Recommending songs

In this module, we focused on building recommender systems to find products, music and movies that interest users. We also built an exciting iPython notebook for recommending songs, which compared the simple popularity-based recommendation with a personalized model, and showed the significant improvement provided by personalization.

In this assignment, we are going to explore the song data and the recommendations made by our model. In the process, you are going to learn how to use one of the most important data manipulation primitives, *groupby.* These techniques will be important to building the intelligent application in your capstone project.

Follow the rest of the instructions on this page to complete your program. When you are done, ***instead of uploading your code, you will answer a series of quiz questions*** (see the quiz after this reading) to document your completion of this assignment. The instructions will indicate what data to collect for answering the quiz.

Learning outcomes

- Execute song recommendation code with the iPython notebook
- Load and transform real, song data
- Build a song recommender model
- Use the model to recommend songs to individual users
- Use groupby to compute aggregate statistics of the data

Resources you will need

You will need to install the software tools or use the free Amazon EC2 machine. Instructions for both options are provided in the reading for Module 1.

Download the data and starter code

Before getting started, you will need to download the dataset and the starter iPython notebook that we used in the module.

- Download the music listening dataset with articles on famous people here in SFrame format: **song_data.gl.zip**
- Download the song recommendation notebook from the module here: Song recommender.ipynb
- Save both of these files in the same directory (where you are calling iPython notebook from) and unzip the data file. **Not sure where to save the files? See this guide.**

Now you are ready to get started!

Note: If you would rather use other ML tools...

You are welcome to use any ML tool for this course, such as scikit-learn. Though, as discussed in the intro module, *we strongly recommend you use IPython Notebook and GraphLab Create. (GraphLab Create is free for academic purposes.)*

If you are choosing to use other packages, we still recommend you use SFrame, which will allow you to scale to much larger datasets than Pandas. (Though, it's possible to use Pandas in this course, if your machine has sufficient memory.) The SFrame package is available in open-source under a permissive BSD license. So, you will always be able to use SFrames for free.

If you are not using SFrame, here is the dataset for this assignment in CSV format, so you can use Pandas or other options out there: song_data.csv

Watch the video and explore the iPython notebook on recommending songs

If you haven't done so yet, before you start, we recommend you watch the video where we go over the iPython notebook on song recommendation from this module. You can then open up the iPython notebook we used and familiarize yourself with the steps we covered in this example.

What you will do

Now you are ready! We are going do three tasks in this assignment. There are several results you need to gather along the way to enter into the quiz after this reading.

1. **Counting unique users:** The method *.unique()* can be used to select the unique elements in a column of data. In this question, you will compute the number of unique users who have listened to songs by various artists. For example, to find out the number of unique users who listened to songs by 'Kanye West', all you need to do is select the rows of the song data where the artist is 'Kanye West', and then count the number of unique entries in the 'user_id' column. Compute the number of unique users for each of these artists: 'Kanye West', 'Foo Fighters', 'Taylor Swift' and 'Lady GaGa'. ***Save these results to answer the quiz at the end.***
2. **Using groupby-aggregate to find the most popular and least popular artist:** each row of *song_data* contains the number of times a user listened to particular song by a particular artist. If we would like to know how many times any song by 'Kanye West' was listened to, we need to select all the rows where 'artist'=='Kanye West' and sum the 'listen_count' column. If we would like to find the most popular artist, we would need to follow this procedure for each artist, which would be very slow. Instead, you will learn about a very important method:

```
.groupby()
```

You can read the documentation about groupby here. The *.groupby* method computes an aggregate (in our case, the sum of the 'listen_count') for each distinct value in a column (in our case, the 'artist' column).

Follow these steps to find the most popular artist in the dataset:

- The *.groupby* method has two important parameters:

i. *key_columns*, which takes the column we want to group, in our case, 'artist'

ii. *operations*, where we define the aggregation operation we using, in our case, we want to sum over the 'listen_count'.

- With this in mind, the following command will compute the sum *listen_count* for each artist and return an SFrame with the results:

```
song_data.groupby(key_columns='artist', operations={'total_count': graphlab.aggregate.SUM('li
sten_count')})
```

the total number of listens for each artist will be stored in *'total_count'*.

- Sort the resulting SFrame according to the *'total_count'*, and find the artist with the most popular and least popular artist in the dataset. ***Save these results to answer the quiz at the end.***

3. **Using groupby-aggregate to find the most recommended songs:** Now that we learned how to use *.groupby()* to compute aggregates for each value in a column, let's use to find the song that is most recommended by the *personalized_model* model we learned in the iPython notebook above. Follow these steps to find the most recommended song:

- Split the data into 80% training, 20% testing, using *seed=0*, as was done in the iPython notebook above.
- Train an *item_similarity_recommender*, as done in the iPython notebook, using the training data.
- Next, we are going to make recommendations for the users in the test data, but there are over 200,000 users (58,628 unique users) in the test set. Computing recommendations for these many users can be slow in some computers. Thus, we will use only the first 10,000 users only in this question. Using this command to select this subset of users:

```
subset_test_users = test_data['user_id'].unique()[0:10000]
```

- Let's compute one recommended song for each of these test users. Use this command to compute these recommendations:

```
personalized_model.recommend(subset_test_users,k=1)
```

- Finally, we can use *.groupby()* to find the most recommended song! :) When we used *.groupby()* in the previous question, we summed up the total *'listen_count'* for each artist, by setting the parameter SUM in the aggregator:

```
operations={'total_count': graphlab.aggregate.SUM('listen_count')}
```

For this question, we simply want to count how often each song is recommended, so we will use the COUNT aggregator instead of SUM, and store the results in a column we will call 'count' by using:

```
operations={'count': graphlab.aggregate.COUNT()}
```

And, since we want to use the song titles as the key to the aggregator instead of of the 'artist', we use:

```
key_columns='song'
```

- By sorting the results, you will find out the most recommended song to the first 10,000 users in the test data! ***Save these results to answer the quiz at the end.***