Recommendation system for a music social network

Exam Specifications
Data Analysis and Classification 2019/2020
Lecturers: M. Papandrea, S. Giordano, L. Luceri

You are requested to build a system which is able to **recommend music to users** belonging to a music social network.

The Dataset

The dataset you are receiving contains social networking, tagging, and music artist listening information from an online music system. For each user in the dataset it contains a list of their top most listened artists including the number of times those artists were played. It also includes user applied tags to artists. The dataset has also information about the social network of people in the music network.

The dataset you are receiving is composed by the following files.

artists.csv

Columns: <*id, name*>
This file contains information about music artists in the dataset, which have been listened and tagged by the users. Each row in the file represents an artist.

- Column *id* contains the artist's id.
- Column *name* contains the name of the artist.

tags.csv

Columns: <*tagID, tagValue*>
This file contains the set of tags available in the dataset, which are then used by users for tagging the music type of the artists on the platform. Each row in the file represents a tag:

- Column *tagID* contains the numerical ID of the tag.
- Column *tagValue* contains textual value of the tag (es. metal, classic, etc).

user_artists.csv

Columns: <*userID, artistID, weight*>
This file contains the artists listened by each user. Each row in the file represents a tuple user-artist:

- Column *userID* contains the numerical ID of a user.
- Column *artistID* contains numerical ID of an artist.

- Column *weight* contains a listening count for the related [user, artist] pair.

<u>user_taggedartists.csv</u>
Columns: <*userID, artistID, tagID, day, month, year*>
<u>user_taggedartists_timestamps.csv</u>
Columns: <*userID, artistID, tagID, timestamp*>

These files contain the tag assignments of artists provided by each particular user. It also contains the date (or respectively the timestamp) information when the tag assignment has been done.

- Column *userID* contains the numerical ID of a user.
- Column *artistID* contains numerical ID of an artist.
- Column *tagID* contains the tag assigned from the user to the artist.
- Columns <*day, month, year*> contain the date of the tag assignment.
- Column *timestamp* contains the timestamp of the tag assignment

<u>user_friends.dat</u>
Columns: <*userID, friendID*>
This file contains the friend relations between users in the database.

- Column *userID* contains the numerical ID of a user.
- Column *friendID* contains the numerical ID of a user's friend (another user on the *music social* network).

The dataset contains also a *<u>testing</u>* part which is composed by a single file. This file will be considered for generating the results file, to be submitted as result of the developed recommendation algorithms (see tasks 4 and 6).

<u>test_user_artists.csv</u>
Columns: <*userID, artistID*>
Each row in the file represents a couple user-artist:

- Column *userID* contains the numerical ID of a user.
- Column *artistID* contains numerical ID of an artist.

Tasks

Given the dataset, you are requested to work on the tasks described below.

Task1: EDA

Explore the dataset (shape of each file, data types, values distribution, ...). Below some suggestion about an *initial* explorative analysis.

1. Calculate the number of users and artists present in the network.
2. Calculate the number of bi-directional user-friend relations pairs.
3. Calculate the average friend relations per user.
4. Calculate the number of user-listened artists relations (couples user-artist).
5. Plot the distribution of the number of artists listened by each user, and calculate the average and mode of the distribution.
6. Plot the distribution of the number of tag assignments per each artist, and calculate the average and mode of the distribution.
7. Plot the distribution of the number of tag assignments per each user, and calculate the average and mode of the distribution.
8. Plot the distribution of the number of distinct tag assignments per each artist, and calculate the average and mode of the distribution.

Task2: Clustering tags

Clustering music tags according to the artists association (content vectors).

1. Cluster tags according to the association to artists. Justify your choices and comments on the results. Consider at least two different clustering algorithms based on similarity between tags, and compare the clustering.
2. Associate to each cluster a meta-tag, and consider using it as tag for the music social network.
3. Group music artists based on the meta-tag and analyse the distribution of the number of artists with respect to the related meta-tags.

Task3: Social network analysis

1. Compute the network properties: degree distribution, density, diameter.
2. Compute the centrality measures, select the one that you consider more representative and justify your choice.
3. Extract:
   - The communities based on social network
   - The communities based on similarity (e.g., preferred artist(s), music type)

4. Compute the global clustering coefficient of:

    - the global network
    - the communities
    - (OPTIONAL) compare them and discuss the result
5. Compute the local clustering coefficients, and extract the average, within:
    - the global network
    - the communities
    - (OPTIONAL) compare them and discuss the result
6. Compute the similarity between users according to:
    - Their common friends
    - The artist(s) listened
    - Music type
    - (OPTIONAL) The artist(s) preferred by their friends
    - (OPTIONAL) The type of music listened by their friends

Task4: Binary classifier for inferring users's artists interest.

