Group Project Description

Introduction to Computer Programming 2021/2022

Group Delegate ID: 264851

1 Dataset Description

For this group project, you will be working on a subset of the Internet Movie DataBase (IMDB) dataset. The dataset contains some data about movies, their cast/crew and their ratings.

For the sake of data handling, the dataset is split into three different `.pkl` files. Such files are so-called *pickle* files, a Python serialisation technique for storing data on your disk[1].

Each file contains a different data structure, hereinafter referred to:

- `title_basics`, stored in the file `title_basics.pkl`, that contains some basic information about movies;
- `title_ratings`, stored in the file `title_ratings.pkl`, that contains the IMDB user ratings;
- `title_principals`, stored in the file `title_principals.pkl`, that contains some information about the movies main cast and crew.

1.1 Description of `title_basics`

`title_basics.pkl` contains a list-of-lists. Each entry of the list corresponds to a specific movie and each movie is characterized by its own information, also stored in a list. A snapshot of this list-of-lists follows:

```
[['tt0000001', 'short', 'Carmencita', '1', 'Documentary,Short', '1894'],
...,
['tt0000004', 'short', 'Un bon bock', '12', 'Animation,Short', '1892'],
...]
```

From the above snippet, we can recognize the six features that characterize each movie:

1. an alphanumeric unique identifier[2] of the movie;

[1] If you are interested, you can find more details about *pickle* files at `https://docs.python.org/3/library/pickle.html`

[2] Since identifiers are unique, there are no different movies sharing the same identifier.

2. the type/format of the movie. For simplicity, this element can assume only two possible values: `'short'` (i.e., short movie) and `'movie'` (i.e., full-length movie);

3. the full title of the movie;

4. the duration, in minutes;

5. the genre(s) of the movie: if a movie spans multiple genres, they are separated by commas;

6. the release year of the movie.

Note: Numerical features (notably, release year and duration) are given as strings.

Recall: To access items in a list-of-lists you have to use a double index. For example `L[i][j]` takes the j^{th} item in the i^{th} list of the list-of-lists `L`. It is the same as doing:

```
sublist = L[i]
my_item = sublist[j]
```

A working example follows. Let

```
L = [['a', 'c', 'z'], [109, 110, 87]]
```

be a list-of-lists. To get the value `'a'`, which is the 0^{th} item[3] of the 0^{th} list, you have to do `L[0][0]`. Similarly, if you want to get the value `87`, which is the 2^{nd} item of the 1^{st} list, you have to do `L[1][2]`.

1.2 Description of `title_ratings`

`title_ratings.pkl` contains a dictionary, where:

- the key is the alphanumeric identifier of the movie;
- the value is a list with two elements:
  1. the first item (a float) is the average rating of the movie;
  2. the second item (an integer) is the total number of votes for that movie.

A snapshot of this dictionary follows:

```
{'tt0000001': [5.7, 1817],
 'tt0000002': [6.0, 233],
 'tt0000003': [6.5, 1574],
 'tt0000004': [6.1, 151],
 'tt0000005': [6.2, 2395],
 'tt0000006': [5.2, 157],
```

[3] Recall that, in Python, indices start at 0.

```
'tt0000007': [5.4, 747],
'tt0000008': [5.5, 1974],
'tt0000009': [5.9, 190],
...}
```

Recall: To access items in such dictionary-of-lists you have to use a double index as well. The first index must dictate the dictionary value that you want to access, whereas the second index must dictate the position within the list. For example, given the above dictionary snapshot, to access the value 233 you have to do `d['tt0000002'][1]`. It is the same as doing:

```
my_list = d['tt0000002']      # now my_list contains [6.0, 233]
my_value = my_list[1]         # now my_value contains 233
```

1.3 Description of title_principals

Finally, `title_principals.pkl` contains another dictionary, where:

- the key is the alphanumeric identifier of the movie;
- the value is another dictionary.

The dictionary that populates the value of the 'main dictionary' has two keys:

1. `'cast'`, which contains a list of the main actors and actresses of the movie;
2. `'director'`, which contains a list with the director(s) of the movie.

A snapshot of these nested dictionaries follows:

```
{'tt0000001': {'director': ['William K.L. Dickson'], 'cast': []},
 'tt0000002': {'director': ['Émile Reynaud'], 'cast': []},
 'tt0000003': {'director': ['Émile Reynaud'], 'cast': []},
 'tt0000004': {'director': ['Émile Reynaud'], 'cast': []},
 'tt0000005': {'cast': ['Charles Kayser', 'John Ott'],
 'director': ['William K.L. Dickson']},
 'tt0000006': {'director': ['William K.L. Dickson'], 'cast': []},
 'tt0000007': {'cast': ['James J. Corbett', 'Peter Courtney'],
 'director': ['William K.L. Dickson', 'William Heise']},
 'tt0000008': {'cast': ['Fred Ott'], 'director': ['William K.L. Dickson']},
 ...}
```

Note: Those lists can as well be empty.

