


Recommender Systems for the Spotify Million Playlist Dataset Challenge

Ana Ezquerro  GitHub

July 15, 2023

Abstract

In this document we provide a theoretical overview of the classical recommendation models implemented in the [recsys-smpd](#)  repository and the implementation and optimization details used to obtain an efficient and parallelized parser for the complete dataset.

1 Notation

Let $\mathcal{P} = \{p_i \mid i = 1, \dots, m\}$ be the complete set of $m = 10^6$ playlists (identified by a unique key) and $\mathcal{T} = \{t_1, \dots, t_n\}$ the complete set of $n \approx 2 \cdot 10^6$ tracks (identified by the track URI). Since the complete evaluation set used in the [official challenge](#) is not available, we randomly extracted $m_{\text{eval}} = 10^4$ playlists from the SMPD to use them as the test set. To simulate an input to the recommender system, we followed the description of the scenarios described in the [website](#) to hide a given number of tracks in the artificially generated test set with similar criteria.

Thus, our splitted SMPD consists of two non-overlapping sets of playlists:

- $\mathcal{P}_{\text{train}} = \{p_i^{\tau} \mid i = 1, \dots, m_{\text{train}}\}$ is the training set of $m_{\text{train}} = 10^6 - 10^4$ playlists.
- $\mathcal{P}_{\text{eval}} = \{p_i^{\varepsilon} = p_i^t \oplus p_i^x, i = 1, \dots, m_{\text{test}}\}$ is the evaluation set of m_{eval} playlists where each playlist p_i^{ε} is divided in two parts: a set of input tracks p_i^t that are used as input to a recommender system, and a set of hidden tracks p_i^x that are expected to be predicted by the model.

Each playlist $p = (t_1, \dots, t_{|p|}) \subseteq \mathcal{T}$ is defined as a tuple of all the tracks included in that playlist. Our recommender model $\mathcal{R} : (\mathcal{T}, \mathcal{P}) \rightarrow \mathbb{R}$ is defined by a set of parameters Θ optimized with $\mathcal{P}_{\text{train}}$ and assigns a score to each track and input playlist, with the restriction that $\mathcal{R}(t, p) = \emptyset$ if $t \in p$, meaning that it is not possible to recommend a track t that it is already included in p .

Given an input playlist p^t , the objective of the challenge is to estimate an accurate rating of each track $t \in \mathcal{T} - p^t$ such that, by ordering the highest 500 scored tracks $t_1^{\mathcal{R}}, \dots, t_{500}^{\mathcal{R}}$ ¹, we obtain a retrieved ordered list $p^{\mathcal{R}} = (t_1^{\mathcal{R}}, \dots, t_{500}^{\mathcal{R}})$ as similar as possible to p_i^x .

¹Note that $\mathcal{R}(t_i^{\mathcal{R}}, \cdot) < \mathcal{R}(t_j^{\mathcal{R}}, \cdot) \iff i < j$.

2 Implemented approaches

2.1 Baseline on popularity

Our first method recalls on the popularity assumptions. We computed the most popular tracks by obtaining the playlist frequency ($\Phi : \mathcal{T} \rightarrow \mathbb{N}$) in $\mathcal{P}_{\text{train}}$ (i.e. the number of playlists of the training set where a track appears at least once, Equation 1).

$$\Phi(t) = \sum_{i=1, \dots, m_{\text{train}}} \mathbb{I}\{t \in p_i^\tau\} \quad (1)$$

Thus, recommendations for each input playlist are generated by obtaining the most popular tracks that are not already included in that playlist.

$$\mathcal{R}_{\text{popular}}(t, p^\ell) = \begin{cases} \Phi(t) & t \notin p^\ell \\ -1 & t \in p^\ell \end{cases} \quad (2)$$

2.2 Neighbor-based Model

Assumptions:

1. $\mathcal{T} = (t_1, \dots, t_n)$ is an ordered set of tracks: and \mathcal{P} is the ordered set of playlists.
2. Each playlist p can be represented via a one-hot vector of size n indicating if j -th track is in the playlist p_i .

