jupyter AnalyzingRecommenderMLalgorithms Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Python [default]

Building and analyzing ML recommender algorithm.

author: bhavesh patel

There are many different models. Here are the few we will work on.

First and simplest: popularity model. This is where you see most popular items. It is not personalized.

Second: Recommendation based on past purchase history and other info like demographics etc.

Third: People who bought also bought. This is collaborative filtering model. This is where we build co-occurrence matrix

Fourth: Matrix factorization to find predict hidden values (e.g. people never bought that item). This works as long as somebody else bought the item.

Measuring performance of different models.

recall = number of items liked and recommended / Total number of items liked

Precision = number of items liked and recommended / total number of items recommended

Ideally, you want both recall and precision to be 1 (100%)

In reality, precision goes down as you try to recall more items as you have fewer data points.

We can compare recommender system by finding out AUC (Area under curve). More the better.

In [1]:
```
import graphlab
```

In [2]:
```
# Limit number of worker processes. This preserves system memory, which prevents hosted notebooks from crashing.
graphlab.set_runtime_config('GRAPHLAB_DEFAULT_NUM_PYLAMBDA_WORKERS', 4)
```

```
[INFO] graphlab.cython.cy_server: GraphLab Create v2.1 started. Logging: /tmp/graphlab_server_1481401877.log
```

```
This non-commercial license of GraphLab Create for academic use is assigned to bhaveshhk8@gmail.com and will expire o
n October 17, 2017.
```

In [3]:
```
#Load the data.

song_data = graphlab.SFrame('song_data.gl/')
```

In [4]:
```
# review data.

song_data.head()
```

Out[4]:

user_id	song_id	listen_count	title	artist
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	SOAKIMP12A8C130995	1	The Cove	Jack Johnson
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	SOBBMDR12A8C13253B	2	Entre Dos Aguas	Paco De Lucia
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	SOBXHDL12A81C204C0	1	Stronger	Kanye West
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	SOBYHAJ12A6701BF1D	1	Constellations	Jack Johnson
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	SODACBL12A8C13C273	1	Learn To Fly	Foo Fighters
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	SODDNQT12A6D4F5F7E	5	Apuesta Por El Rock 'N' Roll ...	Héroes del Silencio
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	SODXRTY12AB0180F3B	1	Paper Gangsta	Lady GaGa
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	SOFGUAY12AB017B0A8	1	Stacked Actors	Foo Fighters
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	SOFRQTD12A81C233C0	1	Sehr kosmisch	Harmonia
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	SOHQWYZ12A6D4FA701	1	Heaven's gonna burn your eyes ...	Thievery Corporation feat. Emiliana Torrini ...

song
The Cove - Jack Johnson
Entre Dos Aguas - Paco De Lucia ...
Stronger - Kanye West
Constellations - Jack Johnson ...
Learn To Fly - Foo

Fighters ...
Apuesta Por El Rock 'N' Roll - Héroes del ...
Paper Gangsta - Lady GaGa
Stacked Actors - Foo Fighters ...
Sehr kosmisch - Harmonia
Heaven's gonna burn your eyes - Thievery ...

[10 rows x 6 columns]

In [5]:
```
# make graph local.

graphlab.canvas.set_target('ipynb')
```

In [6]:
```
song_data.show()
```

user_id

dtype:	str
num_unique (est.):	66,019
num_undefined:	0

frequent items:

No values appear with ≥ 0.01% occurrence.

song_id

dtype:	str
num_unique (est.):	9,971
num_undefined:	0

frequent items:

SOFRQTD12A81C233C0
SOAUWYT12A81C206F1
SOBONKR12A58A7A7E0
SOAXGDH12A8C13F8A1
SOSXLTC12AF72A7F54
SOEGIYH12A6D4FC0E3
SONYKOW12AB01849...
SOFLJQZ12A6D4FADA6
SOLFXKT12AB017E3E0
SODJWHY12A8C142C...
SOUVTSM12AC468F6A7
SOUSMXX12AB0185C24

listen_count

dtype:	int
num_unique (est.):	276
num_undefined:	0
min:	1
max:	920
median:	1
mean:	3.291
std:	7.203

distribution of values:

title

dtype:	str
num_unique (est.):	9,540
num_undefined:	0

frequent items:

Sehr kosmisch
Undo
You're The One
Dog Days Are Over ...
Revelry
Horn Concerto No. ...
Secrets
Tive Sim
Fireflies
Hey_ Soul Sister
Drop The World
OMG

artist

dtype:	str
num_unique (est.):	3,371
num_undefined:	0

frequent items:

Coldplay
Florence + The ...
Kings Of Leon
Justin Bieber
The Black Keys
Jack Johnson
Train
Eminem
OneRepublic
Radiohead
Muse
Daft Punk

