Building a Recommender Engine Using Last.fm's Artist Rating Data

Hamid Niki
Springboard
Capstone Project 2

Recommender Systems:

- They have been around since 1992
- Today we see different flavors and implementations of them
- They predict whether or not a user will like an item they have not seen.
- Examples:
  - Amazon
  - Facebook
  - Hulu
  - Spotify

Springboard

Basic Approaches:

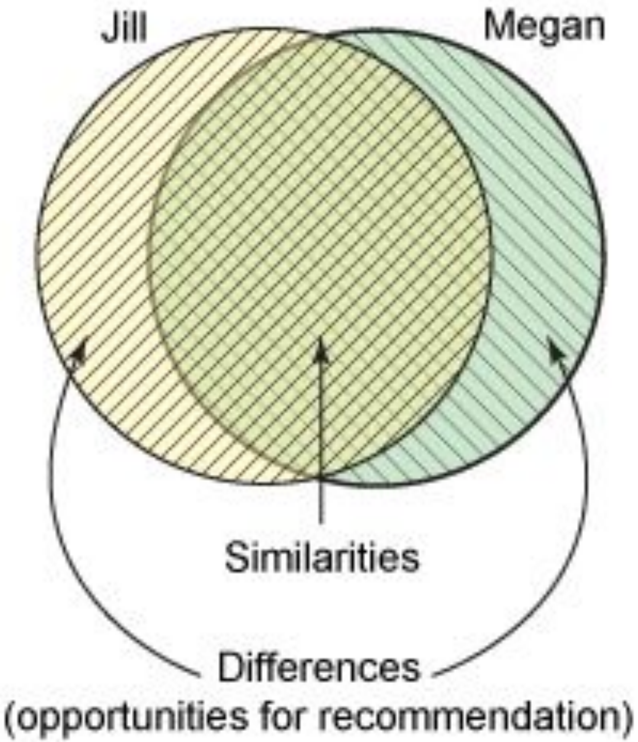
- **Collaborative filtering:** Recommendations are based on a collaboration of multiple users and filtered on those who exhibit similar preferences.

- **Content-Based filtering:** Constructs a recommendation on the basis of a single user's behavior in the past

- **Hybrid:** They combine collaborative and content-based filtering which results in increasing the efficiency, accuracy (and complexity) of recommender systems.

Simple example of collaborative filtering:

Blogs	Marc	Megan	Elise	Jill
Linux	13	3	11	-
OpenSource	10	-	-	3
Cloud Computing	6	1	9	-
Java Technology	-	6	-	9
Agile	-	7	1	8
	Articles read per user			
Cluster	1	2	1	2


Jill
Megan
Similarities
Differences
(opportunities for recommendation)

Simple example of content-based filtering:

Measures of similarity:

- Pearson correlation

$$pearson(x,y) = \frac{(x-\bar{x}).(y-\bar{y})}{\sqrt{(x-\bar{x}).(x-\bar{x}) \bullet (y-\bar{y}).(y-\bar{y})}}$$

- Cosine similarity

$$cosine(x,y) = \frac{(x.y)}{\sqrt{(x.x) \bullet (y.y)}}$$

- Euclidean
- Clustering algorithms
- Markov chain

Building a recommender engine

Artist rating dataset from Last.fm

Dataset:

- Number of unique users: **1,892**
- Number of unique artists: **17,631**
- Number of ratings (i.e. total number of rows): **92,834**
- ‘weight’ is the number of times a user listened to an artist

	userID	artistID	weight
0	2	51	13883
1	2	52	11690
2	2	53	11351
3	2	54	10300
4	2	55	8983

Data cleaning and wrangling:

No missing values found in the data

Python output of the .info() method:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 92834 entries, 0 to 92833

Data columns (total 3 columns):

userID   92834 non-null int64

artistID   92834 non-null int64

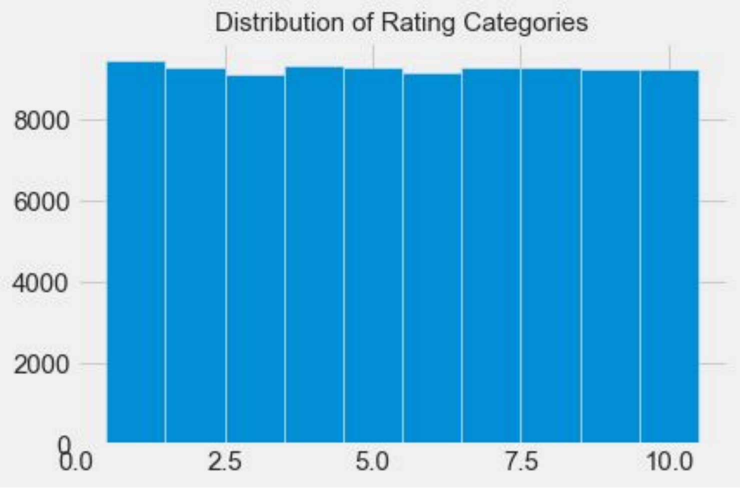
weight   92834 non-null int64

dtypes: int64(3)

memory usage: 2.1 MB

Rating system:

- Range of ‘weight’ variable: 1 - 352,698
- Predicting the exact number of times a user would listen to an artist is not of particular importance
- Define ‘rating’ by grouping ‘weight’ into 10 categories using its percentiles (i.e. min, 10th, 20th, 30th, …, 90th, max)



Springboard

Generating 80-20 train/test sets

Do this by putting 20% of each user's ratings in the test set and the remaining 80% in train set:

- Size of the train set: **74,241** rows
- Size of the test set: **18,593** rows

Evaluation criterion:

We will evaluate each version of the recommender engine using root-mean-squared-error:

$$RMSE = \sqrt{\frac{\sum(y-\hat{y})^2}{n}}$$

3 Versions of the collaborative filtering system

- **Baseline:** using average of ratings from other users:

  RMSE = 2.95

- **Cosine based:** Uses cosine similarity to assess similarity between users:

  RMSE = 2.95

- **Pearson based:** Uses pearson correlation coefficient to assess similarity between users:

  RMSE = 2.83

Conclusion:

The pearson based engine would be the selected solution for this problem

Next steps:

- Add user friendships as features in assessing similarity of users
- Try using fastai approaches