Report ClayRS

Semantic Web Access and Personalization Research Group

December 17, 2023

This LaTeX document was generated from yaml files for the purpose of replicability of experiments done with ClayRS, it contains information about the dataset, preprocessing methods, analysis algorithms and the results of the experimental evaluation.

1 ClayRS configurations of the experiment

1.1 Dataset

In this experiment, the **MovielensDataset** was used.

The statistics of the dataset used are reported in the following table **??**:

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

The embendding techniques used during the processing of the document are the following:

- Gensim glove-twitter-25 has been used as word embedding.

- Sbert has been used as sentence embedding.

1.2 Preprocessing

The preprocessing used is NLTK, a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

In this experiment, those operations of NLTK were used:

- **Strip multiple whitespace**, an operation that removes multiple whitspaces between words.
- **Stopwords removal**, used to remove the stopwords occurring in the text.

spaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python. spaCy is designed specifically for production use and helps you build applications that process and understand large volumes of text. It can be used to build information extraction or natural language understanding systems, or to pre-process text for deep learning. In this experiment, those spaCy operations have been used:

In this experiment, those spaCy operations were used:

- **Strip multiple whitespace**, an operation that removes multiple whitspaces between words.
- **Stopwords removal**, used to remove the stopwords occurring in the text.

1.3 Partitioning

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 splitted into batches.

1.4 Algorithm Used

In this experiment a Graph Based Recommender Algorithm has been used. Specifically the NXPageRank algorithm, a Page Rank algorithm based on the networkx implementation The PageRank can be 'personalized', in this case the PageRank will be calculated with Priors. The alpha value used was: 0.85

1.5 Metrics

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 ratingasdjiasd $\geq$ **3.52986**
- fp_u is the number of items which are in the recommendation list of the user and have a rating $<$

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**.

sys_average, a parameter that specifies how the system average must be computed the default value is 'macro', in this experiment the value of the sys_average is **macro**.

Precision at k is the proportion of recommended items in the top-k set that are relevant. 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 $\geq$ **3.52986**

- $tp@K_u$ is the number of items which are in the recommendation list of the user **cutoff to the first K items** and have a rating $<$ **3.52986**

In this experiment the value **k is 2**, the sys_average is **macro**

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

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

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:

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

In this experiment $\mathbf{k = 1}$, $\boldsymbol{\beta = 1}$ and **sys_average** is **macro**.

1.6 Results

In the following table, we present the results of the evaluation **??**

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

Table 2: Table of the results