In [7]:
```
# number of records.

len(song_data)
```

Out[7]: 1116609

In [8]:
```
# 1.1 million songs.  How about number of users?

song_data['user_id'].unique()
```

Out[8]:
```
dtype: str
Rows: 66346
['c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8', '279292bb36dbfc7f505e36ebf038c81eb1d1d63e', 'c067c22072a17d33310d7223d7b
79f819e48cf42', 'f6c596a519698c97f1591ad89f540d76f6a04f1a', '696787172dd3f5169dc94deef97e427cee86147d', '3a7111f4cdf3
c5a85fd4053e3cc2333562e1e0cb', '31f6fd9d9936adb9f7d0f157fd960c0a676ccfd6', '532e98155cbfd1e1a474a28ed96e59e50f7c5ba
f', 'ee43b175ed753b2e2bce806c903d4661ad351a91', 'e372c27f6cb071518ae500589ae02c126954c148', '83b1428917b47a6b130ed471
b09033820be78a8c', '9b10c5b0569c679d9d7e258c37c0acb99cbbce02', '39487deef9345b1e22881245cabf4e7c53b6cf6e', '88325c1fc
54d4227b223a7ca7c68c2bdc39df54b', '18325842a941bc58449ee71d659a08d1c1bd2383', '3f2a5636078d12dab36f61f92efc1fc62f72f8
15', '75864fe1e1674afe546612ecfd2217f062610a79', 'b472a8e0407792249a28e8a24f0f82e6cc860822', 'b2b08a84654f90585e0dc3c
5155d330fb8e4ba59', 'ca60f0fa15783aac3827a9edb2e1b51ef3a4fe19', '507433946f534f5d25ad1be302edb9a2376f503c', '2e9cbd20
0d67910aeb97b36efe9cd4341cc06030', '647811338b3bed47a2a374e7d0dedb5f799188a3', '8b6cd3340224d31ffab18ba2731feebb9df73
043', '18fafad477f9d72ff86f7d0bd838a6573de0f64a', 'fe85b96ba1983219b296f6b4869dd29eb2b72ff9', '046311081a8703bdee3be2
e6d9da07ccdc135340', 'cc3ca967889ae3343e1325aef7c322bd8a830865', '225ea420b4bede50919d1bfe24a599691522d176', '95dc7e2
b188b1148b2d25f4e6b6e94afacc4efc3', '8d14fcfb9d84cc6f696ca533fd97d8ade9f150dc', '2a806a33a9b8394ea22a14a06eafeb4b0aef
0af3', '4a3a1ae2748f12f7ab921a47d6d79abf82e3e325', 'a2c1a593432f5e19a9174eb1b3b57e02d3212eb6', 'f9958d5c8e88f53bbbf6a
5b82d3062b369497f64', 'c6a3854bd30fe15382815ca2c9196a1e15d01a39', 'c6d5086d22ba5a9c205877770f29bf97e3a5993b', '9bd4c3
ace8717adbd277541d979af6acf6de54f6', '62c4bf887b7b1e5cf6ab62723481099c7f98377e', '0285d30c0892a0bd6169e018203da1e2c43
65f21', '9fd403cf953d4bc8f77980f2bd9dbb174a567d15', 'e6a3b65d0f0d2d5f4b7b8d06a6a1bf7a2e844eb9', 'b7b5408ba99a68a4d8d9
df6007e8516a8fabb43f', '92337c69d4ff1c13f5c9ad4e9c62a0654be9d230', 'e8813fe73c90d69b4f744251c68fce896ec2aede', '5eb76
1c242ec9d014a4c3f79d1496c342b2bf4f6', '36ba692bb04359e9cd3efff4cd3dfd12b6204883', '1c4735ef0f9e2b95524f8c92a70be8b355
eeb651', '8eadbac97d6679515b8fe858d4d210a5249a224f', '9f6166dfd0e587b408c30aeb3cf694240262eaae', '23c026e0dd4526a2fd3
533281f12520144da5a46', 'a42fdb32bf733517fb5be88ad6fd428cb0e68a4c', 'ccc4a6404ec1d4d7ab3add04eeacf79e4ef655de', '486f
c2c57f6a4213bee54f9323a6668a7a306cc3', '40eb78cc05195299242ceb72873377a6cc37b721', '66515166dc465cb51af511db9f6e70809
5af31a6', '88b7f1ddc21c9f83e92e3a3f28650b44ab28ac98', 'a57f4ce047f84f3ffa21a0119cc8b7de548ac79a', '11e266d0d2d7a841bd
1a2604b14cc05f4bcecd8e', '557af535722c4e6edab88151c354381d85dcf8bd', '5827384be24edd3bd6b981eb1f40950e1d985d02', '332
c6355122179f10786c61fff699057b36a15c0', '98643f2e574f6d3e407c5f25bf5029916910fdd0', '2955374e2e24cd324c3fc41ecad45126
40ebcac0', '4886fb3ba039dc4b6d0c6019276c000f7908b709', 'a3744fdc055bff8cc27ab95acfc1c0f39f89ba61', 'e1deb0afe4e908da2
0e96879a39601e28760d62f', 'b0983659c8a8b193166871fcc4a26d46b2b50ba3', '82c458f2e9bc30f87890b408e5b965114989edf4', '79
f8ebdfb8ee78bef9b43251582dd329f0301e7a', '632af0475de5f2d90838520df96e4b681a383114', '7e7a4eede58db53c0b4837b5b302843
d845259e8', '91ab14dcd7173ff5df8c38f2584477180b92135d', '0bfa607c9dcf72df25ccc748e147c42d203c6a89', '181d9c254ff957a3
96b8bbcec10228f84cfa5c2c', 'a58941e09a29f9f540e8af82386f08b09b6ebc05', '53ae48ebbe6e1fb426cd1f3e5be4e343e76abf51', '6
51d28cb6ce2a60c14faf8e510e94016cc54499d', 'a812535294328122910d15e4dbe7084140266938', 'c46038495c5f7fe43b12c3cea01251
7998fd78a0', '751774a078c559b9abf6aceaec9062c4213a92ac', 'a2498682dfb521861f0956256feaa338069cf303', 'f9c56a1a222ce6c
e59021718916cce519d70d5b4', '37b97745b5649ca367aab9e4999528b470ecc692', '09790d63be53c025db51c39423ec6ad844b81f1f',
 '0a8befc6cf4978022588f13c379786101dfc67cd', 'd6dbb0578fe5baf513cad07fbf07801f89fd9313', 'e7e734371172b30fa331332c599
84c7334b6a025', '7563909660feac4f23500550c6a813eab952d4c2', '99d724639a1e51f53770236f24c87e2ca9126ec6', '64d4b02087e9
433c4a82aeba087cd8967453afa1', '3a67a46baabdc2f9bc2cd99370d8ae577068dca1', '9db1d7471ee3e8c1de53995435fd663a46ef972
5', 'd95df4df9d6f25e76f13556ce6cf5b50f979c76c', '2e35cbc9c99533fd2757483f46329b41b0d646dc', '58fcc1ee8eb4bff5a9bedff6
e7510f0167367c1a', '77d0df4368fe8fa3873ba387ca02a36da9c14299', 'cc1c5e821a36a788f9fab1834746a96089f7e972', 'bfe25236e
58ac7915762fdcee75cfe8f72b062ac', '8ee5a2fbe1fea60a7148e0b50ac7fb4d87a81119', ... ]
```

