myICF User Guide

Version 0.0.1

Klismam Pereira

September 7, 2021

Contents

List of Code Listings

1 Introduction

This document explains how to use the code implemented for Incremetal Collaborative Filtering (ICF) recommender system, based on the algorithm presented in the paper by Papagelis et al. (2005), "Incremental Collaborative Filtering for High-Scalable Recommendation Algorithm" [1]. This is an incremental, neighborhood-based (user-based) collaborative filtering recommendation system algorithm. The code was implemented in *Python*.

2 Setup and imports

This implementation only uses the package *csv* that is already included in Python distributions. The package *matplotlib* was also used for the plotting results, but it is not directly related to the algorithm implementation.

While using the annexed example notebook, entitled *myICF Example of usage.ipynb*, only the algorithm utilities and matplotlib must be imported:

```python
from myICF.utils import stream, myICF_helper
import matplotlib.pyplot as plt
```

Listing 1: External packages import

Please, do refer to the annexed example usage notebook and html files in case of doubts.

3 Entities definition

The code is composed by two classes: *myICF* and *myICF_helper*. The algorithm is implemented in the methods of *myICF*, while *myICF_helper* is used as an utility extension that allows the user to obtain item descriptions from the datasets (refer to the example notebook/html files). An utility generator function *stream* was also created to simulate a stream of ratings, from a preexisting dataset, to feed the recommender system.

3.1 Function *stream(filepath, delimiter, max_cases=*500*)*

Stream function generates a stream of users/items interactions in the form of ratings.

3.1.1 Parameters

- **filepath: *string*, *default=None***
  Path of the file containing the *.csv* file, corresponding to the dataset from which the ratings stream is to be generated. Columns named 'user', 'item', and 'rating' are expected.

- **delimiter: *string*, *default=None***
  Character that separates the dataset columns in the *.csv* file.

- **max_cases: *integer*, *default=*500**
  Maximum number of cases yielded by the function.

3.1.2 Returns

- **user, item, rating: *integer*, *integer*, *float***
  Returns a stream of values from the original dataset formed by the user ID, item ID and corresponding interaction rating.

3.2 Class *myICF(corr_threshold=*0.65*, high_rating=*4*)*

This is the main class of the recommender system. It has several methods that initiate data structure for new users (dictionaries), update old ratings or add new ratings, calculate increments and updated factors of the auxiliar variables (needed to have an incremental neighborhood-based recommender system) and recommend a list of 'n' items to user based on similarity to other users.

When a new object from *myICF* class is created, three new dictionaries are created: one to store all the ratings, another to store some cache information about the ratings provided by each user (number of ratings and average) and a third one to store the remaining auxiliar information needed for each pair of users in order to enable the incrementality of the algorithm.

3.2.1 Parameters

- **corr_threshold: *float*, *default=*0.65**
  Minimum Pearson correlation value between users considered to perform a recommendation.

- **high_rating: *integer*, *default=*4**
  Minimum rating value of the items to include in a recommendation within the considered users.

3.2.2 Attributes

- **user_ratings: *dictionary***
  Dictionary to store ratings, grouped by each user.
- **user_meta: *dictionary***
  Dictionary to cache information about the ratings provided by each user (number of ratings and average rating).
- **user_pair_meta: *dictionary***
  Dictionary to cache auxiliar information for each pair of users.

3.2.3 Methods

- **Method *_new_user(self, user, item, rating)***
  - **Parameters:**
    - **self: *object***
      Actual *myICF* object.
    - **user: *string, default=None***
      User ID.
    - **item: *string, default=None***
      Item ID.
    - **rating: *float, default=None***
      Rating assigned by *user* to *item*.
  - **Returns:**
    - **self**
      Modified *myICF* object with updated *user_ratings*, *user_meta* and *user_pair_meta* dictionaries, now including the new user.
- **Method *_new_rating(self, user, item, rating)***
  - **Parameters:**
    - **self: *object***
      Actual *myICF* object.
    - **user: *string, default=None***
      User ID.
    - **item: *string, default=None***
      Item ID.
    - **rating: *float, default=None***
      Rating assigned by *user* to *item*.

- **Returns:**
    * **self**
      Modified *myICF* object with updated *user_ratings*, *user_meta* and *user_pair_meta* dictionaries, now including the new rating.

- **Method** ***_update_rating(self, user, item, rating)***
  - **Parameters:**
    * **self:** ***object***
      Actual *myICF* object.
    * **user:** ***string, default=None***
      User ID.
    * **item:** ***string, default=None***
      Item ID.
    * **rating:** ***float, default=None***
      Rating assigned by *user* to *item*.
  - **Returns:**
    * **self**
      Modified *myICF* object with updated *user_ratings*, *user_meta* and *user_pair_meta* dictionaries, now including the updated rating.

- **Method** ***_get_coratings(self, userA, userB)***
  - **Parameters:**
    * **self:** ***object***
      Actual *myICF* object.
    * **userA:** ***string, default=None***
      User ID.
    * **userB:** ***string, default=None***
      User ID.
  - **Returns:**
    * **A_sum_coratings, B_sum_coratings, n_coratings, key:** ***integer*****,** ***integer*****,** ***integer*****,** ***tuple of two strings***
      Returns the sum of the ratings from the items corated between users A and B, for user A and for user B, the number of items corated and the key from the corresponding pair of users for *user_pair_meta* dictionary.

