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:

plot_0 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`

- `NLTK` used as preprocessing technique with following settings:
  - `lang: english`
  - `lemmatization: False`
  - `named_entity_recognition: 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 plot_0 in this experiment.

genres_0 has been represented with the following techniques:

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

- `Spacy` used as preprocessing technique with following settings:
  - `lemmatization: False`
  - `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 genres_0 in this experiment.

genres_1 has been represented with the following techniques:

- `SentenceEmbeddingTechnique` used as content representation technique with following parameters:
  - `embedding_source: Sbert`

- `Spacy` used as preprocessing technique with following settings:
  - `lemmatization: False`
  - `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 genres_1 in this experiment.

No exogenous techniques have been used in this experiment to represent the data used.

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:

Parameter	Value
n_users	943
n_items	1682
total_interactions	100000
min_score	1.0
max_score	5.0
mean_score	3.52986
sparsity	0.93695

Table 1: Table of the Interactions

2.2 Partitioning techinque used

The partitioning used is the Hold-Out Partitioning. This approach splits the dataset in use into a train set and a test set. The training set is what the model is trained on, and the test set is used to see how well the model will perform on new, unseen data.

The train set size of this experiment is the 80.0% of the original dataset, while the test set is the remaining 20.0%.

The data has been shuffled before being split into batches.

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 a graph representation, the algorithm used is `NXPageRank` and the following are its settings:

- `alpha: 0.85`
- `max_iter: 100`
- `nstart: None`
- `personalized: False`
- `tol: 1e-06`
- `weight: weight`

The type of graph used for the algorithm is a NXFullGraph , the mode used is rank and the number of recommendations given is None. 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 $\geq$ **3.52986**.
- 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 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: 2**
- **relevant threshold:None**
- **sys average: macro**

3.1.3 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: 1**

- **β: 1**
- **relevant threshold: None**
- **sys average: macro**

3.1.4 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.

3.1.5 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.

3.2 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**
F1@1 - macro	0.04221
MRR	0.79602
NDCG	0.93529
Precision - macro	0.55697
Precision@2 - macro	0.6527

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
F1@1 - macro	0.04221
MRR	0.79602
NDCG	0.93529
Precision - macro	0.55697
Precision@2 - macro	0.6527

Table 3: Table of the results

4 Conclusion on the experiment

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