Approximate Nearest Neighbors:

Image Recommendation System via Collaborative Filtering

Please read the instructions very carefully

This is a modified version of the previous question and requires you to use an artificial nearest neighbors library

We suggest you to use one of the following:

- ScaNN
- FAISS
- Annoy

1. Assignment must be implemented in Python 3 only.
2. You are allowed to use libraries for data preprocessing (numpy, pandas, nltk etc) and for evaluation metrics, data visualization (matplotlib etc.).
3. You will be evaluated not just on the overall performance of the model and also on the experimentation with hyper parameters, data prepossessing techniques etc.
4. ⚠ The Assignment will be evaluated automatically. Please adhere to taking proper inputs from `config.csv` file. You can change your `config.csv` file to experiment with your code. But at the end, make sure that your outputs are corresponding to input values in `config.csv`
5. Strict plagiarism checking will be done. An F will be awarded for plagiarism.

About the Dataset

Behance is a community art website where users showcase and discover creative work. Each user is able to "appreciate" (equivalent to a "like" on Instagram or a "react" on Facebook) an image, indicating that they like the image. It is in the website's best interests to show users pictures that they would like, to keep them engaged for longer. For this question, given a set of pictures that a user has already appreciated, you have to show them a new picture that they would like based on what similar users appreciated.

The dataset has information of 1 million appreciates of 63,497 users on 178,788 items. The file Behance appreciate 1M has a triplet in each line in the form of (user id, item id, unix

timestamp).
Task: Take the inputs from the config.csv file and output the recommendations for a particular person

- Collaborative Filtering is a way to predict items to the user based on the the user's history and the history of similar users. The similarity between users can be quantified by the number of images that both the users appreciated.
- The images appreciated by a similar user would be the most suitable images to show a user. Since we can find the similarity between any two users, we would be able to find the "nearest" neighbours of any user, allowing us to use a KNN-based algorithm to recommend new images to a user.
- Since people do not like seeing pictures that they have seen already. Make sure that you do not recommend pictures that a user has appreciated already.
- Output the final response will be saved in the file named `config['output_file']` .

Output file format: Populate the output file with images that the user has not seen of the k most similar users, in descending order of their similarity. Each line in the output file should be a duplet in the form of (item id, user id), where the user id is the id of the kth similar user. The order of the images corresponding to the same similar user would not matter. The output file would look something like this:

```
item_id_1_of_1st_similar_user 1st_most_similar_user_id
item_id_2_of_1st_similar_user 1st_most_similar_user_id
item_id_3_of_1st_similar_user 1st_most_similar_user_id
...
item_id_1_of_2nd_similar_user 2nd_most_similar_user_id
item_id_2_of_2nd_similar_user 2nd_most_similar_user_id
item_id_3_of_2nd_similar_user 2nd_most_similar_user_id
...
item_id_1_of_kth_similar_user kth_most_similar_user_id
item_id_2_of_kth_similar_user kth_most_similar_user_id
item_id_3_of_kth_similar_user kth_most_similar_user_id
```

You may use any other recommendation system that you wish to use. However, evaluation script will score your submission by measuring the similarity between users with the number of common images they appreciated. The dataset was extracted using Behance's API as a part of the paper "Vista: A visually, socially, and temporally-aware model for artistic recommendation, RecSys, 2016". Check out this Google Drive folder for more information about the dataset.

Have fun! The users are waiting to see new pictures!

Import necessary libraries

In [ ]:
```
import numpy as np
import pandas as pd
```

In [ ]:
```
# Config Generation Sample Code.
# ⚠ Only for experimentation on your side.
# ⚠ Should be commented during submission.

# df = pd.DataFrame(data=[{'id':276633,
#                          'k':5,
#                          'dataset_file':'./Behance_appreciate_1M',
#                          'output_file':'./output.txt'}])
# df.to_csv('config.csv')
```

In [ ]:
```
config = pd.read_csv('config.csv').iloc[0]
```

In [ ]:
```
config
```

In [ ]:
```
user = config['id']
k_value = config['k']
```

Read the Data

In [ ]:
```
with open(config['dataset_file'], 'r') as inFile:
    appreciate_data = inFile.readlines()
```

In [ ]:
```
# your code here
```

Initialize a dictionary to store the item_ids that a user likes

Go through each line of the input file and construct the user_likes dictionary

In [ ]:
```
user_likes = dict()
```

In [ ]:
```
for line in appreciate_data:
    line = line.strip()

    user_id = int(line.split()[0])
    item_id = int(line.split()[1])

    if user_id not in user_likes:
        user_likes[user_id] = list()

    user_likes[user_id].append(item_id)
```

Use your choice of Approximate Nearest Neigbor after Collaborative Filtering to find nearest neighbors

In [ ]:
```
# your code here
def neighbors(user,k_value):
    """ returns an iterable object (like list or generator) """
    pass
```

Answer the following questions:

Q1. **Explain how your choice of library works**

your solution here

Q2. **Compare your choice of library with vanilla KNN.**

Hint: Include Time Complexity, and explain the tradeoff with recall

your solution here

Q3. **Compare your choice of library with implementation of ScaNN, faiss and annoy.**

Hint: Include Time Complexity, and explain the tradeoff with recall

your solution here

Open the output file to write all the lines to the file

In [ ]:

```
outFile = open(config['output_file'], 'w')


for n_user in neighbors(user,k_value):
    user_id = list(user_likes.keys())[n_user]
    for item_id in user_likes[user_id]:
        outFile.write(str(item_id) + ' ' + str(user_id) + '\n')


outFile.close()
```