
Indian Institute of Technology
Indore

# Project Midsem Evaluation: Computational Intelligence(CS 354N)

## Title: Music Recommendation System

**Faculty: Dr. Aruna Tiwari**

Project Members:

| | |
|---|---|
| **Vikash Kumar** | **170001054** |
| **Vikram Kushwaha** | **170001055** |

# Problem Definition:

As online music streaming becomes the dominant medium for people to listen to their favorite songs, music streaming services are now able to collect large amounts of data on the listening habits of their customers. These streaming services, like Spotify, Apple Music or Pandora, are using this data to provide recommendations to their listeners. These music recommendation systems are part of a broader class of recommender systems, which filter information to predict a user's preferences when it comes to a certain item.

Building a music recommendation system is a common task that is faced by Spotify, iTunes, JioSaavn, Pandora, etc . The underlying goal of the music recommendation system is to personalize content and identify relevant data for our audiences. The parameters to identify contents can be genre, artist, album, label, etc.

The idea is to build a music recommendation system using real data. There are 3 types of recommendation system: content-based, collaborative and popularity-based.

# Analysis and Design of Problem:

## I. Data Collection and Preprocessing:

Real world data will be used to train the model. The dataset contains millions of training examples to train on. The dataset contains two files: triplet_file and metadat_file. The triplet_file contains user_id, song_id and listening_time. The metadat_file contains song_id, title, release_by and artist_name. Million Songs Dataset is a mixture of songs from various websites such as lastfm.fm, musixmatch, etc with the rating that users gave after listening to the song.

Data collected from:-

//done on jupyter notebook

```
triplets_file =
'https://static.turi.com/datasets/millionsong/10000.txt'
songs_metadata_file =
'https://static.turi.com/datasets/millionsong/song_data.csv'
```

| | song_id | title | release | artist_name | year |
|---|---|---|---|---|---|
| 2 | SOQMMHC12AB0180CB8 | Silent Night | Monster Ballads X-Mas | Faster Pussy cat | 2003 |
| 3 | SOVFVAK12A8C1350D9 | Tanssi vaan | KarkuteillÃ¤ | Karkkiautomaatti | 1995 |
| 4 | SOGTUKN12AB017F4F1 | No One Could Ever | Butter | Hudson Mohawke | 2006 |
| 5 | SOBNYVR12A8C13558C | Si Vos QuerÃ©s | De Culo | Yerba Brava | 2003 |
| 6 | SOHSBXH12A8C13B0DF | Tangle Of Aspens | Rene Ablaze Presents Winter Sessions | Der Mystic | 0 |
| 7 | SOZVAPQ12A8C13B63C | Symphony No. 1 G minor "Sinfonie Serieuse" | Berwald: Symphonies Nos. 1/2/3/4 | David Montgomery | 0 |
| 8 | SOQVRHI12A6D4FB2D7 | We Have Got Love | Strictly The Best Vol. 34 | Sasha / Turbulence | 0 |
| 9 | SOEYRFT12AB018936C | 2 Da Beat Ch'yall | Da Bomb | Kris Kross | 1993 |
| 10 | SOPMIYT12A6D4F851E | Goodbye | Danny Boy | Joseph Locke | 0 |
| 11 | SOJCFMH12A8C13B0C2 | Mama_ mama can't you see ? | March to cadence with the US marines | The Sun Harbor's Chorus | 0 |
| 12 | SOYGNWH12AB018191E | L'antarctique | Des cobras des tarentules | 3 Gars Su'l Sofa | 2007 |