In [9]:
```
# Total 1.1 million songs rated by 66,000 users.
# Now let's find out most popular artist.
```

```
# We can do this by how many users listened to song by an artist.

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

In [10]:
```
popular_artist=popular_artist.sort('total_count', ascending=False)
```

In [11]:
```
popular_artist.head()
```

Out[11]:

artist	total_count
Kings Of Leon	43218
Dwight Yoakam	40619
Björk	38889
Coldplay	35362
Florence + The Machine	33387
Justin Bieber	29715
Alliance Ethnik	26689
OneRepublic	25754
Train	25402
The Black Keys	22184

[10 rows x 2 columns]

In [12]:
```
# first create training and test data set.

train_data, test_data = song_data.random_split(0.8, seed=0)
```

Item Similarity ML algorithm.

In [13]:
```
sim_item_song_reco_model = graphlab.item_similarity_recommender.create(train_data,
                                                           user_id='user_id',
                                                           item_id='song')
```

```
Recsys training: model = item_similarity
Warning: Ignoring columns song_id, listen_count, title, artist;
    To use one of these as a target column, set target =
    and use a method that allows the use of a target.
Preparing data set.
    Data has 893580 observations with 66085 users and 9952 items.
    Data prepared in: 1.2356s
Training model from provided data.
Gathering per-item and per-user statistics.
+--------------------------------+------------+
| Elapsed Time (Item Statistics) | % Complete |
+--------------------------------+------------+
| 1.428ms                        | 1.5        |
| 54.173ms                       | 100        |
+--------------------------------+------------+
Setting up lookup tables.
Processing data in one pass using dense lookup tables.
+-------------------------------------+------------------+-----------------+
| Elapsed Time (Constructing Lookups) | Total % Complete | Items Processed |
+-------------------------------------+------------------+-----------------+
| 507.384ms                           | 0                | 0               |
| 2.32s                               | 100              | 9952            |
+-------------------------------------+------------------+-----------------+
Finalizing lookup tables.
Generating candidate set for working with new users.
Finished training in 3.41233s
```

In [14]:
```
# let's check for the first user.

