POLIMI RECSYS CHALLENGE 2018

Daniele Montesi - Federico Piccinini

POLITECNICO
MILANO 1863

OVERVIEW

Objective

Create the best recommender system for a music streaming service.

Our tools

- PyCHARM
- Jupyter notebook

Results

Ranked 1st at the Polimi RecSys challenge 2018

MAP@10 = 0.10020

CODE STRUCTURE

Recommender

- __init__(params) //command pattern
- **fit**(URM)
- **get_expected_ratings**(playlist_id)
- **recommend**(playlist_id)
- Use of cosine similarity

Runner

- **run**(is_test, recommender_obj, split_method)

Other objects

- **Helper**: contains shared and support methods
- **Evaluator**: evalution and split methods

ALGORITHMS

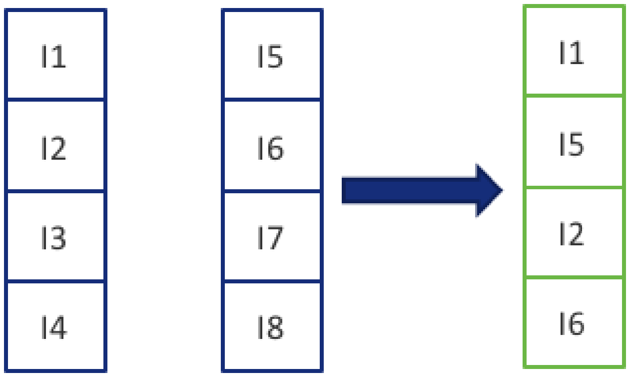
Hybrid weights

	Sequential playlists	Short randomic playlists	Long randomic playlists
User_CF	0.5	0.03	0.03
Item_CF	0.1	0.25	0.35
CBF	0.72	0.15	0.2
ALS	0.14	0.3	0.3
SLIM BPR	2.06	0.6	0.22
SLIM ElasticNet	0.07	1.5	1.5

HYBRIDIZATION

We experimented many methods:

1- Round Robin



2- Weighted sum of *normalized* expected ratings

3- Weighted sum of expected ratings ← Best one

FIRST ENSEMBLES

ITEM CF

USER CF

CBF

MAP@10 = 0.08912 (public)

Adding the fourth ingredient

SLIM BPR

MAP@10 = 0.09218 (public)

		Copenaghen				
3	▲ 2	10498975_10452804	0.09275	42	1d
4	▼ 3	10652325_10686344	0.09262	27	2h
5	▲ 4	10542302_10544314	0.09218	55	now

Your Best Entry ↑

You advanced 1 place on the leaderboard!

Your submission scored 0.09218, which is an improvement of your previous score of 0.09150. Great job!

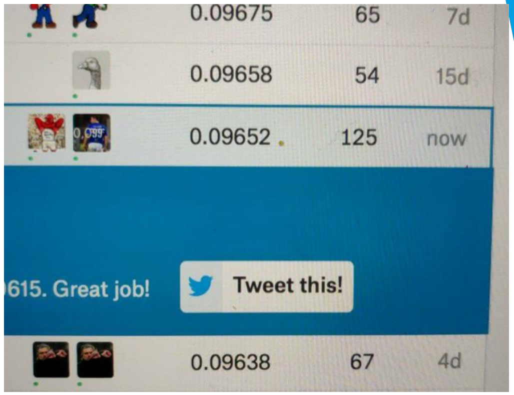
Tweet this!

NEW INGREDIENTS

ALS (alternative least square)

- Efficient cython implementation of MF
- technique adapted by “implicit” module by *benfred*

MAP@10 = 0.09652 (public)



ElasticNET

- SLIM with penalization
- Adapted from Massimo Quadrana’s version

WHEN MAGIC STARTS

2 types of playlists orders were analyzed

- Randomically ordered → **MAP@10 = 0.105** (local)
- Sequentially ordered → **MAP@10 = 0.0645** (local)

2 approaches were followed:

- **BIB (Before Is Better)** - awful performance
- **LIB (handcrafted *Tail boost*)** - very **interesting** performance

TAIL BOOST

- Old Sequentially ordered → MAP@10 = 0.0645 (local)
- New Sequentially ordered → $\boxed{\text{MAP@10} = 0.0802 \text{ (local)}}$

$\vec{x}_u = \begin{bmatrix} 3 & 36 & 7 & 54 & 154 & 46 & 79 & 69 & 34 \end{bmatrix}$ **User's tracks**

$\vec{x}_u = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$ **User's ratings**

$\vec{x}_u = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$ **Boosted user's ratings**

***Step* = 1**

LastN = 4

MAP@10 = 0.1002 (private)

SPLIT & TUNING

We created multiple splitting methods aimed at imitating the original split

- Sequential or randomic playlists
- Imitation of the statistical distribution of the original dataset
- Selection of playlists based on their length

Tuning principles:

- Bottom-up tuning & grid search
- Use of automatic tools to better optimization
- Hold-out 10-20% of data used for testing

- Round Robin
- Normalization of intermediate ratings
- Before is better
- MF BPR
- Track's duration

- Weighted sum of ratings
- Tail boost
- SLIM BPR + SLIM ElasticNet
- ALS
- MaurizioFD's repo

CONCLUSION

Our journey

1- October – First approach with RecSys

2- November – SLIM lovers and first hybrids

3- December: explored new methods of parameter tuning based on playlists length with *slightly* improvements

4- Mid December: discovered a new working MF technique (ALS)

5- Late December: Santa told us the ingredient for success (Tail Boost)

6- Mid January – Won POLIMI RecSys Challange 2018

ALGORITHMS (1)

Content Based Filtering

- Album
    - BM_25
    - KNN = 45
    - Shrink = 8
    - Weight = 0.85
- Artist
    - TF_IDF, l2 normalization
    - KNN = 25
    - Shrink = 0
    - Weight = 0.15

ALGORITHMS (2)

User Collaborative Filtering

- KNN = 140
- Shrink = 0

Item Collaborative Filtering

- TF_IDF, l2 norm
- KNN = 310
- Shrink = 0

SLIM BPR

- Epochs = 40
- topK = 200
- Sgd_mode = adagrad
- Batch_size = 1
- Lambda_i = 0.01
- Lambda_j = 0.001

ALGORITHMS (3)

SLIM ElasticNet

- Alpha = 1e-4
- L1-ratio = 0.1
- Max_iter = 100
- topK = 100

ALS

- N_factors = 300
- Regularization = 0.15
- Iterations = 30

TEAM

Daniele Montesi

Data Science student
@ EIT Digital Master
School (POLIMI, KTH)

danmontesi

Federico Piccinini

Data Science student
@ EIT Digital Master
School (POLIMI, TU/e)

APPiccio