metadat_file

```
 1  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOAKIMP12A8C130995  1
 2  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOBBMDR12A8C13253B  2
 3  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOBXHDL12A81C204C0  1
 4  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOBYHAJ12A6701BF1D  1
 5  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SODACBL12A8C13C273  1
 6  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SODDNQT12A6D4F5F7E  5
 7  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SODXRTY12AB0180F3B  1
 8  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOFGUAY12AB017B0A8  1
 9  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOFRQTD12A81C233C0  1
10  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOHQWYZ12A6D4FA701  1
11  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOIYTOA12A6D4F9A23  1
12  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOIZAZL12A6701C53B  5
13  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOJNNUA12A8AE48C7A  1
14  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOJPFQG12A58A7833A  1
15  b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOKRIMP12A6D4F5DA3  5
```

triplet_file

Our first job is to integrate our dataset, which is very important every time we want to build a data processing pipeline.To integrate both triplet_file and metadata_file, we are going to use a popular Python library called pandas.

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

Out[3]:

| | user_id | song_id | listen_count | title | release | artist_name | year |
|---|---|---|---|---|---|---|---|
| 0 | b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOAKIMP12A8C130995 | 1 | The Cove | Thicker Than Water | Jack Johnson | 0 |
| 1 | b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOBBMDR12A8C13253B | 2 | Entre Dos Aguas | Flamenco Para Niños | Paco De Lucia | 1976 |
| 2 | b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOBXHDL12A81C204C0 | 1 | Stronger | Graduation | Kanye West | 2007 |
| 3 | b80344d063b5ccb3212f76538f3d9e43d87dca9e | SOBYHAJ12A6701BF1D | 1 | Constellations | In Between Dreams | Jack Johnson | 2005 |
| 4 | b80344d063b5ccb3212f76538f3d9e43d87dca9e | SODACBL12A8C13C273 | 1 | Learn To Fly | There Is Nothing Left To Lose | Foo Fighters | 1999 |

The second step  is data transformation, where we select a subset of this data (the first 10,000 songs). We then merge the song and artist_name into one column, aggregated by the number of times a particular song is listened to in general by all users.

The percentage is calculated by dividing the listen_count by the sum of listen_count of all songs and then multiply by 100.

In [6]:
```
song_grouped = song_df.groupby(['song']).agg({'listen_count': 'count'}).reset_index()
grouped_sum = song_grouped['listen_count'].sum()
song_grouped['percentage']  = song_grouped['listen_count'].div(grouped_sum)*100
song_grouped.sort_values(['listen_count', 'song'], ascending = [0,1])
```

Out[6]:

| | song | listen_count | percentage |
|---|---|---|---|
| 3660 | Sehr kosmisch - Harmonia | 45 | 0.45 |
| 4678 | Undo - Björk | 32 | 0.32 |
| 5105 | You're The One - Dwight Yoakam | 32 | 0.32 |
| 1071 | Dog Days Are Over (Radio Edit) - Florence + Th... | 28 | 0.28 |
| 3655 | Secrets - OneRepublic | 28 | 0.28 |
| ... | ... | ... | ... |
| 5139 | high fives - Four Tet | 1 | 0.01 |
| 5140 | in white rooms - Booka Shade | 1 | 0.01 |
| 5143 | paranoid android - Christopher O'Riley | 1 | 0.01 |
| 5149 | ¿Lo Ves? [Piano Y Voz] - Alejandro Sanz | 1 | 0.01 |
| 5150 | Época - Gotan Project | 1 | 0.01 |

5151 rows × 3 columns

# II. Study & Understanding of Algorithm

There are 3 types of recommendation system: content-based, collaborative and popularity-based.

*Popularity based recommender*: Recommends songs to the user based on the popularity of the songs in our dataset irrespective of the user's listening activity.

*Collaborative based recommender:* The collaborative filtering approach to recommendation algorithms involves collecting a "large amount of information on users' behaviors, activities or preferences and predicting what users will like based on their similarity to other users". One of the major issues with the collaborative filtering approach is the

so-called “cold start problem”, in that the system needs a large amount of data to make accurate recommendations.

*Content based recommender:* The content-based filtering approach differs from the collaborative filtering approach as it filters based on an analysis of both the item being recommended and the user. Content-based filtering closely examines the actual item to determine which features are most important in making recommendations and how those features interact with the user’s preferences.

