Building a song recommender

Fire up GraphLab Create

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

Load music data

In [2]:
```
song_data = graphlab.SFrame('song_data.gl/')
```

```
[INFO] graphlab.cython.cy_server: GraphLab Create v2.1 started. Logging: /tm
p/graphlab_server_1470185075.log

This non-commercial license of GraphLab Create for academic use is assigned t
o robertmckee@utexas.edu and will expire on July 10, 2017.
```

Explore data

Music data shows how many times a user listened to a song, as well as the details of the song.

In [3]:
```
song_data.head()
```

Out[3]:

user_id	song_id	listen_count	title
b80344d063b5ccb3212f76538f3d9e43d87dca9e ...	SOAKIMP12A8C130995	1	The Cove
b80344d063b5ccb3212f76538f3d9e43d87dca9e ...	SOBBMDR12A8C13253B	2	Entre Dos Aguas
b80344d063b5ccb3212f76538f3d9e43d87dca9e ...	SOBXHDL12A81C204C0	1	Stronger
b80344d063b5ccb3212f76538f3d9e43d87dca9e ...	SOBYHAJ12A6701BF1D	1	Constellations
b80344d063b5ccb3212f76538f3d9e43d87dca9e ...	SODACBL12A8C13C273	1	Learn To Fly
b80344d063b5ccb3212f76538f3d9e43d87dca9e ...	SODDNQT12A6D4F5F7E	5	Apuesta Por El Rock 'N' Roll ...
b80344d063b5ccb3212f76538f3d9e43d87dca9e ...	SODXRTY12AB0180F3B	1	Paper Gangsta
b80344d063b5ccb3212f76538f3d9e43d87dca9e ...	SOFGUAY12AB017B0A8	1	Stacked Actors
b80344d063b5ccb3212f76538f3d9e43d87dca9e ...	SOFRQTD12A81C233C0	1	Sehr kosmisch
b80344d063b5ccb3212f76538f3d9e43d87dca9e ...	SOHQWYZ12A6D4FA701	1	Heaven's gonna burn your eyes ...

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 ...

Showing the most popular songs in the dataset

In [4]:
```
graphlab.canvas.set_target('ipynb')
```

In [5]:
```
song_data['song'].show()
```

Most frequent items from *<SArray>*

Value	Count	Percent	
Sehr kosmisch - ...	5,970	0.535%	
Undo - Björk	5,281	0.473%	
You're The One - ...	4,806	0.43%	
Dog Days Are Over ...	4,536	0.406%	
Revelry - Kings Of ...	4,339	0.389%	
Horn Concerto No. ...	3,949	0.354%	
Secrets - ...	3,916	0.351%	
Tive Sim - Cartola	3,185	0.285%	
Fireflies - ...	3,171	0.284%	
Hey_ Soul Sister - ...	3,132	0.28%	
Drop The World - ...	2,570	0.23%	
OMG - Usher ...	2,533	0.227%	
Catch You Baby ...	2,393	0.214%	
Marry Me - Train	2,300	0.206%	
Use Somebody - ...	2,253	0.202%	
Canada - Five Iron ...	2,215	0.198%	
Sincerité Et ...	2,111	0.189%	
Représente - ...	2,106	0.189%	
Ain't Misbehavin - ...	2,006	0.18%	
Invalid - Tub Ring	1,986	0.178%	
Billionaire [feat. ...	1,954	0.175%	
Pursuit Of ...	1,923	0.172%	
Alejandro - Lady ...	1,853	0.166%	
The Scientist - ...	1,852	0.166%	
Just Dance - Lady ...	1,757	0.157%	
Somebody To Love - ...	1,754	0.157%	
Lucky (Album ...	1,736	0.155%	
Bulletproof - La ...	1,724	0.154%	
Clocks - Coldplay	1,681	0.151%	
I Gotta Feeling - ...	1,651	0.148%	

Show More

In [6]:
```
len(song_data)
```

Out[6]: 1116609

Count number of unique users in the dataset

In [7]:
```
users = song_data['user_id'].unique()
```

In [8]:
```
len(users)
```

Out[8]: 66346

Create a song recommender

In [9]:
```
train_data,test_data = song_data.random_split(.8,seed=0)
```

Simple popularity-based recommender

In [10]:
```
popularity_model = graphlab.popularity_recommender.create(train_data,
                                                          user_id='user_id',
                                                          item_id='song')
```

```
Recsys training: model = popularity
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: 1.35876s
893580 observations to process; with 9952 unique items.
```

Use the popularity model to make some predictions

A popularity model makes the same prediction for all users, so provides no personalization.

