **micro** average has been chosen:

$$Precision@K_{sys} - micro = \frac{\sum_{u \in U} tp@K_u}{\sum_{u \in U} tp@K_u + \sum_{u \in U} fp@K_u}$$

$$Precision@K_{sys} - macro = \frac{\sum_{u \in U} Precision@K_u}{|U|}$$

During the experiment Precision@K has been used with the following settings:

- **K: 5**
- **relevant threshold:None**
- **sys average: macro**

3.1.5 Recall@K

The Recall@K metric is calculated as such for the **single user**:

$$Recall@K_u = \frac{tp@K_u}{tp@K_u + fn@K_u}$$

Where:

- $tp@K_u$ is the number of items which are in the recommendation list of the user **cutoff to the first K items** and have a rating >= relevant threshold in its ground truth
- $tp@K_u$ is the number of items which are NOT in the recommendation list of the user **cutoff to the first K items** and have a rating >= relevant threshold in its ground truth

And it is calculated as such for the entire system, depending on if **macro** average or **micro** average has been chosen:

$$Recall@K_{sys} - micro = \frac{\sum_{u \in U} tp@K_u}{\sum_{u \in U} tp@K_u + \sum_{u \in U} fn@K_u}$$

$$Recall@K_{sys} - macro = \frac{\sum_{u \in U} Recall@K_u}{|U|}$$

During the experiment Recall@K has been used with the following settings:

- **K: 5**
- **relevant threshold: None**
- **sys average: macro**

3.1.6 FMeasure@K

The FMeasure@K metric combines Precision@K and Recall@K into a single metric. It is calculated as such for the **single user**:

$$FMeasure@K_u = (1 + \beta^2) \cdot \frac{P@K_u \cdot R@K_u}{(\beta^2 \cdot P@K_u) + R@K_u}$$

Where:

- $P@K_u$ is the Precision at K calculated for the user **u**.
- $R@K_u$ is the Recall at K calculated for the user **u**.

- β is a real factor which could weight differently Recall or Precision based on its value:
  - $\beta = 1$: Equally weight Precision and Recall.
  - $\beta > 1$: Weight Recall more.
  - $\beta < 1$: Weight Precision more.

A famous FMeasure@K is the F1@K Metric, where $\beta = 1$, which basically is the harmonic mean of recall and precision:

$$F1@K_u = \frac{2 \cdot P@K_u \cdot R@K_u}{P@K_u + R@K_u}$$

The FMeasure@K metric is calculated as such for the entire system, depending on if **macro** average or **micro** average has been chosen:

$$FMeasure@K_{sys} - micro = (1 + \beta^2) \cdot \frac{P@K_u \cdot R@K_u}{(\beta^2 \cdot P@K_u) + R@K_u}$$

$$FMeasure@K_{sys} - macro = \frac{\sum_{u \in U} FMeasure@K_u}{|U|}$$

During the experiment FMeasure@K has been used with the following settings:

- **K: 5**
- **β: 1**
- **relevant threshold: None**
- **sys average: micro**

3.1.7 R-Precision

The R-Precision metric is calculated as such for the **single user**:

$$R - Precision_u = \frac{tp@R_u}{tp@R_u + fp@R_u}$$

Where:

- R it's the number of relevant items for the user **u**.
- $tp@R_u$ is the number of items in the recommendation list of the user, up to the first R items, that have a rating $\geq$ the relevant threshold in its ground truth.
- $tp@R_u$ is the number of items in the recommendation list of the user, up to the first R items, that have a rating $<$ `relevant_threshold` in their ground truth.

And it is calculated as such for the entire system, depending on if **macro** average or **micro** average has been chosen:

$$Precision@R_{sys} - micro = \frac{\sum_{u \in U} tp@R_u}{\sum_{u \in U} tp@R_u + \sum_{u \in U} fp@R_u}$$

$$Precision@R_{sys} - macro = \frac{\sum_{u \in U} R - Precision_u}{|U|}$$

During the experiment R-Precision has been used with the following settings:

- **relevant threshold:None**
- **sys average: macro**

3.1.8 MSE

The MSE abbreviation Mean Squared Error metric is calculated as such for the **single user**:

$$MSE_u = \sum_{i \in T_u} \frac{(r_{u,i} - \hat{r}_{u,i})^2}{|T_u|}$$

Where:

- T_u is the test set of the user **u**.
- $r_{u,i}$ is the actual score give by user **u** to item **i**.
- $\hat{r}_{u,i}$ is the predicted score give by user **u** to item **i**.

It is calculated as such for the entire system:

$$MSE_{sys} = \sum_{u \in T} \frac{MSE_u}{|T|}$$

Where:

- T is the **test set**.
- MSE_u is the MSE calculated for user **u**.