**In short, a Content based system predicts what a user likes based on what that user likes in the past. Collaborative based systems predict what a particular user likes based on what other similar users like.**

### Division of Training and Test Dataset :

```
In [11]: train_data, test_data = train_test_split(song_df, test_size = 0.20, random_state=0)
         print(train_data.head(5))
```

```
                                      user_id             song_id  \
7389  94d5bdc37683950e90c56c9b32721edb5d347600  SOXNZOW12AB017F756
9275  1012ecfd277b96487ed8357d02fa8326b13696a5  SOXHYVQ12AB0187949
2995  15415fa2745b344bce958967c346f2a89f792f63  SOOSZAZ12A6D4FADF8
5316  ffadf9297a99945c0513cd87939d91d8b602936b  SOWDJEJ12A8C1339FE
356   5a905f000fc1ff3df7ca807d57edb608863db05d  SOAMPRJ12A8AE45F38

      listen_count                title  \
7389             2     Half Of My Heart
9275             1  The Beautiful People
2995             1    Sanctify Yourself
5316             4    Heart Cooks Brain
356             20                Rorol

                                              release      artist_name  \
7389                                  Battle Studies       John Mayer
9275            Antichrist Superstar (Ecopac Explicit)   Marilyn Manson
2995                           Glittering Prize 81/92     Simple Minds
5316  Everything Is Nice: The Matador Records 10th A...     Modest Mouse
356                             Identification Parade  Octopus Project

      year                                  song
7389     0          Half Of My Heart - John Mayer
9275     0  The Beautiful People - Marilyn Manson
2995  1985       Sanctify Yourself - Simple Minds
5316  1997       Heart Cooks Brain - Modest Mouse
356   2002                Rorol - Octopus Project
```

## II.A) Creating a popularity Based Recommender

The popularity based system recommends the most popular songs played by all the users. The recommended songs are independent of the user during testing because most popular songs played are the same for all.

```
In [12]: pm = Recommenders.popularity_recommender_py()
         pm.create(train_data, 'user_id', 'song')
```

Recommendations for user number - 6

```
In [13]: user_id = users[5]
         pm.recommend(user_id)
```

Out[13]:

| | user_id | song | score | Rank |
|---|---|---|---|---|
| **3194** | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Sehr kosmisch - Harmonia | 37 | 1 |
| **4083** | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Undo - Björk | 27 | 2 |
| **931** | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Dog Days Are Over (Radio Edit) - Florence + Th... | 24 | 3 |
| **4443** | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | You're The One - Dwight Yoakam | 24 | 4 |
| **3034** | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Revelry - Kings Of Leon | 21 | 5 |
| **3189** | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Secrets - OneRepublic | 21 | 6 |
| **4112** | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Use Somebody - Kings Of Leon | 21 | 7 |
| **1207** | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Fireflies - Charttraxx Karaoke | 20 | 8 |
| **1577** | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Hey_ Soul Sister - Train | 19 | 9 |
| **1626** | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Horn Concerto No. 4 in E flat K495: II. Romanc... | 19 | 10 |

Recommendation for user number - 9

```
In [14]: ###Fill in the code here
         user_id = users[8]
         pm.recommend(user_id)
```

Out[14]:

| | user_id | song | score | Rank |
|---|---|---|---|---|
| **3194** | 9bb911319fbc04f01755814cb5edb21df3d1a336 | Sehr kosmisch - Harmonia | 37 | 1 |
| **4083** | 9bb911319fbc04f01755814cb5edb21df3d1a336 | Undo - Björk | 27 | 2 |
| **931** | 9bb911319fbc04f01755814cb5edb21df3d1a336 | Dog Days Are Over (Radio Edit) - Florence + Th... | 24 | 3 |
| **4443** | 9bb911319fbc04f01755814cb5edb21df3d1a336 | You're The One - Dwight Yoakam | 24 | 4 |
| **3034** | 9bb911319fbc04f01755814cb5edb21df3d1a336 | Revelry - Kings Of Leon | 21 | 5 |
| **3189** | 9bb911319fbc04f01755814cb5edb21df3d1a336 | Secrets - OneRepublic | 21 | 6 |
| **4112** | 9bb911319fbc04f01755814cb5edb21df3d1a336 | Use Somebody - Kings Of Leon | 21 | 7 |
| **1207** | 9bb911319fbc04f01755814cb5edb21df3d1a336 | Fireflies - Charttraxx Karaoke | 20 | 8 |
| **1577** | 9bb911319fbc04f01755814cb5edb21df3d1a336 | Hey_ Soul Sister - Train | 19 | 9 |
| **1626** | 9bb911319fbc04f01755814cb5edb21df3d1a336 | Horn Concerto No. 4 in E flat K495: II. Romanc... | 19 | 10 |

➢ **We can see that recommendations are the same for both the users since it's a popularity based recommender system.**

## II.B) Creating a Similarity Based Personalized Recommender

This system recommends user songs based on what they are listening to and what are other similar songs to a given song.

```
In [15]: is_model = Recommenders.item_similarity_recommender_py()
         is_model.create(train_data, 'user_id', 'song')
```

## Personalized songs based on listening choice of user - 6

In [16]:
```
#Print the songs for the user in training data
user_id = users[5]
user_items = is_model.get_user_items(user_id)
#
print("------------------------------------------------------------------------------------")
print("Training data songs for the user userid: %s:" % user_id)
print("------------------------------------------------------------------------------------")

for user_item in user_items:
    print(user_item)

print("----------------------------------------------------------------------")
print("Recommendation process going on:")
print("----------------------------------------------------------------------")

#Recommend songs for the user using personalized model
is_model.recommend(user_id)
```

```
------------------------------------------------------------------------------------
Training data songs for the user userid: 4bd88bfb25263a75bbdd467e74018f4ae570e5df:
------------------------------------------------------------------------------------
Just Lose It - Eminem
Without Me - Eminem
16 Candles - The Crests
Speechless - Lady GaGa
Push It - Salt-N-Pepa
Ghosts 'n' Stuff (Original Instrumental Mix) - Deadmau5
Say My Name - Destiny's Child
My Dad's Gone Crazy - Eminem / Hailie Jade
The Real Slim Shady - Eminem
Somebody To Love - Justin Bieber
Forgive Me - Leona Lewis
Missing You - John Waite
Ya Nada Queda - Kudai
----------------------------------------------------------------------
Recommendation process going on:
----------------------------------------------------------------------
No. of unique songs for the user: 13
no. of unique songs in the training set: 4483
Non zero values in cooccurence_matrix :2097
```

Out[16]:

| | user_id | song | score | rank |
|---|---|---|---|---|
| 0 | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Superman - Eminem / Dina Rae | 0.088692 | 1 |
| 1 | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Mockingbird - Eminem | 0.067663 | 2 |
| 2 | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | I'm Back - Eminem | 0.065385 | 3 |
| 3 | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | U Smile - Justin Bieber | 0.064525 | 4 |
| 4 | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Here Without You - 3 Doors Down | 0.062293 | 5 |
| 5 | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Hellbound - J-Black & Masta Ace | 0.055769 | 6 |
| 6 | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | The Seed (2.0) - The Roots / Cody Chestnutt | 0.052564 | 7 |
| 7 | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | I'm The One Who Understands (Edit Version) - War | 0.052564 | 8 |
| 8 | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Falling - Iration | 0.052564 | 9 |
| 9 | 4bd88bfb25263a75bbdd467e74018f4ae570e5df | Armed And Ready (2009 Digital Remaster) - The ... | 0.052564 | 10 |

