Music Recommender System

Prakyath Madadi
pm3140

What’s a recommender system?

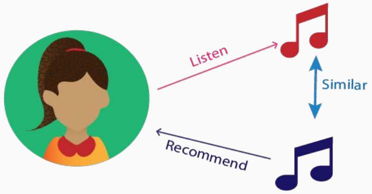
- A system which recommends content to users.
  - Spotify, Netflix, Amazon ...
- Why?
  - Better User experience
    - More profits
  - Proxy for recommendations from social interactions.

How?

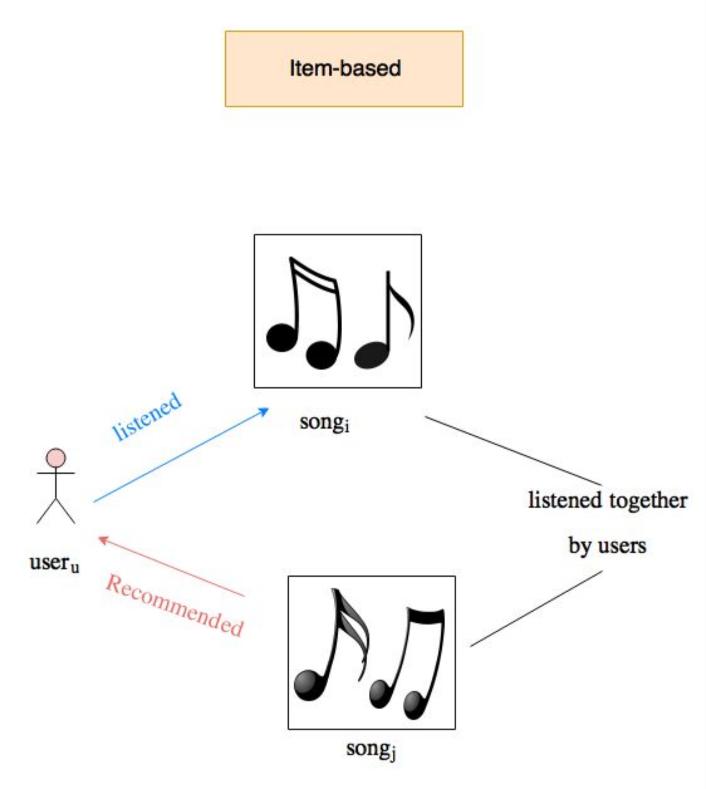
- Based on user content interactions.
- Sources:
  - Implicit - plays, watch/listen history
  - Explicit - ratings

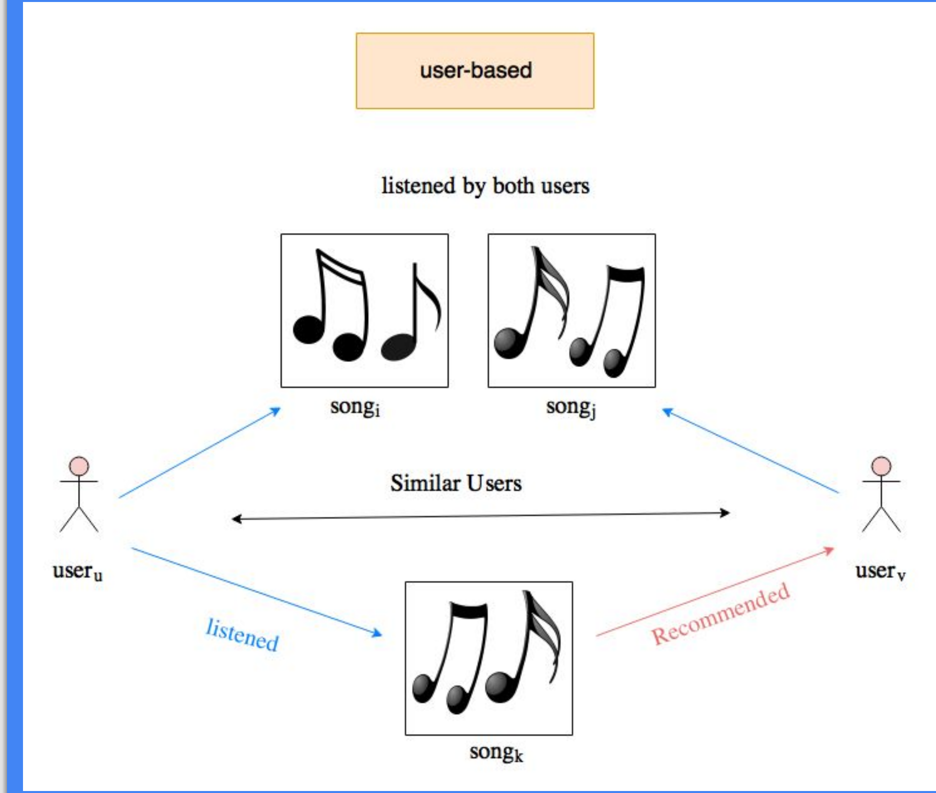
- Build a system that predicts content that a user will most likely engage with.