There may be cases in which some items of the test set of the user could not be predicted *e.g. a CBRS was chosen and items were not present locally.* In those cases, the MSE_u formula becomes:

$$MSE_u = \sum_{i \in T_u} \frac{(r_{u,i} - \hat{r}_{u,i})^2}{|T_u| - unk}$$

Where:

- *unk* stay for unknown is the number of items of the user test set that could not be predicted.

If no items of the user test set have been predicted $|T_u| - unk = 0$, then:

$$MSE_u = NaN$$

3.1.9 RMSE

The RMSE (Root Mean Squared Error) metric is calculated as such for the **single user**:

$$RMSE_u = \sqrt{\sum_{i \in T_u} \frac{(r_{u,i} - \hat{r}_{u,i})^2}{|T_u|}}$$

Where:

- T_u is the test set of the user u.
- $r_{u,i}$ is the actual score give by user u to item i.
- $\hat{r}_{u,i}$ is the predicted score give by user u to item i.

It is calculated as such for the entire system:

$$RMSE_{sys} = \sum_{u \in T} \frac{RMSE_u}{|T|}$$

Where:

- T is the test set.
- $RMSE_u$ is the RMSE calculated for user u.

There may be cases in which some items of the test set of the user could not be predicted *e.g. a CBRS was chosen and items were not present locally, a methodology different from TestRatings was chosen.* In those cases, the $RMSE_u$ formula becomes:

$$RMSE_u = \sqrt{\sum_{i \in T_u} \frac{(r_{u,i} - \hat{r}_{u,i})^2}{|T_u| - unk}}$$

Where:

- *unk* (unknown) is the number of items of the user test set that could not be predicted.

If no items of the user test set have been predicted $|T_u| - unk = 0$, then:

$$RMSE_u = NaN$$

3.1.10 MAE

The MAE abbreviation of Mean Absolute Error metric is calculated as such for the **single user**:

$$MAE_u = \sum_{i \in T_u} \frac{|r_{u,i} - \hat{r}_{u,i}|}{|T_u|}$$

Where:

- T_u is the test set of the user u.
- $r_{u,i}$ is the actual score give by user u to item i.
- $\hat{r}_{u,i}$ is the predicted score give by user u to item i.

It is calculated as such for the entire system:

$$MAE_{sys} = \sum_{u \in T} \frac{MAE_u}{|T|}$$

Where:

- T is the test set.
- MAE_u is the MAE calculated for user u.

There may be cases in which some items of the test set of the user could not be predicted *e.g. a CBRS was chosen and items were not present locally, a methodology different from TestRatings was chosen.* In those cases, the MAE_u formula becomes:

$$MAE_u = \sum_{i \in T_u} \frac{|r_{u,i} - \hat{r}_{u,i}|}{|T_u| - unk}$$

Where:

- unk (unknown) is the number of items of the user test set that could not be predicted.

If no items of the user test set have been predicted $|T_u| - unk = 0$, then:

$$MAE_u = NaN$$

3.1.11 NDCG

The NDCG abbreviation of Normalized Discounted Cumulative Gain metric is calculated for the **single user** by first computing the DCG score using the following formula:

$$DCG_u(scores_u) = \sum_{r \in scores_u} \frac{f(r)}{log_x(2+i)}$$

Where:

- $scores_u$ are the ground truth scores for predicted items, ordered according to the order of said items in the ranking for the user u.
- f is a gain function *linear or exponential, in particular.*
- x is the base of the logarithm.
- i is the index of the truth score r in the list of scores $scores_u$.

If f is "linear", then the truth score r is returned as is. Otherwise, in the "exponentia" case, the following formula is applied to r:

$$f(r) = 2^r - 1$$

The NDCG for a single user is then calculated using the following formula:

$$NDCG_u(scores_u) = \frac{DCG_u(scores_u)}{IDCG_u(scores_u)}$$

Where:

- $IDCG_u$ is the DCG of the ideal ranking for the truth scores.

So the basic idea is to compare the actual ranking with the ideal one. Finally, the NDCG of the entire system is calculated instead as such:

$$NDCG_{sys} = \frac{\sum_u NDCG_u}{|U|}$$

Where:

- $NDCG_u$ is the NDCG calculated for user u.
- U is the set of all users.

The system average excludes NaN values.

3.1.12 NDCG@k

The NDCG@K abbreviation of Normalized Discounted Cumulative Gain at K metric is calculated for the **single user** by using the [framework implementation of the NDCG][clayrs.evaluation.NDCG] but considering $scores_u$ cut at the first k predictions. The K used for the experiment is .

3.1.13 MRR

The MRR abbreviation of Mean Reciprocal Rank metric is a system-wide metric, so only its result it will be returned and not those of every user. MRR is calculated as such:

$$MRR_{sys} = \frac{1}{|Q|} \cdot \sum_{i=1}^{|Q|} \frac{1}{rank(i)}$$

Where:

- Q is the set of recommendation lists.
- $rank(i)$ is the position of the first relevant item in the i-th recommendation list.