## Personalized songs based on listening choice of user - 8

In [17]:
```
user_id = users[7]
#Fill in the code here
user_items = is_model.get_user_items(user_id)
#
print("----------------------------------------------------------------------------------------")
print("Training data songs for the user userid: %s:" % user_id)
print("----------------------------------------------------------------------------------------")

for user_item in user_items:
    print(user_item)

print("----------------------------------------------------------------------")
print("Recommendation process going on:")
print("----------------------------------------------------------------------")

#Recommend songs for the user using personalized model
is_model.recommend(user_id)
```

```
----------------------------------------------------------------------------------------
Training data songs for the user userid: 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec:
----------------------------------------------------------------------------------------
Swallowed In The Sea - Coldplay
Life In Technicolor ii - Coldplay
Life In Technicolor - Coldplay
The Scientist - Coldplay
Trouble - Coldplay
Strawberry Swing - Coldplay
Lost! - Coldplay
Clocks - Coldplay
----------------------------------------------------------------------
Recommendation process going on:
----------------------------------------------------------------------
No. of unique songs for the user: 8
no. of unique songs in the training set: 4483
Non zero values in cooccurence_matrix :3429
```

Out[17]:

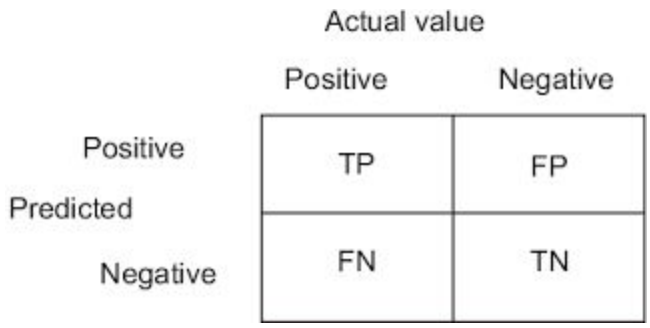
| | user_id | song | score | rank |
|---|---|---|---|---|
| 0 | 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec | She Just Likes To Fight - Four Tet | 0.281579 | 1 |
| 1 | 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec | Warning Sign - Coldplay | 0.281579 | 2 |
| 2 | 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec | We Never Change - Coldplay | 0.281579 | 3 |
| 3 | 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec | Puppetmad - Puppetmastaz | 0.281579 | 4 |
| 4 | 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec | God Put A Smile Upon Your Face - Coldplay | 0.281579 | 5 |
| 5 | 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec | Susie Q - Creedence Clearwater Revival | 0.281579 | 6 |
| 6 | 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec | The Joker - Fatboy Slim | 0.281579 | 7 |
| 7 | 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec | Korg Rhythm Afro - Holy Fuck | 0.281579 | 8 |
| 8 | 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec | This Unfolds - Four Tet | 0.281579 | 9 |
| 9 | 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec | high fives - Four Tet | 0.281579 | 10 |

➢ **We can see that recommendations are different for both the users since it's a similarity based personalized recommender system.**

# III. Study of Performance Measurement criteria

We will measure the performance of this model using *a precision recall curve* to quantitatively compare the popularity based model and personalized collaborative filtering model.

To quantitatively measure the performance of the recommender system, we use three different metrics: Precision , Recall and F-1 Score.




TP = True Positive FP = False Positive
FN = False Negative TN = True Negative

Precision = $(True\ Positive)/(True\ Positive\ +\ False\ Positive)$
Recall = $(True\ Positive)/(True\ Positive\ +True\ Negative)$
F1 score = $2*(Precision\ *Recall)/(Precision\ +Recall)$

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

*According to Wikipedia*: In measurement of a set, **accuracy** refers to closeness of the measurements to a specific value., while **precision** refers to the closeness of the measurements to each other.

In pattern recognition, information retrieval and classification (machine learning), **precision** (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while **recall** (also known as sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved.