Types



- Recommender Systems
  - Content Based Filtering
    - Recommend content similar to the ones user interacted with, based on content similarity.
  - Collaborative Filtering
    - Memory Based Filtering
      - User Based Filtering
        - Recommend similar content to similar users.
      - Item Based filtering
        - Recommend content similar to the ones user interacted with, by memorizing user content interactions.
    - Model Based Filtering
      - Matrix Factorization, Clustering, Deep Learning
        - Recommend content based on user, content features learned from user interactions.
  - Hybrid Filtering
    - Recommend content based on a combined (content, collaborative) approach


Item-based
listened
song_i
listened together
by users
user_u
Recommended
song_j


user-based
listened by both users
song_i
song_j
Similar Users
user_u
user_v
listened
Recommended
song_k

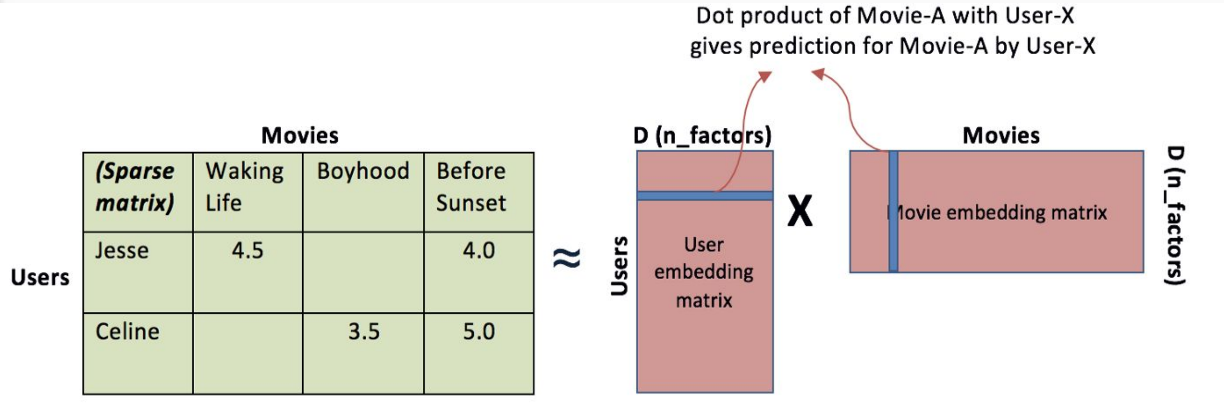
Matrix Factorization

→ I've used the matrix factorization method of Collaborative filtering.

→ Specifically, I've used the Alternating Least squares (ALS) method.

→ The user interaction matrix (The user-artist ratings) is decomposed into user and content embeddings.

→ The embeddings are used to get the predictions.



Movies	User 1	User 2	User 3	User 4
Item 1	1		4	5
Item 2	5	4	1	2
Item 3	4	4		3
Item 4	2	2	4	
Item 5		5	3	2

Movies				User 1	User 2	User 3	User 4
				x6	x7	x8	x9
				y6	y7	y8	y9
Item 1	x1	y1		x1.x6+y1.y6	x1.x7+y1.y7	x1.x8+y1.y8	x1.x9+y1.y9
Item 2	x2	y2		x2.x6+y2.y6	x2.x7+y2.y7	x2.x8+y2.y9	x2.x9+y2.y9
Item 3	x3	y3		x3.x6+y3.y6			
Item 4	x4	y4		x4.x6+y4.y6			
Item 5	x5	y5		x5.x6+y5.y6			