$$\vec{p} = (o_{p,t_1} \cdots o_{p,t_n}), \quad \text{where } o_{p,t_j} = \begin{cases} 1 & t_j \in p \\ 0 & \text{otherwise} \end{cases}, \quad j = 1, \dots, n \quad (3)$$

3. Analogously, each track $t \in \mathcal{T}$ can be represented via a one-hot vector where each component indicates if it is included in a i -th playlist or not.

$$\vec{t} = (e_{t,p_1} \cdots e_{t,p_m}), \quad \text{where } e_{t,p_i} = \begin{cases} 1 & t \in p_i \\ 0 & \text{otherwise} \end{cases}, \quad i = 1, \dots, m \quad (4)$$

This encoding might be seen as a vector representation of the ratings assigned by each user (playlist) to the different items of the catalog (tracks). Thus, we could take the Neighbor-based Recommendation Model to compute a rating for each pair (test user, item) based on (i) item/track neighbors or (ii) user/playlist neighbors.

User-based recommendation Let $\mathcal{V}_k : \mathcal{P} \rightarrow \mathcal{P}_{\text{train}}^k$ be the function which obtains the k nearest neighbors of a given playlist using the one-hot representation and a similarity metric $\varsigma : \mathbb{R}^d \rightarrow \mathbb{R}$. The user-based estimated rating of a track t for an input playlist p^ℓ is defined as:

$$\mathcal{R}_{\text{user}}^k(t, p^\ell) = \sum_{p^\tau \in \mathcal{V}_k(p^\ell)} \varsigma(\vec{p}^\ell, \vec{p}^\tau) \cdot o_{p^\tau, t} \quad (5)$$

Item-based recommendation : Let $\mathcal{I}_k : \mathcal{T} \rightarrow \mathcal{T}^k$ be the function which obtains the k nearest neighbors of a given track using the one-hot representation and the similarity metric ς . The item-based estimated rating of a track t for an input playlist p^ℓ is defined as:

$$\mathcal{R}_{\text{item}}^k(t, p^\ell) = \sum_{t_\ell \in \mathcal{I}_k(t)} \varsigma(\vec{t}, \vec{t}_\ell) \cdot e_{t_\ell, p^\ell} \quad (6)$$

Optimization via matrix representation of estimated ratings In order to optimize and parallelize the computation of similarities and estimated ratings, our implementation represents the complete set of training and input playlists with matrices using the one-hot encoding defined before.

Let $R_{\text{train}} \in \{0, 1\}^{m_{\text{train}} \times n}$ be the matrix of ratings in the training set, and $R_{\text{input}} \in \{0, 1\}^{m_{\text{eval}} \times n}$ the matrix of ratings in the evaluation set (only input ratings available). Formally:

$$R_{\text{train}} \sim r_{i,j} = \begin{cases} 1 & t_j \in p_i^\tau \\ 0 & \text{otherwise} \end{cases}, \quad R_{\text{input}} \sim r_{i,j} = \begin{cases} 1 & t_j \in p_i^\ell \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Let $S_{\text{user}} \in \mathbb{R}^{m_{\text{train}} \times m_{\text{eval}}}$ be the user similarity matrix, where each element $(i, j) = \varsigma(\vec{p}_i^\tau, \vec{p}_j^\ell)$ is the similarity between the playlists p_i^τ and p_j^ℓ , and $S_{\text{item}} \in \mathbb{R}^{n \times n}$ the item similarity matrix, where each element $(i, j) = \varsigma(\vec{t}_i, \vec{t}_j)$ is the similarity between tracks t_i and t_j . We took as similarity metric the cosine similarity (Equation 8).

$$\varsigma(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \|\vec{v}\|} \quad (8)$$

Following the Equation 5, 6 and 8, it is possible to compute similarity matrices by:

$$\begin{aligned} S_{\text{user}} &= (R_{\text{train}} \cdot R_{\text{input}}^\top) \oslash \|R_{\text{train}}\|_r^{-1} \ominus \|R_{\text{eval}}\|_r^{-1} \\ S_{\text{item}} &= (R_{\text{train}}^\top \cdot R_{\text{train}}) \ominus \|R_{\text{train}}\|_c^{-1} \oslash \|R_{\text{train}}\|_c^{-1} \end{aligned} \quad (9)$$

where $\|\cdot\|_r$ indicates the matrix normalization by rows and $\|\cdot\|_c$ indicates the matrix normalization by columns, and \ominus and \oslash operations represent the elementwise product between a vector and all the rows or columns, respectively, of a matrix (see Table 1 for a detailed definition).