The MRR metric needs to discern relevant items from the not relevant ones. To achieve this, one could pass a custom `relevant_threshold` parameter that will be applied to every user. If the rating of an item is $\geq$ `relevant_threshold`, then it is considered relevant; otherwise, it is not. If no `relevant_threshold` parameter is passed, then for every user, its mean rating score will be used. In this experiment, the relevant threshold used is None.

3.1.14 MRR@K

The MRR@K abbreviation of Mean Reciprocal Rank at K metric is a system-wide metric, so only its result will be returned and not those of every user. MRR@K is calculated as such:

$$MRR@K_{sys} = \frac{1}{|Q|} \cdot \sum_{i=1}^{K} \frac{1}{rank(i)}$$

Where:

- K is the cutoff parameter.
- Q is the set of recommendation lists.
- $rank(i)$ is the position of the first relevant item in the i-th recommendation list.

In this experiment, the relevant threshold used is None.

3.1.15 MAP

The MAP metric abbreviation of Mean average Precision is a ranking metric computed by first calculating the AP abbreviation of Average Precision for each user and then taking its mean. The AP is calculated as such for the **single user**:

$$AP_u = \frac{1}{m_u} \sum_{i=1}^{N_u} P(i) \cdot rel(i)$$

Where:

- m_u is the number of relevant items for the user u.
- N_u is the number of recommended items for the user u.
- $P(i)$ is the precision computed at cutoff i.
- $rel(i)$ is an indicator variable that says whether the i-th item is relevant ($rel(i) = 1$) or not ($rel(i) = 0$).

After computing the AP for each user, we can compute the MAP for the whole system:

$$MAP_{sys} = \frac{1}{|U|} \sum_{u} AP_u$$

This metric will return the AP computed for each user in the dataframe containing users results, and the MAP computed for the whole system in the dataframe containing system results. In this experiment the MAP has

been calculated using a relevant threshold: None.

3.1.16 Correlation

The Correlation metric calculates the correlation between the ranking of a user and its ideal ranking. The current correlation methods implemented are:

- ‘pearson‘
- ‘kendall‘
- ‘spearman‘

Every correlation method is implemented by the pandas library, so refer to its documentation for more information.

The correlation metric is calculated as such for the **single user**:

$$Corr_u = Corr(ranking_u, ideal_ranking_u)$$

Where:

- $ranking_u$ is ranking of the user.
- $ideal_ranking_u$ is the ideal ranking for the user.

The ideal ranking is calculated based on the rating inside the ground truth of the user. The Correlation metric calculated for the entire system is simply the average of every $Corr$:

$$Corr_{sys} = \frac{\sum_u Corr_u}{|U|}$$

Where:

- $Corr_u$ is the correlation of the user u.
- U is the set of all users.

The system average excludes NaN values. It's also possible to specify a cutoff parameter using the `top_n` parameter. If specified, only the first n results of the recommendation list will be used to calculate the correlation. For this experiment, the settings for the correlation metrics are:

- method: pearson.
- top_n: None.

3.1.17 Gini Index

The Gini Index metric measures inequality in recommendation lists. It's a system-wide metric, so only its result it will be returned and not those of every user. The metric is calculated as such:

$$Gini_{sys} = \frac{\sum_i (2i - n - 1)x_i}{n \cdot \sum_i x_i}$$

Where:

- n is the total number of distinct items that are being recommended.
- x_i is the number of times that the item i has been recommended.

A perfectly equal recommender system should recommend every item the same number of times, in which case the Gini index would be equal to 0.

The more the recsys is "unequal", the more the Gini Index is closer to 1. If the `'top_n'` parameter is specified, then the Gini index will measure inequality considering only the first n items of every recommendation list of all users. For this experiment, the `top_n`: None.

3.1.18 Prediction Coverage

The Prediction Coverage metric measures in percentage how many distinct items are being recommended in relation to all available items. It's a system wise metric, so only its result it will be returned and not those of every user. The metric is calculated as such:

$$PredictionCoverage_{sys} = (\frac{|I_p|}{|I|}) \cdot 100$$

Where:

- I is the set of all available items.
- I_p is the set of recommended items.

The I must be specified through the 'catalog' parameter.

3.1.19 Catalog Coverage

The Catalog Coverage metric measures in percentage how many distinct items are being recommended in relation to all available items. It's a system-wide metric, so only its result it will be returned and not those of every user. It differs from the Prediction Coverage since it allows for different parameters to come into play. If no parameter is passed then it's a simple Prediction Coverage. The metric is calculated as such:

$$CatalogCoverage_{sys} = (\frac{|\bigcup_{j=1...N} reclist(u_j)|}{|I|}) \cdot 100$$

Where:

- N is the total number of users.
- $reclist(u_j)$ is the set of items contained in the recommendation list of user j.
- I is the set of all available items.

