Assignment 7: YouTube Trending Videos

YouTube is a video-sharing platform founded in 2005 by Chad Hurley, Steve Chen, and Jawed Karim. One year later it was bought by Google. Millions of users access YouTube's content every single day, and from day to day we see how certain videos become viral. In this assignment we will briefly analyze a dataset containing some of the most trending videos from 2017 and 2018 in the United States of America. We will start working also with the `statistics` module, which can be used to compute **central tendency** measures (describing typical values) and **variability** measures (describing how far data fall from the center). Some of the central tendency measures we will use are **median** and **mean**; and the central tendency measure corresponds to the **standard deviation**. You don't need to be an expert on these topics to solve the tasks. They are designed in such a way that you only need your programming skills to provide a solution. However, we encourage you to start understanding these terms.

- About central tendency measures
- About variability measures

Introduction

In this assignment you are requested to complete the program YouTube Trending Videos.

To start with the assignment, go to the file `__main__.py` located in the path `youtube/youtube/__main__.py`. Then, fill in the information related to the author name, ID, and Date:

```
* Author: ...          (e.g. Ludwig von Beethoven)
* TU/e ID number: ... (e.g. '1234567')
* Date: ...           (e.g. '2020-08-31')
```

Afterwards, you can focus on the development of the project. To complete the program, you will need to complete six tasks. Each task is related to the definition of a set of classes and functions, which are declared within the template project with the **marker line**:

```
# TODO: [Label] Description
```

Notice that one task might involve the definition of more than one class or function. If so, the description of the task will make it explicit.

Grading

This assignment consists of 6 tasks. Each task accounts for a different percentage of the grade. The correct development of each class and function within the project is automatically checked by a set of unit tests. The assignment will eventually be graded by a tutor. Make sure all functions have type hints and a docstring that describes their purpose. You must follow all coding conventions defined on the Python Coding Standard document available via Canvas, otherwise points will be deducted from your grade

Note that all passing unit tests do not mean that your assignment is perfect. However, failing tests mean that your assignment contains errors, such as incomplete tasks or bad code, which will certainly lead to a lower grade.

Dataset

This project relies on the `youtube-trends-us.csv` dataset. This dataset was obtained from kaggle, and it was created by Mitchell Jolly in 2017 and last updated in 2019 under the CC0 1.0 license. The last access of the file was done on the 5th of October of 2020. The dataset and its related code can be found here, however, do not download this dataset and only use the one that is provided with the project template.

The dataset is stored as a CSV file, whose values are separated by commas. The CSV file contains the following 13 headers:

Column	Description
video_id	Id of the YouTube video
trending_date	Date in which the video got viral
title	Title of the video
channel_title	Name of the channel that published the video
category_id	Id pointing to the category of the video
publish_time	Time and date when the video was published
tags	List of tags associated to the video
views	Number of views of the video
likes	Number of likes of the video
dislikes	Number of dislikes of the video
comment_count	Number of comments of the video
thumbnail_link	Link to the thumbnail image of the video
description	Description of the video

Project Content

The Netflix project is organized as follows.

`youtube`
> `data`
>> `youtube-trends-us.csv`
> `docs`
>> `assignment-week7.pdf`
> `tests`
>> `test_youtube_core.py` >> `test_youtube_types.py` `tests`
> `youtube`
>> `__init__.py`
>> `__main__.py`
>> `youtube_core.py`

>> `youtube_types.py`
>> `youtube_utils.py`

- **data:** The `data` folder contains the Netflix dataset (i.e. `youtube-trends-us.csv`) used along the assignment of week 7.
- **docs:** The `docs` folder contains a PDF version of this documentation (i.e. `assignment-week7.pdf`).
- **tests:** The folder `tests` contains two test files (i.e. `test_youtube_core.py` and `test_youtube_types.py`) to validate the code written in the `youtube_core.py` and `youtube_types.py` files.
- **youtube:** The `youtube` package is the main package of the project. It contains:
    - `__init__.py`: defines `youtube` as a module.
    - `__main__.py`: it is the main file to run the project and it also contains the author information. It is also the point of connection with the command-line.
    - `youtube_core.py`: this module contains the main functionality of the project.
    - `youtube_types.py`: this module contains the `Video` type and some helper functions.
- `netflix_utils.py`: this module contains utility functions used along the project.

Tasks

General Remarks

- For this assignment, we will work with a command-line program which interacts with the user through the command line. A GUI is not used in this case.
- All your code should be written on the `youtube_core.py` or `youtube_types.py` files.
- The header of the function may already be given without type hints. It is expected that these type hints are inserted.
- We suggest you to follow the reStructuredText (reST) style when creating your docstrings.
- If you want to add additional functions or methods that can help you solve each task, you are free to do it.

