Data Mining Techniques

Spring Semester 2022-2023

2nd Exercise-Individual or group work (2 people)

The objective of the task

The purpose of the assignment is to familiarize you with the basic steps of the process followed for the application of data mining techniques, namely: collection, pre-processing / cleaning, transformation, application of data mining techniques and evaluation. The implementation will be done in the Python programming language using the tools/libraries: jupyter notebook, pandas and SciKit Learn.

Description

The**Goodreads**is a social platform in combined with a book database that looks a lot like IMDb for movies: users can search for books, tag them, share their thoughts and discuss. Most importantly, they can rate the books they've read on a scale of 1 to 5 and discover new books to read. Our goal is to study one

subset of data from this platform and create a program that recommends books. Another goal of the paper is to automatically categorize books into their genre based on their descriptions. Finally, those who want can use book covers to build a simple image retrieval system based on its content (***bonus***).

The file you will use is the Best Books Ever Dataset. You can see the file in the zenodo repository at the link below (there you will also find the button to download it)https://zenodo.org/record/4265096#.Y-N2DnbP1jE .

Wanted

First you will import the entire file into a dataframe and study if there are any Nan values in the columns to remove those rows from the dataframe. Then you will write the appropriate commands in python to answer the following requests. What more

times the answers will be given with a graph and you can use any python library you want for this purpose.

Pretreatment (10%)

Notice the ratingsByStars column, it contains 5 values separated by commas, split these values and add to the dataframe separately the ratings, i.e. ratingStar5, ratingStar4, ratingStar3 etc.

Also, the genres column contains more than one genre for each book. Create a new column (name it genreSingle) and put only the first genre of all the genres found in each line. (eg ['Fantasy', 'Young Adult', 'Fiction', 'Magic', 'Childrens', 'Adventure', 'Audiobook', 'Middle Grade', 'Classics', 'Science Fiction Fantasy'] -> the new column will have 'Fantasy') deleting books without any genre information.

Use the publishDate column and create a new column with the publication year of each book (you can use the to_datetime() method provided by pandas or whatever you want).

Questions to study the data - answer 5 of the following (20%)

1. Construct the histogram of the ratings in the data set (use the rating column)

2. What are the 10 books with the most pages.
3. What are the 10 books with the most 5-star ratings (use only the books that have received over 10,000 5-star ratings from the ratingStar5 column).
4. What are the most common words in book titles (after stop words are removed)

5. Who are the 10 authors with the most books
6. Who are the top 10 most reviewed authors (use the numRatings column).

7. Rank authors by their books by year.
8. What are the most common languages in which books are written in your data
9. Who are the 10 publishers who have published the most books.
10. Do the books with the most pages (eg more than 1000 pages) have higher ratings?
11. Gather in a graph or in a table all the unique types of books (genres). What are the most common genres? The same for the awards.
12. Make the wordclouds for the description column. In this query remove the stop words, experiment with the wordcloud parameters and find the most characteristic words used in the books in your data set.

13. How many books are published per year?

Recommendation system implementation (35%)

The goal of such a system is to (1) predict a user's ratings for books he has not yet read, and (2) display a ranked list of the top **N**books we think they'd like to know more about. Another goal of a Recommender is (3) to help users discover relevant books that they would not have found otherwise;

In this query you will need the BookId
columns
Description
And only those lines that have "English" language.

Create him**TF-IDF**(Term Frequency - Inverse Document Frequency) array of unigrams and bigrams for the description column (use TfidfVectorizer's stop_word parameter).

Cosine Similarity: This metric calculates the similarity between two vectors x,y, using the angle between them (when the angle is 0 it means that x and y are equal, if we exclude their length). Go through the TF-IDF table and calculate the similarity of each book to the rest. Store in a python dictionary the 100 most similar books. Prediction: Make a function that takes as input an id and an integer N, and returns the N most similar books.

recommend(item_id = 4085439, num = 5)

The output of the function should be of the following format

Recommending 5 books similar to: The Hunger Games

------ Recommended: NAME
Description: DESCRIPTION
(score:0.12235188993161432)

Recommended: NAME
Description: DESCRIPTION
(score:0.12235188993161432)
.....