In [11]:
```
popularity_model.recommend(users=[users[0]])
```

Out[11]:

user_id	song	score	rank
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Sehr kosmisch - Harmonia	4754.0	1
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Undo - Björk	4227.0	2
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	You're The One - Dwight Yoakam ...	3781.0	3
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Dog Days Are Over (Radio Edit) - Florence + The ...	3633.0	4
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Revelry - Kings Of Leon	3527.0	5
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Horn Concerto No. 4 in E flat K495: II. Romance ...	3161.0	6
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Secrets - OneRepublic	3148.0	7
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Fireflies - Charttraxx Karaoke ...	2532.0	8
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Tive Sim - Cartola	2521.0	9
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Drop The World - Lil Wayne / Eminem ...	2053.0	10

[10 rows x 4 columns]

In [12]:
```
popularity_model.recommend(users=[users[1]])
```

Out[12]:

user_id	song	score	rank
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Sehr kosmisch - Harmonia	4754.0	1
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Undo - Björk	4227.0	2
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	You're The One - Dwight Yoakam ...	3781.0	3
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Dog Days Are Over (Radio Edit) - Florence + The ...	3633.0	4
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Revelry - Kings Of Leon	3527.0	5
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Horn Concerto No. 4 in E flat K495: II. Romance ...	3161.0	6
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Secrets - OneRepublic	3148.0	7
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Hey_ Soul Sister - Train	2538.0	8
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Fireflies - Charttraxx Karaoke ...	2532.0	9
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Tive Sim - Cartola	2521.0	10

[10 rows x 4 columns]

Build a song recommender with personalization

We now create a model that allows us to make personalized recommendations to each user.

In [13]:
```
personalized_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: 1.47619s
Training model from provided data.
Gathering per-item and per-user statistics.
+--------------------------------+------------+
| Elapsed Time (Item Statistics) | % Complete |
+--------------------------------+------------+
| 3.143ms                        | 4.5        |
| 52.636ms                       | 100        |
+--------------------------------+------------+
Setting up lookup tables.
Processing data in one pass using dense lookup tables.
+-------------------------------------+------------------+-----------------+
| Elapsed Time (Constructing Lookups) | Total % Complete | Items Processed |
+-------------------------------------+------------------+-----------------+
| 599.499ms                           | 0                | 0               |
| 2.11s                               | 100              | 9952            |
+-------------------------------------+------------------+-----------------+
Finalizing lookup tables.
Generating candidate set for working with new users.
Finished training in 3.30214s
```

Applying the personalized model to make song recommendations

As you can see, different users get different recommendations now.

In [19]:
```
personalized_model.recommend(users=[users[0]])
```

Out[19]:

user_id	song	score	rank
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Cuando Pase El Temblor - Soda Stereo ...	0.0194504536115	1
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Fireflies - Charttraxx Karaoke ...	0.0144737317012	2
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Love Is A Losing Game - Amy Winehouse ...	0.0142865960415	3
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Marry Me - Train	0.014133471709	4
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Secrets - OneRepublic	0.013591665488	5
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Sehr kosmisch - Harmonia	0.0133987894425	6
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Te Hacen Falta Vitaminas - Soda Stereo ...	0.0129302831796	7
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	OMG - Usher featuring will.i.am ...	0.0127778282532	8
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	Y solo se me ocurre amarte (Unplugged) - ...	0.0123411279458	9
c66c10a9567f0d82ff31441a9fd5063e5cd9dfe8 ...	No Dejes Que... - Caifanes ...	0.0121042499175	10

[10 rows x 4 columns]

In [15]:
```
personalized_model.recommend(users=[users[1]])
```

Out[15]:

user_id	song	score	rank
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Riot In Cell Block Number Nine - Dr Feelgood ...	0.0374999940395	1
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Sei Lá Mangueira - Elizeth Cardoso ...	0.0331632643938	2
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	The Stallion - Ween	0.0322580635548	3
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Rain - Subhumans	0.0314159244299	4
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	West One (Shine On Me) - The Ruts ...	0.0306771993637	5
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Back Against The Wall - Cage The Elephant ...	0.0301204770803	6
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Life Less Frightening - Rise Against ...	0.0284431129694	7
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	A Beggar On A Beach Of Gold - Mike And The ...	0.0230024904013	8
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Audience Of One - Rise Against ...	0.0193938463926	9
279292bb36dbfc7f505e36ebf038c81eb1d1d63e ...	Blame It On The Boogie - The Jacksons ...	0.0189873427153	10