The I must be specified through the 'catalog' parameter. The recommendation list of every user ($reclist(u_j)$) can be reduced to the first n parameter with the top-n parameter, so that catalog coverage is measured considering only the highest ranked items. With the 'k' parameter one could specify the number of users that will be used to calculate catalog coverage: k users will be randomly sampled and their recommendation lists will be used. The formula above becomes:

$$CatalogCoverage_{\text{sys}} = \left(\frac{|\bigcup_{j=1\ldots k} \text{reclist}(u_j)|}{|I|} \right) \cdot 100$$

Where:

- k is the parameter specified.

Obviously 'k' $<$ N, else simply recommendation lists of all users will be used.

3.1.20 Long Tail Distribution

This metric generates the Long Tail Distribution plot and saves it in the output directory with the file name specified. The plot can be generated both for the truth set or the predictions set based on the on parameter:

- **on = 'truth'**: in this case the long tail distribution is useful to see which are the most popular items the most rated ones.
- **on = 'pred'**: in this case the long tail distribution is useful to see which are the most recommended items.

The plot file will be saved as `out_dir/file_name.format`. Since multiple splits could be evaluated at once, the overwrite parameter comes into play: if set to `False`, files with the same name will be saved as `file_name_(1).format`, `file_name_(2).format`, etc. so that for every split a plot is generated without overwriting any previously generated files.

For this experiment the Long Tail Distribution has been used with the following settings:

- on: truth.
- format: png.
- overwrite: False.

3.1.21 Pop Recs Correlation

This metric generates a plot which has as the X-axis the popularity of each item and as Y-axis the recommendation frequency, so that it can be easily seen the correlation between popular niche items and how many times are being recommended. The popularity of an item is defined as the number of times it is rated in the 'original_ratings' parameter divided by the total number of users in the 'original_ratings'.

- The plot file will be saved as out_dir/file_name.format'

Since multiple splits could be evaluated at once, the overwrite parameter comes into play: if set to `False`, files with the same name will be saved as `'file_name_(1).format'`, `'file_name_(2).format'`, etc., so that for every

split, a plot is generated without overwriting any previously generated files.

There exists cases in which some items are not recommended even once, so in the graph could appear zero recommendations. One could change this behaviour thanks to the 'mode' parameter:

- **mode='both'**: Two graphs will be created, the first one containing eventual zero recommendations, the second one where zero recommendations are excluded. This additional graph will be stored as `out_dir/file_name_no_zeros.format` *the string '_no_zeros' will be added to the file_name chosen automatically.*
- **mode='w_zeros'**: Only a graph containing eventual zero recommendations will be created.
- **mode='no_zeros'**: Only a graph excluding eventual zero recommendations will be created. The graph will be saved as `out_dir/file_name_no_zeros.format` *the string '_no_zeros' will be added to the file_name chosen automatically.*

For this experiment the PopRecsCorrelation has been used with the following settings:

- mode: both.
- format: png.
- overwrite: False.

3.2 Results for sys - fold1

In this section we show the results obtained on the fold: **sys - fold1** , the metrics used during the experiment will be grouped into the following table, that represent the results of the evaluation conducted 2

Metric	Value
CatalogCoverage (PredictionCov)	16.67
F1 - macro	1.0
F1@5 - micro	1.0
Gini	0.0
MAE	1.5
MAP	1.0
MRR	1.0
MRR@5	1.0
MSE	2.25
NDCG	1.0
NDCG@5	1.0
Precision - macro	1.0
Precision@5 - macro	1.0
PredictionCoverage	16.67
R-Precision - macro	1.0
RMSE	1.5
Recall - macro	1.0
Recall@5 - macro	1.0

Table 2: Table of the results

3.3 Results for sys - mean

In this section we show the results obtained on the fold: **sys - mean** , the metrics used during the experiment will be grouped into the following table, that represent the results of the evaluation conducted 3

Metric	Value
CatalogCoverage (PredictionCov)	16.67
F1 - macro	1.0
F1@5 - micro	1.0
Gini	0.0
MAE	1.5
MAP	1.0
MRR	1.0
MRR@5	1.0
MSE	2.25
NDCG	1.0
NDCG@5	1.0
Precision - macro	1.0
Precision@5 - macro	1.0
PredictionCoverage	16.67
R-Precision - macro	1.0
RMSE	1.5
Recall - macro	1.0
Recall@5 - macro	1.0

Table 3: Table of the results

4 Conclusion on the experiment

This part is for conclusion to be sum up as needed.