Once similarity matrices have been computed (Equation 9), the estimated ratings can be computed with matrix operations:

$$\begin{aligned} \hat{R}_{\text{user}} &= \mathcal{Z}_k(S_{\text{user}}) \cdot R_{\text{input}} \\ \hat{R}_{\text{item}} &= R_{\text{input}} \cdot \mathcal{Z}_k(S_{\text{item}}) \end{aligned}, \quad \hat{R}_{\text{user}}, \hat{R}_{\text{item}} \in \mathbb{R}^{m_{\text{eval}} \times n} \quad (10)$$

where $\mathcal{Z}_k : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$ sets to zero the diagonal of a matrix and those elements with lower values than the k higher values of their corresponding row.

$$\mathcal{Z}_k(R) = \left(z_{i,j} \right)_{i=1 \dots m}^{j=1 \dots n} \quad \text{where} \quad z_{i,j} = \begin{cases} 0 & i = j \\ r_{i,j} \sum_{\ell=1, \dots, n} \mathbb{I}\{r_{i,j} < r_{i,\ell}\} < k \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

Table 1: Special operators. Let $R \in \mathbb{R}^{m \times n}$ be an arbitrary matrix where row i and column j is denoted as $\vec{r}_i \in \mathbb{R}^n$ and $\vec{c}_j \in \mathbb{R}^m$, respectively, and $\vec{u} = (u_1 \cdots u_m)$ and $\vec{v} = (v_1 \cdots v_n)$ be arbitrary vectors.

Operator	Definition	Formula
$\ \cdot\ _r$	$\mathbb{R}^{m \times n} \rightarrow \mathbb{R}^m$	$\ R\ _r = (\ \vec{r}_1\ \cdots \ \vec{r}_m\)$
$\ \cdot\ _c$	$\mathbb{R}^{m \times n} \rightarrow \mathbb{R}^n$	$\ R\ _c = (\ \vec{c}_1\ \cdots \ \vec{c}_n\)$
\ominus	$\mathbb{R}^{m \times n} \ominus \mathbb{R}^m = \mathbb{R}^{m \times n}$	$R \ominus \vec{u} = \begin{pmatrix} \vec{r}_1 \cdot u_1 \\ \vdots \\ \vec{r}_m \cdot u_m \end{pmatrix}$
\oslash	$\mathbb{R}^{m \times n} \oslash \mathbb{R}^n = \mathbb{R}^{m \times n}$	$R \oslash \vec{v} = \begin{pmatrix} \vec{c}_1 \cdot v_1 & \cdots & \vec{c}_n \cdot v_n \end{pmatrix}$

2.3 Pure SVD

The SVD (Singular Value Decomposition) approach consists of summarizing the information of the rating matrix $R \in \mathbb{R}^{m \times n}$ in three different matrices of lower dimension $h < n$ with matrix factorization methods:

$$R \approx U \cdot \Sigma \cdot V^\top \quad (12)$$

where $U \in \mathbb{R}^{m \times h}$, $\Sigma \in \mathbb{R}^{h \times h}$ and $V \in \mathbb{R}^{n \times h}$.

Compute the decomposition with training data Strictly, in a production environment we only dispose of the training ratings (R_{train}) to compute SVD. Given a new input rating, PureSVD algorithm projects its vector $\vec{r}_{\text{input}} \in \mathbb{R}^{1 \times n}$ to the latent space of h dimensions, obtaining a new feature representation $\vec{u}_{\text{input}} \in \mathbb{R}^{1 \times h}$ by using the projection matrix V :

$$\vec{u}_{\text{input}} = \vec{r}_{\text{input}} \cdot V \quad (13)$$

Rating estimation Let $U_{\text{input}} \in \mathbb{R}^{m_{\text{eval}} \times h}$ be the projected input matrix to the h -dimensional latent space via Equation 14.