Task 1: Create the Video Class (15%)

- Create the `Video` class in the `youtube_types.py` module. The class has the following attributes and they should be passed to the constructor in the given order: `title` (string), `channel` (string), `publish_date` (datetime.date), `trending_date` (datetime.date), `tags` (list of strings), `views` (integer), `likes` (integer), `dislikes` (integer), `comments` (integer), and `links` (list of strings).

- Override the `__str__` method so it returns a string in the following form: '`title` published by `channel`'.
- Override the `__eq__`, `__ge__`, `__gt__`, `__le__`, `__lt__`, and `__ne__` methods, so they perform a normal string comparison between the titles of a video.
- Define the `__str__()` method in the `Video` class that returns a string such as: '`title` published by `channel`'.

Task 2: Extract Dates (15%)

- Define the `extract_publish_date` function in the `youtube_types.py` module. The function gets the publish time of a video as a string. This string is of the form `YYYY-MM-DDThh:mm:ss.fffZ` (e.g. `2017-11-13T07:30:00.000Z`). Then, use regular expressions to extract the year, month, and day out from the given string. Return a `datetime.date` consisting of the extracted year, month, and day in the given order.
- Define the `extract_trending_date` function in the `youtube_types.py` module. The function gets the trending date of a video as a string. This string is of the form `YY.DD.MM` (e.g. `17.31.01`). Then, use regular expressions to extract the year, month, and day out from the given string. Notice that the year only has the last two integers of the number. You must add this value to `2000`. Return a `datetime.date` consisting of the extracted year, month, and day in the given order.

Task 3: Extract Tags and Links (20%)

- Define the `extract_tags` function in the `youtube_types.py` module. The function gets all tags associated to a video as a string. This string is a list of tags separated by `|`. Some of these tags are enclosed with double quotes. You must return a list of strings where each element represents a tag. Make sure you remove all double quotes and whitespaces at the beginning and end of the string from each tag. Transform each tag to lower case. If an input string is equal to `[none]`, then return an empty list. Use comprehensions to implement this function.
- Define the `extract_links` function in the `youtube_types.py` module. The function gets the whole description of a video as a string. Sometimes this string contains links of the form `http://my/link/here` or `https://another/link/here`. You must extract all links from the description by means of using regular expressions. We assume that a link is any string that starts with `http://` or `https://` followed by non-whitespace characters. Then, return a list of strings where each item is a link extracted from the description given as input. If there are no links, return an empty list.

Task 4: Read the YouTube Dataset (20%)

Define the function `read_dataset` in the `youtube_core.py` module. The function takes a path to a file as input and returns a list of `Video` objects. To read the file use the `csv` module. Remember to set the

`encoding` parameter to `'utf-8-sig'`. Additionally, use the data extraction functions defined in tasks 2 and 3 to pass the right arguments to the `Video` constructor.

Task 5: Find Viral Videos (20%)

- Define the function `compute_median_diff_dates` in the `youtube_core.py` module. The function takes a list of `Video` objects as input and returns an integer. You must compute the number of days that passed between the publication of a video and its trending date. Then, use the `statistics` module to compute the median of all computed differences. Use comprehensions to solve this problem.
- Define the function `find_viral_videos` in the `youtube_core.py` module. The function takes a list of `Video` objects as input. Then, it finds all the videos whose number of days between publication and trend is less than the median computed with the `compute_median_diff_dates` function. Use comprehensions and generators to solve this problem.

Task 6: Like, Dislike, and Comments Measures (10%)

- Define the function `compute_measures_likes` in the `youtube_core.py` module. The function takes a list of `Video` objects as input and returns a tuple with three values: the median (integer), the mean (float), and the standard deviation (float) of the number of **likes** in the dataset. Use comprehensions and the `statistics` module to compute these values. Float numbers should be rounded to two decimal places.
- Define the function `compute_measures_dislikes` in the `youtube_core.py` module. The function takes a list of `Video` objects as input and returns a tuple with three values: the median (integer), the mean (float), and the standard deviation (float) of the number of **dislikes** in the dataset. Use comprehensions and the `statistics` module to compute these values. Float numbers should be rounded to two decimal places.
- Define the function `compute_measures_comments` in the `youtube_core.py` module. The function takes a list of `Video` objects as input and returns a tuple with three values: the median (integer), the mean (float), and the standard deviation (float) of the number of **comments** in the dataset. Use comprehensions and the `statistics` module to compute these values. Float numbers should be rounded to two decimal places.