Recommender System Challenge 2018

Şemsi Yiğit Özgümüş
10606748

Politecnico di Milano

February 5, 2019

Table of Contents

- **Application domain** : Music streaming service, where users listen to tracks (songs) and create playlists of favorite songs.
- **Goal** : discover which track a user will likely add to a playlist, therefore "continuing" the playlist.
- **Evaluation Method** : MAP@10 (Mean Average Precision)

$$\mathrm{AP@10} = \sum 10_{k=1} \frac{P(k) \times rel(k)}{min(m, 10)} \tag{1}$$

$$\mathrm{MAP@10} = \frac{\sum_{u=1}^{N} AP@10_u}{N} \tag{2}$$

Table of Contents

- ICM is built by binary mapping each track to corresponding album and artist
- The Matrix size is 20635 x 19412
- URM test dataset is created by separating 20% of the tracks of the target playlists
- Prioritizing songs using the randomness and sequential ordering didn't work.

Table of Contents

Following models are used seperately to get a baseline.

- User-based KNN Collaborative Filtering Recommender

Following models are used seperately to get a baseline.

- User-based KNN Collaborative Filtering Recommender
- Item-based KNN Collaborative Filtering Recommender

Following models are used seperately to get a baseline.

- User-based KNN Collaborative Filtering Recommender
- Item-based KNN Collaborative Filtering Recommender
- Item-based KNN Content Based Filtering Recommender

Following models are used seperately to get a baseline.

- User-based KNN Collaborative Filtering Recommender
- Item-based KNN Collaborative Filtering Recommender
- Item-based KNN Content Based Filtering Recommender
- Sparse Linear Methods with Bayesian Personalized Ranking

Following models are used seperately to get a baseline.

- User-based KNN Collaborative Filtering Recommender
- Item-based KNN Collaborative Filtering Recommender
- Item-based KNN Content Based Filtering Recommender
- Sparse Linear Methods with Bayesian Personalized Ranking
- SLIM with ElasticNet implementation

Hybrid Approach

- User Item Average Recommender $\rightarrow$ performed slightly better than Item KNN

Hybrid Approach

- User Item Average Recommender $\rightarrow$ performed slightly better than Item KNN
- KNN based + Slim Recommender $\rightarrow$ performed better significantly

Hybrid Approach

- User Item Average Recommender $\rightarrow$ performed slightly better than Item KNN
- KNN based + Slim Recommender $\rightarrow$ performed better significantly
- Sequential- Random Appoach $\rightarrow$ Using track ordering, didn't work

Hybrid Approach 2.0

- Bayesian Search and Saving offline Model features are added to the project $\rightarrow$ Huge speed improvement.

Hybrid Approach 2.0

- Bayesian Search and Saving offline Model features are added to the project $\rightarrow$ Huge speed improvement.
- P3Alpha and RP3Beta are added to the Hybrid Implementations

Hybrid Approach 2.0

- Bayesian Search and Saving offline Model features are added to the project → Huge speed improvement.
- P3Alpha and RP3Beta are added to the Hybrid Implementations
- Coefficient value ranges are increased

Hybrid Approach 2.0

- Bayesian Search and Saving offline Model features are added to the project → Huge speed improvement.
- P3Alpha and RP3Beta are added to the Hybrid Implementations
- Coefficient value ranges are increased
- Merging different Hybrids → Infinitesimal performance increase for a very long parameter tuning session

Final Hybrid Recommender

Contain 8 Stand alone Models

Final Hybrid Recommender

```
from models.KNN.User_KNN_CFRecommender import UserKNNCFRecommender
from models.KNN.Item_KNN_CFRecommender import ItemKNNCFRecommender
from models.Slim_mark2.Cython.Slim_BPR_Cython import Slim_BPR_Recommender_Cython as Slim_mark2
from models.Slim_ElasticNet.SlimElasticNetRecommender import SLIMElasticNetRecommender
from models.graph.P3AlphaRecommender import P3alphaRecommender
from models.graph.RP3BetaRecommender import RP3betaRecommender
from models.Slim_mark1.Cython.Slim_BPR_Cython import Slim_BPR_Recommender_Cython as Slim_mark1
from models.KNN.Item_KNN_CBFRecommender import ItemKNNCBFRecommender
from models.FW_Similarity.CFWBoostingRecommender import CFWBoostingRecommender
```

Final Hybrid Recommender

Using models offline decreased the training + prediction time from potential 30-40 minutes to roughly 1 minute

Final Hybrid Recommender

```
m = OfflineDataLoader()
self.m_user_knn_cf = UserKNNCFRecommender(self.URM_train)
folder_path_ucf, file_name_ucf = m.get_model(UserKNNCFRecommender.RECOMMENDER_NAME, training=self.submission)
self.m_user_knn_cf.loadModel(folder_path=folder_path_ucf, file_name=file_name_ucf)

self.m_item_knn_cf = ItemKNNCFRecommender(self.URM_train)
folder_path_icf, file_name_icf = m.get_model(ItemKNNCFRecommender.RECOMMENDER_NAME, training=self.submission)
self.m_item_knn_cf.loadModel(folder_path=folder_path_icf, file_name=file_name_icf)

self.m_item_knn_cbf = ItemKNNCBFRecommender(self.URM_train,self.ICM)
folder_path_icf, file_name_icf = m.get_model(ItemKNNCBFRecommender.RECOMMENDER_NAME, training=self.submission)
self.m_item_knn_cbf.loadModel(folder_path=folder_path_icf, file_name=file_name_icf)

self.m_slim_mark1 = Slim_mark1(self.URM_train)
folder_path_slim, file_name_slim = m.get_model(Slim_mark1.RECOMMENDER_NAME, training=self.submission)
self.m_slim_mark1.loadModel(folder_path=folder_path_slim, file_name=file_name_slim)

self.m_slim_mark2 = Slim_mark2(self.URM_train)
folder_path_slim, file_name_slim = m.get_model(Slim_mark2.RECOMMENDER_NAME, training=self.submission)
self.m_slim_mark2.loadModel(folder_path=folder_path_slim, file_name=file_name_slim)

self.m_alpha = P3alphaRecommender(self.URM_train)
folder_path_alpha, file_name_alpha = m.get_model(P3alphaRecommender.RECOMMENDER_NAME, training=self.submission)
self.m_alpha.loadModel(folder_path=folder_path_alpha, file_name=file_name_alpha)

self.m_beta = RP3betaRecommender(self.URM_train)
folder_path_beta, file_name_beta = m.get_model(RP3betaRecommender.RECOMMENDER_NAME, training=self.submission)
self.m_beta.loadModel(folder_path=folder_path_beta, file_name=file_name_beta)

self.m_slim_elastic = SLIMElasticNetRecommender(self.URM_train)
folder_path_elastic, file_name_elastic = m.get_model(SLIMElasticNetRecommender.RECOMMENDER_NAME,
                                                     training=self.submission)
self.m_slim_elastic.loadModel(folder_path=folder_path_elastic, file_name=file_name_elastic)
```

Final Hybrid Recommender

Final Submission score with the Hybrid approach was **0.09372** on public and **0.09280** on private leaderboard

Thank you for your Attention

Any Questions ?