sim_item_song_reco_model.recommend(users=[song_data['user_id'][0]])
```

Out[14]:

user_id	song	score	rank
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	Meadowlarks - Fleet Foxes	0.0248072429707	1
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	Quiet Houses - Fleet Foxes ...	0.0240329645182	2
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	Heard Them Stirring - Fleet Foxes ...	0.0203885561542	3
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	Tiger Mountain Peasant Song - Fleet Foxes ...	0.0199806752958	4

b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	Your Protector - Fleet Foxes ...	0.0193978893129	5
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	Oliver James - Fleet Foxes ...	0.0190611293441	6
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	Great Indoors - John Mayer ...	0.0149489750988	7
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	Innocent Son - Fleet Foxes ...	0.0148925859677	8
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	White Winter Hymnal - Fleet Foxes ...	0.0148194040123	9
b80344d063b5ccb3212f76538 f3d9e43d87dca9e ...	City Love - John Mayer	0.0138473055865	10

[10 rows x 4 columns]

In [15]:
```
# now let's for 1000th user.

sim_item_song_reco_model.recommend(users=[song_data['user_id'][1000]])
```

Out[15]:

user_id	song	score	rank
20d0638c7ada27ac12346b0ed 5ab99b39524291d ...	Lights & Music - Cut Copy	0.00933748483658	1
20d0638c7ada27ac12346b0ed 5ab99b39524291d ...	Oh! - Boys Noize	0.00898901266711	2
20d0638c7ada27ac12346b0ed 5ab99b39524291d ...	Waters Of Nazareth (album version) - Justice ...	0.00894576098238	3
20d0638c7ada27ac12346b0ed 5ab99b39524291d ...	Strangers In The Wind - Cut Copy ...	0.00881623370307	4
20d0638c7ada27ac12346b0ed 5ab99b39524291d ...	Auto-Dub - Skream	0.00863063548292	5
20d0638c7ada27ac12346b0ed 5ab99b39524291d ...	Thrills - LCD Soundsystem	0.00838310803686	6
20d0638c7ada27ac12346b0ed 5ab99b39524291d ...	Clock - Simian Mobile Disco ...	0.00832954687732	7
20d0638c7ada27ac12346b0ed 5ab99b39524291d ...	On Repeat - LCD Soundsystem ...	0.00831711079393	8
20d0638c7ada27ac12346b0ed 5ab99b39524291d ...	Lava Lava - Boys Noize	0.00791249317782	9
20d0638c7ada27ac12346b0ed 5ab99b39524291d ...	Shine Shine - Boys Noize	0.00781313436372	10

[10 rows x 4 columns]

In [16]:
```
# Now let's figure which song is more recommended.
# We have 225,000 recommendation among 66,000 unique users.
# We can limit users with following command.
# But I believe my computer is powerful enough!
#subset_users=test_data['user_id'].unique()[0:25000]
```

In [17]:
```
# So let's run model on all users.
sim_item_song_reco_model.recommend(test_data['user_id'],k=1)
```

```
recommendations finished on 1000/223029 queries. users per second: 20581.2
recommendations finished on 2000/223029 queries. users per second: 19247.8
recommendations finished on 3000/223029 queries. users per second: 20230.4
recommendations finished on 4000/223029 queries. users per second: 20082.4
recommendations finished on 5000/223029 queries. users per second: 19705.8
recommendations finished on 6000/223029 queries. users per second: 17010.3
recommendations finished on 7000/223029 queries. users per second: 16594.6
recommendations finished on 8000/223029 queries. users per second: 16181.1
recommendations finished on 9000/223029 queries. users per second: 15592.8
recommendations finished on 10000/223029 queries. users per second: 15272.8
recommendations finished on 11000/223029 queries. users per second: 15068.2
```

In [18]:
```
# now let's find out which song was recommended by our recommender.
# we should count the number of times each song is recommended.
# then we can sort to find out most recommended song and least recommendded song.
rec_song=song_data.groupby(key_columns='song', operations={'count': graphlab.aggregate.COUNT()})
```

In [19]:
```
# now let's sort the data to find out most recommended song.

rec_song=rec_song.sort('count', ascending=False)
```

In [21]:
```
# Now let's view which song was most recommended by our ML algorithm.

rec_song.head()
```

Out[21]:

song	count
Sehr kosmisch - Harmonia	5970
Undo - Björk	5281
You're The One - Dwight Yoakam ...	4806
Dog Days Are Over (Radio	4536

Dog Days Are Over (Radio Edit) - Florence + The ...	[illegible]
Revelry - Kings Of Leon	4339
Horn Concerto No. 4 in E flat K495: II. Romance ...	3949
Secrets - OneRepublic	3916
Tive Sim - Cartola	3185
Fireflies - Charttraxx Karaoke ...	3171
Hey_ Soul Sister - Train	3132

[10 rows x 2 columns]

```
In [ ]:
```