Implementation of Classification (35%)

Use the genreSingle column, find the 10 most frequent genres and keep in a new dataframe those books that belong to these 10 most frequent categories. You will need bookId, description and genreSingle. Then clean the description column using the methods we saw in the tutorials (eg removing punctuation, converting all characters to lowercase, etc.). Apply the word2vec method for the descriptions and then using the embeddings to calculate for each description a vector with 200-300 values (features) - this will be the average of the embeddings of the words that make up the description.

*Use Python's pickle library to store attributes in *.pkl files. This way you don't have to recalculate the attributes every time you run your program, but you can just load them into memory using the corresponding method* ***load****.*

Divide the data set into train (80%) and test (20%) using the method ***train_test_split()*** of the librarysklearn.

In this query, your program should be able to find the categories (genre) of the test set (test) using the following Classification methods:

- Naive Bayes
- Support Vector Machines (SVM, experiment with kernel parameters (rbf, linear), c and gamma. Parameter selection can also be done with GridSearchCV)
- Random Forests

All the above models will be trained **MONO** totaltrain and will be evaluated in the test set. You should also evaluate and record the performance of each method using 10-fold Cross Validation using the following metrics:

- Precision / Recall / F-Measure
- Accuracy

At the end prepare a table with the results of your experiments for each metric/ parameter you used.

BONUS - Judging a book by its cover..!

Content-based image retrieval (*Content Based Image Retrieval*) is a technique that uses visual features (such as color, texture, shape) to search

images. Color features are considered to be among the most used low-level features for searching images in large image databases. A color histogram is simply a histogram showing the color level for each individual RGB color channel (where pixel values are in the range 0-255).

In eclass you will find a python notebook which you can use to download all the Best Books Ever Dataset covers in jpg format. Instructions for downloading the images will also be given to you in the tutorials.

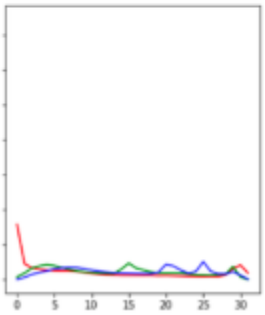
Step one: Use the code we give you to download one locally set of dataset images (you can download eg 400-500. The more images you have, the better results your search will have in the last step). Keep some images for testing later (eg 5%) .

Step Two: Calculate a histogram for each image. For this purpose it will use the OpenCV library and specifically the calcHist method.

cv2.calcHist([images], [channels], [mask], [bins], [hist_range])

imagesis the image in formBGR. This argument expects a list of images, so we've enclosed an image in brackets [] if it's an image.
channelsis the color channel (BGR) for which we want to create a histogram - we do this for a single channel at a time.
maskis another image array consisting of values0 and 1 which allow us to mask (eg hide) part of the images if we want. We won't use this, so we set it to the value**None**.

binsis the number of histogram bars we place our values on. We can set this to 256 if we want to keep all the original values, but we recommend 32 or 8 bins for performance reasons.
hist_rangeis the range of color values we expect. As we use RGB/BGR, we expect a minimum value of 0 and a maximum value of 255, so we write [0, 256].



You can create a histogram for each color channel (BGR) - we do this for a single channel at a time and then merge the results.

After converting our images into three vectors representing the three color channels, you need to composite these three vectors into a single vector. This process will be done for each image and what will result is a dataframe with the histograms for each image and an identifier id so that you can distinguish them.

Step Three: Choose one image from the test set and calculate for that too hertogram.

Step Four: Search for the closest images in the set of images by comparing them histograms with either Euclidean distance or cosine similarity. For each question, display the 4 closest images. Use matplotlib to display the results. What you need to know about this library is that it loads images in format**B**lue**G**reen**R**ed (BGR) and not the classic RGB mentioned earlier. You will need to transform each image (mainly flipping with the flip() command of the numpy library) to display correctly in the output.