The Algorithm

- In ALS, the algorithm tries to minimize 2 loss functions based on the user features (V) and content features (U).
- A is the rating matrix.
- The loss for minimizing with respect to content features:

$$\Sigma^n{}_j \Sigma_i{}^m (A_{ij} - U_i V_j{}^T)^2 \quad \text{for } r(i,j)=1 \text{ or } (i,j) \in (\text{rated}) + \lambda/2\ \Sigma_i{}^m (U_i)^2$$

- The loss for minimizing with respect to user features:

$$\Sigma_i{}^m \Sigma^n{}_j (A_{ij} - U_i V_j{}^T)^2 \quad \text{for } r(i,j)=1 \text{ or } (i,j) \in (\text{rated}) + \lambda/2\ \Sigma^n{}_j\ (V_j)^2$$

- The algorithm alternates between the two equations, solving for ‘U’ and ‘V’.

Advantages over other methods

- The user interaction matrix is **sparse**, making the models difficult to train.
- **Cold start problem:** New Users and Content will have very few interactions making it difficult for the predictions
- **Scalability:** When the number of users and Content scale, they might not fit in a single machine.
- ALS can deal with sparsity and cold start problem through feature projection. It also can be performed in a distributed way.

Disadvantages

- The ALS also can’t predict the ratings for new items or features.
- Can be solved using projections.
- For every new item solve the following equation: (Similarly for new users)

$$\min_{v_{i_0} \in \mathbb{R}^d} ||A_{i_0} - U v_{i_0}||$$

- It gets difficult to include side features like explicit ratings for songs/movies.

Data

- I've used a subset of the Last.Fm Dataset.
- The data consists of users, artists and ratings for the artists.
- The data is imported from json files.
- There are about 30,000 ratings for 10,000 artists.

Data

Users

_id	first_name	last_name
srunolfsdottir	Seamus	Runolfsdottir
cbartoletti	Celestine	Bartoletti
kschamberger	Kattie	Schamberger
sschamberger	Sidney	Schamberger
vschamberger	Victor	Schamberger
dmacejkovic	Delfina	Macejkovic
rchristiansen	Roy	Christiansen
eankunding	Everette	Ankunding
erutherford	Emilio	Rutherford
evandervort	Elvera	Vandervort
gankunding	Gregoria	Ankunding
hbartoletti	Hector	Bartoletti
hcartwright	Hailee	Cartwright
hgusikowski	Howard	Gusikowski
karmstrong	Kenyatta	Armstrong
drosenbaum	Dorothy	Rosenbaum
ghalvorson	Gaylord	Halvorson
kswaniawski	Kyler	Swaniawski
arosenbaum	Amanda	Rosenbaum
aschimmel	Adelbert	Schimmel

Artists

_id	name
00010eb3-ebfe-4965-81ef-0ac64cd49fde	La Niña de los Peines
00034ede-a1f1-4219-be39-02f36853373e	O Rappa
0003fd17-b083-41fe-83a9-d550bd4f00a1	安倍なつみ
0004537a-4b12-43eb-a023-04009e738d2e	Ultra Naté
0013bcdd-fe35-4c9f-ac41-4b41b9000e17	Siobhan Magnus
00302a51-04f5-4b2d-8993-fcd1c2ae1cf4	柚楽弥衣
0035a150-bceb-4abe-88ca-a494cdf14968	Laura McCormick
0035c656-9853-44a8-b105-c44089a43cea	Diandra
00376321-ce0f-4bd7-a98f-fcabdbf06ea7	Raappana
0039c7ae-e1a7-4a7d-9b49-0cbc716821a6	Death Cab for Cutie
003b2747-b74a-46c1-a51e-aeaffe88256c	Erdmöbel
0040c89b-f2e6-4bc3-b75d-5152fb0c890e	Bernhard Romberg
00413ec4-1dd2-4878-b98a-743cf6499501	Rauno Lehtinen
0042aa4d-a972-4a2a-b7cf-8044d09bbc67	Andrew Fletcher
00445d42-1e63-48e7-ba2f-066cd37a0927	Hilding Rosenberg
004e5eed-e267-46ea-b504-54526f1f377d	The Gathering
004e6286-a1c3-4174-b9f5-710666e39ecf	Richie Loop
005a1712-6d74-47cb-9032-8bd63febd966	Nichole Cordova
005dd8c0-1c1c-4ed2-8963-4249449f5901	Ernst Jandl
006631f4-5214-49ff-a386-a064853d1a1e	Marco Antonio Solís