In statistical analysis of binary classification, the **$F_1$ score** (also **F-score** or **F-measure**) is a measure of a test's accuracy. It considers both the precision *p* and the recall *r* of the test to compute the score.

In our case, the definition relevant for our problem domain is the length that a song is listened to, a number of users have all liked the song. Recall would “measure the proportion of all relevant results included in the top results”.In our case, it means *precision* seeks to measure the relevancy of songs in relation to the top ten results of recommended song, whereas *recall* seeks to measure the relevancy of songs in relation to all the songs.

## Using the precision recall calculator class to calculate the evaluation measures

In [20]:
```
start = time.time()

#Define what percentage of users to use for precision recall calculation
user_sample = 0.05

#Instantiate the precision_recall_calculator class
pr = Evaluation.precision_recall_calculator(test_data, train_data, pm, is_model)

#Call method to calculate precision and recall values
(pm_avg_precision_list, pm_avg_recall_list, ism_avg_precision_list, ism_avg_recall_list) = pr.calculate_measures(user_sample)

end = time.time()
print(end - start)
```

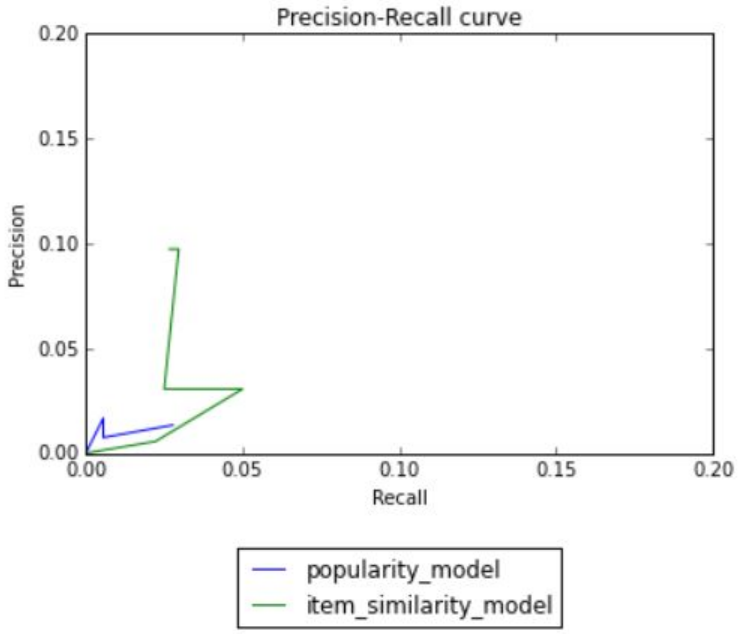
```
Length of user_test_and_training:319
Length of user sample:15
Getting recommendations for user:ea3b77e3f9b5688dc3998b2e706ea2c0ca48b8eb
No. of unique songs for the user: 15
no. of unique songs in the training set: 4483
Non zero values in cooccurence_matrix :1742
```

## Code to plot precision recall curve

In [21]:
```
import pylab as pl

#Method to generate precision and recall curve
def plot_precision_recall(m1_precision_list, m1_recall_list, m1_label, m2_precision_list, m2_recall_list, m2_label):
    pl.clf()
    pl.plot(m1_recall_list, m1_precision_list, label=m1_label)
    pl.plot(m2_recall_list, m2_precision_list, label=m2_label)
    pl.xlabel('Recall')
    pl.ylabel('Precision')
    pl.ylim([0.0, 0.20])
    pl.xlim([0.0, 0.20])
    pl.title('Precision-Recall curve')
    #pl.legend(loc="upper right")
    pl.legend(loc=9, bbox_to_anchor=(0.5, -0.2))
    pl.show()
```

Plotting precision recall curves.




# Conclusion

Observing the precision recall curve of both our popularity based model and personalized item similarity model, item similarity model perform better (i.e. having higher number of recall and precision) up to certain point in precision-recall curve.The curve shows that the personalized model provides better performance over the popularity model.

# Thank You