1.4 Dataset Loading

In order to load these files, you have to do the following steps:

```
import pickle      # this will load the pickle module
title_basics = pickle.load(open('/path/to/title_basics.pkl','rb'))
```

```
title_ratings = pickle.load(open('/path/to/title_ratings.pkl','rb'))
title_principals = pickle.load(open('/path/to/title_principals.pkl','rb'))
```

where `title_basics`, `title_ratings` and `title_principals` are the name of the variables that will contain the three data structures detailed in Sections 1.1, 1.2 and 1.3, respectively. Of course, you may change their names, if you prefer.

Inside the `open()` function, the `'rb'` flag must not be changed[4], whereas `'/path/to/some_file.pkl'` is the path on your disk pointing to `'some_file.pkl'` and must be changed accordingly.

2 Task 1: Data Cleaning

Real-world data are (most of the times) noisy. This holds also for the IMDB dataset. Luckily, IMDB provides a flag, namely `'\N'`, that identifies whether a particular field is missing or null for that title/name.

Your first task is to cleanup (a little bit) this dataset, by:

- removing all movies having the infamous `'\N'` flag in any of its fields
- converting all numerical attributes in numerical data types.

Hint: After cleaning the data, it is recommended to save the cleaned version of the dataset so you can resume your work without having to clean it again every time. To do so, you can use the following lines of code:

```
import pickle      # this will load the pickle module, if not already loaded
pickle.dump(X, open('/path/to/filename.pkl','wb'))
```

where `X` is the variable that you want to save and `'/path/to/filename.pkl'` is the path on disk (filename included) of the *pickle* file that will be created (change them accordingly). The `'wb'` flag must not be changed[5]. Trivially, do not overwrite the original *pickle* files, so you can always roll back to the 'original' version of the dataset. You can resume your work by loading the cleaned dataset thanks to the very same instructions from Section 1.4.

3 Task 2: Query #1

Write a Python script that calculates the movie with the highest number of personnel (both actors/actresses and directors) amongst all movies with a runtime of at least 1 hour.

The script must generate a list that contains:

1. the title of the movie;

[4] It's an abbreviation for *read binary*. It basically tells `pickle.load()` to read a Python object serialised in binary format.

[5] It's an abbreviation for *write binary*. It basically tells `pickle.dump()` to serialise the Python object in binary format.

2. its duration;

3. the number of actors/actresses;

4. the number of directors.

The Python script has to start with the `import` of the cleaned dataset *pickle* files and end with the `dump` of the above-defined list.

Note: If multiple movies satisfy the duration constraint and have the same number of personnel, just return one of them. It is up to you to choose which one to return, there are no additional constraints.

4 Task 3: Query #2

Write a Python script that considers the movies released in the following four year ranges:

1. 1900 and before;

2. from 1901 to 1950;

3. from 1951 to 2000;

4. 2001 and later.

For each time range, the script must calculate the most popular genre in terms of number of movies with that genre.

The script must generate a *pickle* file with a list of the form:

`[g1, n1, g2, n2, g3, n3, g4, n4]`

where:

- `g1`, `g2`, `g3`, `g4` are the resulting most popular genres in the four time ranges, respectively;
- `n1`, `n2`, `n3`, `n4` are the corresponding number of movies.

The Python script has to start with the `import` of the cleaned dataset *pickle* files and end with the `dump` of the above-defined list.

Note: If a movie has multiple genres, that movie counts for all genres separately.

5 Final Remarks

The deadline for submitting the project is 2 days before the exam date.

Submissions must include:

1. three Python source files: `cleaning.py`, `query1.py` and `query2.py` containing the source code for the three tasks in Sections 2, 3 and 4, respectively;
2. two *pickle* files: `query1.pkl` and `query2.pkl` containing the output of the two tasks in Sections 3 and 4, respectively.

The submission must be done by sending the above material to

`amartino@luiss.it`

The subject of the e-mail must read as

`[Intro to CP 2021/2022] Group <ID> Project Submission`

where `<ID>` must be replaced with the student ID of the Group Delegate.

If you are stuck and you feel difficulties in going on with your project, please send me an e-mail.