- **Method** ***_update_get_coratings(self, userA, userB, item, rating, new_rating=*****True*****)***

- **Parameters:**
  - **self:** ***object***
    Actual *myICF* object.
  - **userA:** ***string, default=None***
    User ID.
  - **userB:** ***string, default=None***
    User ID.
  - **item:** ***string, default=None***
    Item ID.
  - **rating:** ***float, default=None***
    Rating assigned by *userA* to *item*.
  - **new_rating:** ***boolean, default=*True**
    True if the rating assigned by *userA* to *item* is new.
- **Returns:**
  - **A_sum_coratings, B_sum_coratings, n_coratings, key:** ***integer*, *integer*, *integer*, *tuple of two strings***
    Updates and returns the updated sum of the ratings from the items corated between users A and B, for user A and for user B, the number of items corated and the key from the corresponding pair of users for *user_pair_meta* dictionary.

- **Method** ***run(self, user, item, rating)***
  - **Parameters:**
    - **self:** ***object***
      Actual *myICF* object.
    - **user:** ***string, default=None***
      User ID.
    - **item:** ***string, default=None***
      Item ID.
    - **rating:** ***float, default=None***
      Rating assigned by *user* to *item*.
  - **Returns:**
    - ***Nothing***
      Calls *_new_user*, *_new_rating* or *_update_rating* methods depending on whether the user and/or item is/are new or not.

- **Method** ***recommend(self, user, n_recs=*10*)***
  - **Parameters:**

    * **self:** ***object***
      Actual *myICF* object.
    * **user:** ***string, default=None***
      User ID.
    * **n_recs:** ***integer, default=10***
      Number of items to be recommended to the *user*.
  - **Returns:**
    * **recommended_items:** ***list of integers***
      Returns a list of *n_recs* recommended items IDs.

3.3 Class *myICF_helper(filepath, delimiter='\t', description_column='title', corr_threshold=0.65, high_rating=4)*

This is a helper class that was implemented to improve the interpretability of the recommendations made by the recommender system, labelling the items' IDs with the corresponding descriptions. This class inherits all methods from *myICF*. Given that movielens 100k is the dataset used in this project, this class has utility methods to recover the title of movies, in order to present the recommendations as movie titles instead of movie IDs. If another dataset other than movielens is used, the parameter 'description_column' must be modified to accept a proper item description column instead of the 'title' column.

3.3.1 Parameters

- **filepath:** ***string*, *default=None***
  Path of the file containing the *.csv* file where the information relating items' IDs to the corresponding descriptions is stored.
- **delimiter:** ***string*, *default=None***
  Character that separates the dataset columns in the *.csv* file.
- **description_column:** ***string*, *default=*'title'**
  Column in the tabular data that contains descriptive information about items. i.e. for movielens, this column is represented by 'title', which contains the title of movies.
- **corr_threshold:** ***float*, *default=*0.65**
  Minimum Pearson correlation value between users considered to perform a recommendation.

- **high_rating:** ***integer*, *default*=4**
  Minimum rating value of the items to include in a recommendation within the considered users.

3.3.2 Attributes

- **labels_dict:** ***dictionary***
  Dictionary with items' IDs descriptions.

3.3.3 Methods

- **Method *get_titles(self)***
  - **Parameters:**
    * **self:** ***object***
      Actual *myICF* object.
  - **Returns:**
    * **labels_dict:** ***dictionary***
      Returns the dictionary with items' IDs descriptions.
- **Method *user_favorites(self, user, n_items=10)***
  - **Parameters:**
    * **self:** ***object***
      Actual *myICF* object.
    * **user:** ***string, default=None***
      User ID.
    * **n_items:** ***string, default=None***
      Number of *user*'s favorite items to be presented.
  - **Returns:**
    * ***list of strings***
      Returns a list of *n_items user*'s favorite items descriptions.
- **Method *show_recommended_items(self, user, n_items=*10*)***
  - **Parameters:**
    * **self:** ***object***
      Actual *myICF* object.
    * **user:** ***string, default=None***
      User ID.

    * **n_items: *string, default=None***
      Number of *user*'s recommended items to be presented.
  - **Returns:**
    * ***list of strings***
      Returns a list of *n_items user*'s recommended items descriptions.

4 Recommender system object creation

Below, an example is shown about how to create the recommender system object, to feed it with users/items interations (in the form of ratings) and to obtain recommendations.

You can also use the annexed Jupyter Notebook entitled 'myICF Example of usage' to perform these operations yourself.

To create the recommender system object with interpretability capabilities, we just have to import the recommender algorithm utilities and instantiate a *myICF_helper* object.

```python
from myICF.utils import stream, myICF_helper

icf = myICF_helper()
```

Listing 2: *myICF_helper* object creation

To add or update a user/item interaction within the recommender system, we just have to call the *run* method.

```python
for user, item, rating in stream(filepath, delimiter, max_cases):
    icf.run(user, item, rating)
```

Listing 3: *myICF_helper run* method

To show the items with highest ratings from an user, we just have to call the *user_favorites* method.

```python
icf.show_recommended_titles(user, n_items=15)
```

Listing 4: *myICF_helper user_favorites* method

To show the items recommended for an user, we just have to call the *show_recommended_titles* method.

```
icf.user_favorites(user, n_items=15)
```

Listing 5: *myICF_helper show_recommended_titles* method

5 References

[1] Papagelis M., Rousidis I., Plexousakis D., Theoharopoulos E. (2005) Incremental Collaborative Filtering for Highly-Scalable Recommendation Algorithms. In: Hacid MS., Murray N.V., Raś Z.W., Tsumoto S. (eds) Foundations of Intelligent Systems. ISMIS 2005. Lecture Notes in Computer Science, vol 3488. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11425274_57