$$U_{\text{input}} = R_{\text{input}} \cdot V \quad (14)$$

Then, new ratings can be computed by projecting the matrix U_{input} to the original space:

$$\hat{R}_{\text{svd}} = U_{\text{input}} \cdot \Sigma \cdot V^\top \in \mathbb{R}^{m_{\text{eval}} \times n} \quad (15)$$

2.4 Word2Vec

We adapted the proposal of [Parapar et al., 2013] to model the recommendation task as a query expansion task. Viewing the playlists as documents and tracks as words, we trained

Word2Vec model [Mikolov et al., 2013] with the sequence of tracks of each playlist. The result after the training process is a new d -dimensional space and a mapping function $\mathcal{M} : \mathbb{R}^n \rightarrow \mathbb{R}^d$ to project each track one-hot representation to a dense feature vector.

Let $W_{\text{train}} = \mathcal{M}(R_{\text{train}}^\top) \in \mathbb{R}^{n \times d}$ the projection of all one-hot representation tracks into the k -dimensional feature space. We can use the item-based formulation (Equation 6) using W_{train} instead of R_{train} . Thus, to compute the item similarity matrix $S_{\text{w2v}} \in \mathbb{R}^{n \times n}$ we use the track representations of Word2Vec and estimate the evaluation ratings with the item-based formulation (Equation 16).

$$\begin{aligned} S_{\text{w2v}} &= (W_{\text{train}} \cdot W_{\text{train}}^\top) \oslash \|W_{\text{train}}\|_r^{-1} \ominus \|W_{\text{train}}\|_r^{-1} \\ \hat{R}_{\text{w2v}} &= R_{\text{input}} \cdot \mathcal{Z}_k(S_{\text{w2v}}) \end{aligned} \quad (16)$$

3 Evaluation

We followed the [evaluation metrics](#) of the challenge. Let $p_i^{\mathcal{R}}$ be the **ordered** estimated playlist with a \mathcal{R} rating approach and $p_i^{\mathcal{X}}$ the **ordered** expected sequence of tracks (ground truth).

- **R-Precision** (ρ): Returns the ratio of correct tracks in the estimated playlist between the number of relevant tracks of the ground truth.

$$\rho(p_i^{\mathcal{R}}, p_i^{\mathcal{X}}) = \frac{1}{|p_i^{\mathcal{X}}|} \sum_{t \in p_i^{\mathcal{R}}} \mathbb{I}\{t \in p_i^{\mathcal{X}}\} \quad (17)$$

- **Normalized Discounted Cumulative Gain (nDCG)**: Measures the ranking quality of recommended tracks given importance to the ordered list.

$$\begin{aligned} \text{DCG}(p_i^{\mathcal{R}}, p_i^{\mathcal{X}}) &= \sum_{t_j \in p_i^{\mathcal{R}}} \frac{\mathbb{I}\{t_j \in p_i^{\mathcal{X}}\}}{\log_2(j+1)} \\ \text{iDCG}(p_i^{\mathcal{X}}) &= \sum_{j=1}^{|p_i^{\mathcal{X}}|} \frac{1}{\log_2(j+1)} \\ \text{nDCG}(p_i^{\mathcal{R}}, p_i^{\mathcal{X}}) &= \frac{\text{DCG}(p_i^{\mathcal{R}}, p_i^{\mathcal{X}})}{\text{iDCG}(p_i^{\mathcal{X}})} \end{aligned} \quad (18)$$

where t_j represents the j -th track of a playlist.

- **Recommended song clicks**: Slicing the recommended sequence of tracks $p_i^{\mathcal{R}}$ in slices of 10 tracks, the number of refreshes needed to encounter a track that is in the ground truth.

$$\text{clicks}(p_i^{\mathcal{R}}, p_i^{\mathcal{X}}) = \left\lceil \frac{1}{10} \cdot \arg \min_{j=1, \dots, 500} \{t_j \in p_i^{\mathcal{X}}\} \right\rceil \quad (19)$$

4 Results

References

- [Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.
- [Parapar et al., 2013] Parapar, J., Bellogín, A., Castells, P., and Álvaro Barreiro (2013). Relevance-based language modelling for recommender systems. *Information Processing Management*, 49(4):966–980.