[10 rows x 4 columns]

We can also apply the model to find similar songs to any song in the dataset

In [16]:
```
personalized_model.get_similar_items(['With Or Without You - U2'])
```

Out[16]:

song	similar	score	rank
With Or Without You - U2	I Still Haven't Found What I'm Looking For ...	0.042857170105	1
With Or Without You - U2	Hold Me_ Thrill Me_ Kiss Me_ Kill Me - U2 ...	0.0337349176407	2
With Or Without You - U2	Window In The Skies - U2	0.0328358411789	3
With Or Without You - U2	Vertigo - U2	0.0300751924515	4
With Or Without You - U2	Sunday Bloody Sunday - U2	0.0271317958832	5
With Or Without You - U2	Bad - U2	0.0251798629761	6
With Or Without You - U2	A Day Without Me - U2	0.0237154364586	7
With Or Without You - U2	Another Time Another Place - U2 ...	0.0203251838684	8
With Or Without You - U2	Walk On - U2	0.0202020406723	9
With Or Without You - U2	Get On Your Boots - U2	0.0196850299835	10

[10 rows x 4 columns]

In [17]:
```
personalized_model.get_similar_items(['Chan Chan (Live) - Buena Vista Social Club'])
```

Out[17]:

song	similar	score	rank
Chan Chan (Live) - Buena Vista Social Club ...	Murmullo - Buena Vista Social Club ...	0.188118815422	1
Chan Chan (Live) - Buena Vista Social Club ...	La Bayamesa - Buena Vista Social Club ...	0.18719214201	2
Chan Chan (Live) - Buena Vista Social Club ...	Amor de Loca Juventud - Buena Vista Social Club ...	0.184834122658	3
Chan Chan (Live) - Buena Vista Social Club ...	Diferente - Gotan Project	0.0214592218399	4
Chan Chan (Live) - Buena Vista Social Club ...	Mistica - Orishas	0.0205761194229	5
Chan Chan (Live) - Buena Vista Social Club ...	Hotel California - Gipsy Kings ...	0.0193049907684	6
Chan Chan (Live) - Buena Vista Social Club ...	Nací Orishas - Orishas	0.0191571116447	7
Chan Chan (Live) - Buena Vista Social Club ...	Le Moulin - Yann Tiersen	0.018796980381	8
Chan Chan (Live) - Buena Vista Social Club ...	Gitana - Willie Colon	0.018796980381	9
Chan Chan (Live) - Buena Vista Social Club ...	Criminal - Gotan Project	0.0187793374062	10

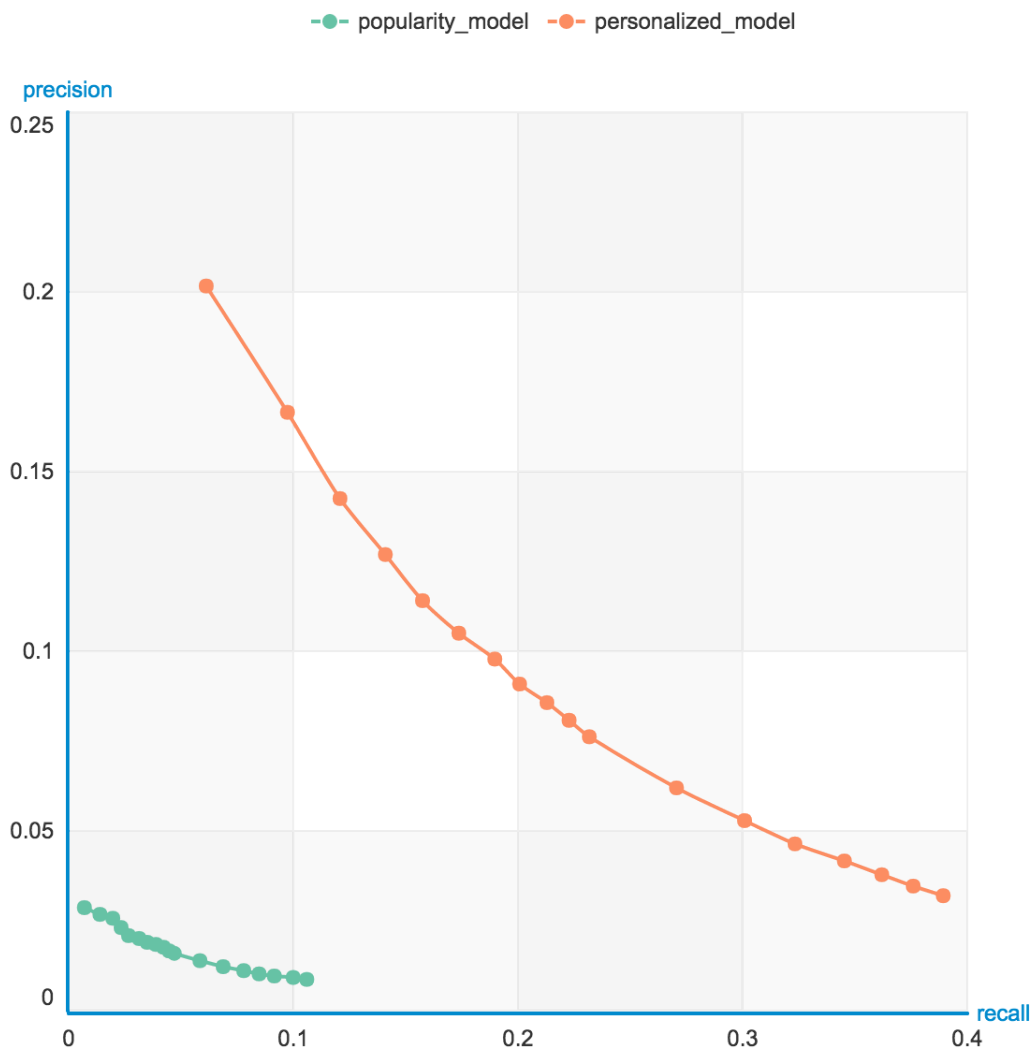
[10 rows x 4 columns]