only showing top 20 rows

Data

artist_name	artist_idx	rating	first_name	last_name	user_idx
Johann Ludwig Dei...	4103	1.5	Quinton	Kuphal	147
Wojciech Kilar	8156	1.0	Grace	Keebler	135
Me G	1061	3.0	Annabelle	Kovacek	31
Johann Ludwig Dei...	4103	4.5	Genevieve	Koepp	136
Marianne Hoppe	486	3.5	Paolo	Roob	295
Ken Mary	7399	4.0	Liza	Hayes	284
Christopher Smith	4577	4.0	Anabel	Raynor	156
Jānis Volkinšteins	3181	4.0	Maia	Turner	231
Jacob Appelbaum	9710	1.5	Alexis	Rice	273
Demetrio Stratos	2591	1.0	Hank	Greenholt	56
Laura McCormick	7	0.5	Burley	Cormier	97
Robert Pete Williams	7170	1.5	Alaina	Hoppe	212
Ilmari Kianto	7815	3.5	Patrick	Sipes	208
Andreas Melzer	1381	5.0	Maximilian	Beer	177
Terry Zwigoff	5393	3.0	Dorcas	Nitzsche	50
Sally Liebling	6705	1.0	Grant	Denesik	134
MINX	1417	4.5	Roslyn	Frami	235
Naked Lunch	5996	4.5	Mellie	Farrell	117
Ludwig van Beethoven	1257	0.5	Vallie	Bins	288
R5	7357	2.5	Elvie	Streich	130

Implementation

- I’ve used python and pyspark to implement the algorithm.
- Mean square error is the metric used to check the accuracy of the algorithm.
- I’ve also tuned the hyperparameters such as the ‘rank’, which is the number of features in the user/artists matrices, and regularization parameters using cross validation.

Results

user_idx	recommendations
1	[{7544, 3.4299242}, {6788, 3.4201376}, {1819, 3.391148}, {8089, 3.383186}, {3136, 3.3413465}]
2	[{6788, 3.3839545}, {8089, 3.3699589}, {4243, 3.359496}, {8918, 3.3417923}, {7544, 3.333958}]
3	[{6788, 3.6798096}, {7544, 3.523285}, {8089, 3.4815512}, {1819, 3.4570453}, {7303, 3.3736715}]
4	[{7544, 3.4695644}, {6788, 3.434499}, {1819, 3.4197497}, {8089, 3.3995671}, {3136, 3.3704789}]
5	[{6788, 3.6248646}, {8089, 3.4687984}, {7544, 3.4566078}, {1819, 3.4042354}, {7694, 3.3865192}]
6	[{7544, 3.5000014}, {6788, 3.476198}, {1819, 3.4270995}, {8089, 3.4048772}, {3136, 3.3552952}]
7	[{3136, 3.3202245}, {4243, 3.3086903}, {7544, 3.2900515}, {8918, 3.2878594}, {1819, 3.2855673}]
8	[{6788, 3.4894404}, {7694, 3.369728}, {2341, 3.344026}, {8089, 3.312599}, {8918, 3.2796593}]
9	[{6788, 3.6929147}, {8089, 3.5765963}, {7544, 3.5674694}, {1819, 3.5080776}, {2341, 3.505864}]
10	[{6788, 3.2889228}, {8089, 3.2568095}, {7544, 3.2424614}, {4243, 3.235199}, {1819, 3.228875}]
11	[{6788, 3.487557}, {8089, 3.2995489}, {7544, 3.2794623}, {1819, 3.2380385}, {7694, 3.233576}]
12	[{7544, 3.3825257}, {8089, 3.3678699}, {1819, 3.3520553}, {6788, 3.3459547}, {4243, 3.3431408}]
13	[{7544, 3.2962694}, {6788, 3.2626708}, {8089, 3.2562191}, {1819, 3.2518914}, {3136, 3.2206428}]
14	[{8918, 3.3932278}, {4243, 3.3337202}, {6788, 3.2984304}, {8089, 3.2972484}, {6316, 3.2892783}]
15	[{7544, 3.5592954}, {6788, 3.5539331}, {8089, 3.5320318}, {1819, 3.5154307}, {4243, 3.478233}]
16	[{7544, 3.4293418}, {3136, 3.4085493}, {1819, 3.4084687}, {8089, 3.3893166}, {4243, 3.3833141}]
17	[{8918, 3.4218934}, {4243, 3.3960438}, {8089, 3.3795955}, {6788, 3.3744016}, {1819, 3.3549037}]
18	[{6788, 3.4827743}, {7544, 3.429376}, {8089, 3.3829737}, {1819, 3.3740323}, {4243, 3.2913897}]
19	[{7544, 3.4727392}, {1819, 3.393508}, {3136, 3.3823984}, {8089, 3.3384373}, {6788, 3.3067024}]
20	[{7544, 3.5019748}, {1819, 3.443106}, {3136, 3.4262924}, {8089, 3.4194005}, {6788, 3.3979461}]

