

# RecSys Challenge

## MultiBeerBandits

---

Emanuele Ghelfi    Leonardo Arcari

[https://github.com/MultiBeerBandits/recsys\\_challenge\\_2017](https://github.com/MultiBeerBandits/recsys_challenge_2017)

January 24, 2018

Politecnico di Milano



# Our Approach

---

Basic models:

- User-based filtering
- Item-based filtering
- Content-based filtering
- SVD (Content and Collaborative)
- SLIM (RMSE and BPR)
- Matrix Factorization (RMSE and BPR)
- Factorization Machine

Hybrid models:

- ICM + URM Content Based
- CSLIM BPR
- Ranked Lists Merging
- Similarity Ensemble

## What Worked

---

**ICM + URM Content Based**

**Rationale:** the similarity between two items is strongly influenced by their attributes and by the number of users they've in common.

$$ICM_{augmented} = \begin{bmatrix} \alpha_1 \cdot ICM \\ \alpha_2 \cdot URM \end{bmatrix}$$

Here users are seen as features of an item.

## Successful Preprocessing

- Feature weighing (Hyperparameters to tune)
- TFIDF + topK filtering (on tags)



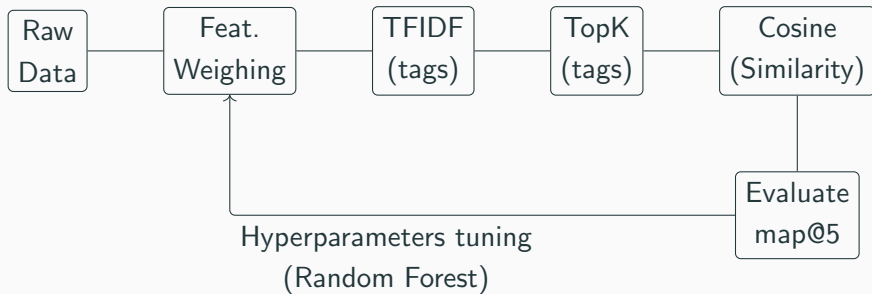
## Unsuccessful Preprocessing

- TFIDF on all the features
- Clustering on continuous-valued features (duration and playcount)
- Inference on missing data (album, duration and playcount)
- Aggregated features (tags) <sup>1</sup>

---

<sup>1</sup>Daniele Grattarola et al 2017. Content-Based approaches for Cold-Start Job Recommendations

# Tuning process



**CSLIM BPR**

## Preprocessing

Tag aggregation:

- Select the topK most popular tags
- For each tuple T of tags compute the logical AND
- The result is a new feature, describing the items that share tags in T

## Our Improvements

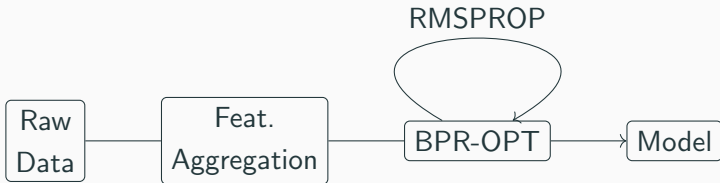
- *Momentum* update rule implemented
- Samples to train the model are drawn from ICM and URM with different probabilities (in our case sampling more from ICM than URM gave better results).

## Results

Even if this approach is a model-based one, we were able to achieve similar results with respect to the memory-based one:

*ICM + URM Content Based.*

## Training process



# Similarity Ensemble

## Rationale:

- Combining different sets of features is very hard. Some features are more present than others and stating their relative importance is even harder
- Different models capture different aspects of the similarity between items

Our approach was to optimize each model separately and combine their similarity matrices.



# Similarity Ensemble

## The model

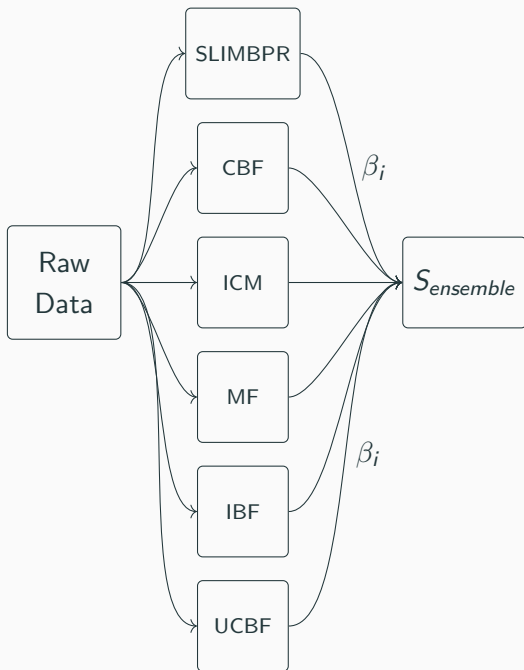
$$S_{ensemble} = \sum_{i \in M} \beta_i S_i$$

Where  $M$  is the set of models.

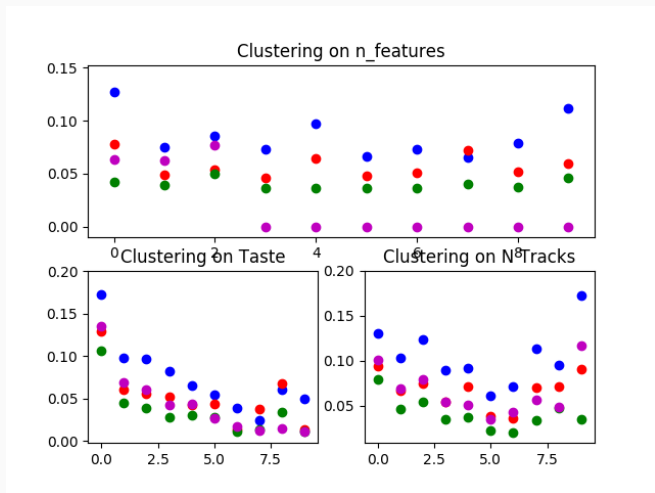
The coefficients  $\beta_i$  are hyperparameters that we tune with Random Forest.

### Properties:

- The similarity of each item with the others must be normalized
- For each item in  $S_{ensemble}$  we keep only the topK similar items
- $S_{ensemble}$  is again normalized



# Similarity Ensemble with Clustering



**Figure 1:** This figure shows the performance of some models on users clustered with respect to different features.

# Similarity Ensemble with Clustering

## The model

$$S_{ensemble_j} = \sum_{i \in M} \beta_{ij} S_i$$

$$\hat{r}_i^j = r_i \cdot S_{ensemble_j}$$

Let  $i$  be a user belonging to cluster  $j$ , the prediction  $\hat{r}_i^j$  is computed from the ensemble similarity matrix personalized for that cluster ( $S_{ensemble_j}$ ).

## Lessons Learned

---

- Exploration/Exploitation trade off
- No Free Lunch theorem (No superior algorithm)
- Do Ensemble

**Thank you all!**