Playlist continuation challenge

Introduction

Recommender System for Playlist Continuation

- Started in 2018
- Hosted on AIcrowd
- More than 1000 submissions

Dataset:

- 1 Million Playlists
- 63 Million total songs
- 2 Million unique songs
- 345K unique artists

Playlists created by users from 2010 to 2017

Reduced Dataset

Sampled 10% of the playlists

- 100K Playlists
- 600K Unique Songs
- 100K Artists

From here, build a distributed recommender system that given a playlist, it recommend new relevant songs that continuate it.

How was the data structured

1,000 json files


```
{
"metadata": {"bla bla bla"},
"playlist": {
      "name": "musical",
      "collaborative": "false",
      "pid": 5,
      "modified_at": 1493424000,
      "num_albums": 7,
      "num_tracks": 12,
      "num_followers": 1,
      "num_edits": 2,
      "duration_ms": 2657366,
      "num_artists": 6,
      "tracks": [
         {
            "pos": 0,
            "artist_name": "Degiheugi",
            "track_uri": "spotify:track:7vqa3sDmtEaVJ2gcvxtRID",
            "artist_uri": "spotify:artist:3V2paBXEoZIAhfZRJmo2jL",
```

Data Visualization

Developed Systems

User-Based Collaborative Filtering;

Item-Based Collaborative Filtering;

Neural Network Approach:

Implemented by using a Denoising Autoencoder developed by the 2nd place winners of the challenge.

User-Based Collaborative Filtering - Data Preparation

How the playlists are encoded

1. Map each song in the playlist to a position.
2. Create the encoding vector:

   1 in the i-th position, if the song at position i is in the playlist;

   0 otherwise.

$$p = [0, 0, 0, 0, 0, 1, 0, 1, 0, 1]$$

Average of 66 songs in a playlist, the vectors are very sparse (x % spraseness).

Memory efficiency via pyspark's `SparseVector`, which stores only the indices and the values.

Generate the Recommendations

The pipeline for the recommendation, given the `SparseVector` of a playlist that has to be continuated, is the following:

1.