Results

artist_idx	recommendations
1	[{264, 2.8941636}, {64, 2.714923}, {144, 2.6994388}, {167, 2.6796587}, {151, 2.66272}]
2	[{264, 2.9103405}, {189, 2.7131872}, {143, 2.6199155}, {64, 2.5764272}, {144, 2.505271}]
3	[{264, 2.8888388}, {143, 2.5938532}, {189, 2.5587416}, {64, 2.548776}, {144, 2.4307094}]
4	[{271, 2.8172092}, {151, 2.8040452}, {144, 2.8008206}, {8, 2.7958412}, {167, 2.7786841}]
5	[{264, 3.2092507}, {189, 2.9546824}, {144, 2.9401858}, {64, 2.9273157}, {151, 2.8702097}]
6	[{264, 3.593741}, {189, 3.2614257}, {64, 3.2275956}, {143, 3.2243874}, {144, 3.1612148}]
7	[{264, 3.4889274}, {143, 2.98212}, {64, 2.975236}, {189, 2.9597526}, {144, 2.857447}]
8	[{264, 3.439353}, {64, 3.0507367}, {143, 3.0077093}, {144, 2.9767878}, {272, 2.9580722}]
9	[{264, 2.9044318}, {258, 2.8173397}, {167, 2.8134878}, {143, 2.7507231}, {43, 2.7214255}]
10	[{264, 3.3771493}, {189, 3.1540482}, {143, 3.1138847}, {258, 3.0464172}, {64, 3.0440028}]
11	[{264, 2.6846313}, {143, 2.4745443}, {189, 2.461676}, {64, 2.442401}, {258, 2.4124374}]
12	[{264, 3.226561}, {189, 2.994365}, {143, 2.9604566}, {64, 2.9286032}, {258, 2.8598876}]
13	[{264, 2.9316895}, {64, 2.7189112}, {144, 2.710484}, {151, 2.6634367}, {189, 2.6629453}]
14	[{264, 3.365703}, {143, 3.0124838}, {64, 2.9809248}, {189, 2.9762921}, {144, 2.8620853}]
15	[{264, 2.82973}, {64, 2.4995947}, {143, 2.4728336}, {189, 2.4676938}, {144, 2.4461305}]
16	[{264, 3.7510567}, {189, 3.3900373}, {143, 3.322585}, {64, 3.264126}, {144, 3.143033}]
17	[{264, 2.4740593}, {64, 2.2544217}, {143, 2.2348223}, {144, 2.2053916}, {189, 2.1996322}]
18	[{264, 2.8612745}, {64, 2.6036758}, {144, 2.5715928}, {143, 2.5552077}, {189, 2.5330293}]
19	[{264, 4.0491486}, {189, 3.5195308}, {143, 3.442467}, {64, 3.2949424}, {35, 3.1672206}]
20	[{264, 2.9501026}, {189, 2.6810246}, {64, 2.6440535}, {143, 2.6231587}, {144, 2.6115296}]

Future Scope

- The embeddings can be learnt using Neural networks.
- Predict by analyzing the content. For example, giving recommendations based on lyrics.
- Accommodate side features like explicit ratings.

Questions?

Thanks!