Build a binary classifier which is able to infer, given a tuple <user-artist>, if the user would be interested in the artist or not. In order to build the classifier consider a set of features (at least 10 features are expected) which characterize a tuple user-artist. Some examples of features are listed below, given a tuple of user U and artist A.

- Feature 1: popularity of artist A.
- Feature 2: popularity of the N clusters C(i), which artist A belongs to.
- Feature 3: number of artists listened to by user U, belonging to the same clusters C(i) of artist A.
- Feature 4: number of friends of user U, which listened to artist A.
- Feature 5: boolean representing if artist A similarity to the user's most listened artists is lower than a certain distance threshold *d*.
- Feature 6: the communities computed in Task 3.

Task5: (OPTIONAL) Scalable classifier

Similarly to Task 4, build a classifier which is able to infer, given a tuple <*user-artist*>, if the user would be interested in the artist or not, but also consider that new users can enter the network. In this sense, the proposed solution should be able to minimize the training phase, that is: it should be avoided to have to restart the training phase from scratch.
In order to build such a dynamic and scalable classifier consider only part of the original network (e.g ¾ of the original network) and then simulate to add the remaining users.
Show the performances of the proposed classifier and compare it with the performances of the one produced in Task4 in the case of ⅚ of the original network in the beginning and the remaining ⅙ entering the network later.

Task6: Inferring users interest in artist.

Build a system which is able to recommend the weight of a couple <*user-artist*>, which is the number of times a user is going to listen to this artist. The recommendation should be categorical, and the possible values are the following.

- HIGH: weight greater than 1000 (high interest in the artist).
- MEDIUM: weight between 1 and 1000 (medium interest in the artist).
- LOW: weight equal to zero (no interest in the artist).

In order to build the classification model, consider a set of features (at least 10 features are expected) which characterize a tuple user-artist. Some examples of features are listed in task 4.

Submission Details

The testing dataset contains a file *test_user_artists.csv*. This file contains a list of tuples <*user **U** - artist **A***>.

(Task4) Per each tuple <U,A> predict **1** (True) if you predict the user would be interested in listening to artist A, predict **0** (False) if you predict the user would not be interested in listening to artist A.

(Task6) Per each tuple <U,A> predict **HIGH (2)** if you predict the user would be highly interested in listening to artist A, predict **MEDIUM (1)** if you predict the user would have medium interested in listening to artist A, predict **LOW (0)** if you predict the user would have no interested in listening to artist A.

This results have to be submitted in a file formatted as follows (an example is reported below).

results.csv

Columns: <*userID, artistID, isInterested, weight*>

- Columns <*userID,artistID*> correspond to the same columns in the *test_user_artists.csv* file, and contains all the columns of the test file (same order of appearance).
- Column *isInterested* is a boolean column, contains 0 in correspondence of rows whose prediction is negative, and 1 in correspondence of rows whose prediction is positive. This column is related to Task 4.
- Column *weight* is a numerical column, contains 0 in correspondence of rows predicted with LOW interest, contains 1 in correspondence of MEDIUM interest and contains 2 in correspondence of HIGH interest. This column is related to Task 4.

```
userIDartistID    isInterestedweight


2      52     0      0
2      53     1      2
230    53     0      1
510    120    1      1
714    199    0      0
```

<u>Submission Dates</u>

You are requested to submit, by **<u>February 2nd</u>**:

1. The *result.csv* file described above;
2. All the source code you wrote to generate the *result.csv* file;
3. A text document which describes your algorithm, your reasonings and choices, and your work per each task presented above. A 10-pages-maximum summary of this document can be carried with you at the written exam of January 27th, as a printed copy.

<u>Evaluation</u>

The Evaluation of the classification results will be provided in terms of *accuracy*, *precision*, and *recall*, calculated over the the GT associated to the test file.

You can submit the result file before the final submission of the complete Project at any time, and multiple times (maximum 5 times per student) before the final submission, in order to get the updated evaluation of your results.

For getting the classification results send an email:

- to michela.papandrea@supsi.ch
- object: **DaCla Final Project, attempt <N> <Surname>**
  (es. *DaCla Final Project, attempt 1 Papandrea*)
- Attachment: **result.csv**
  the formatting of the file is specified above