Quantitative comparison between the models

We now formally compare the popularity and the personalized models using precision-recall curves.

In [18]:
```
if graphlab.version[:3] >= "1.6":
    model_performance = graphlab.compare(test_data, [popularity_model, personal
ized_model], user_sample=0.05)
    graphlab.show_comparison(model_performance,[popularity_model, personalized_
model])
else:
    %matplotlib inline
    model_performance = graphlab.recommender.util.compare_models(test_data, [po
pularity_model, personalized_model], user_sample=.05)
```

```
compare_models: using 2931 users to estimate model performance
PROGRESS: Evaluate model M0

recommendations finished on 1000/2931 queries. users per second: 11206.4

recommendations finished on 2000/2931 queries. users per second: 11009.5

Precision and recall summary statistics by cutoff
+--------+-----------------+------------------+
| cutoff |  mean_precision |   mean_recall    |
+--------+-----------------+------------------+
|   1    |  0.028659160696 | 0.00698147335814 |
|   2    | 0.0267826680314 | 0.0139158104869  |
|   3    | 0.0257022631639 | 0.0196238056269  |
|   4    | 0.0231149778233 |  0.02333294951   |
|   5    | 0.0208802456499 | 0.0266386093152  |
|   6    |  0.02007278517  | 0.0313313340437  |
|   7    | 0.0190086269923 | 0.0349270133559  |
|   8    | 0.0184237461617 | 0.0388254188664  |
|   9    | 0.0176276583646 | 0.0422212392786  |
|   10   |  0.016615489594 | 0.0447152415423  |
+--------+-----------------+------------------+
[10 rows x 3 columns]

PROGRESS: Evaluate model M1

recommendations finished on 1000/2931 queries. users per second: 9247.02

recommendations finished on 2000/2931 queries. users per second: 5820.57

Precision and recall summary statistics by cutoff
+--------+-----------------+-----------------+
| cutoff |  mean_precision |   mean_recall   |
+--------+-----------------+-----------------+
|   1    |  0.201637666325 | 0.0612209515598 |
|   2    |  0.166496076424 | 0.0973709396929 |
|   3    |  0.142499715683 |  0.120768195369 |
|   4    |  0.126919140225 |  0.140957182854 |
|   5    |  0.114090754009 |  0.157508088208 |
|   6    |  0.105026725805 |  0.173665124811 |
|   7    | 0.0978213189063 |  0.189666955959 |
|   8    | 0.0908819515524 |  0.200706035345 |
|   9    |  0.085712119489 |  0.212878540519 |
|   10   |  0.080791538724 |  0.222739007302 |
+--------+-----------------+-----------------+
[10 rows x 3 columns]

Model compare metric: precision_recall
```



The curve shows that the personalized model provides much better performance.

In [ ]:

In [ ]: