Artist-driven layering and user's behaviour impact on recommendations in a playlist continuation scenario

Sebastiano Antenucci, Simone Boglio, Emanuele Chioso, Ervin Dervishaj, Shuwen Kang, Tommaso Scarlatti and Maurizio Ferrari Dacrema

Politecnico di Milano

The Challenge

The **Spotify RecSys Challenge 2018** focuses on the music recommendation task, in particular on automatic playlist continuation. Two parallel tracks:

- *Main track:* only the Million Playlist Dataset can be used to train the model.
- *Creative track:* external, public and freely available data are allowed too.

Preprocessing

- To face the **cold-start problem**, we apply information retrieval techniques to build a feature space from playlists titles.

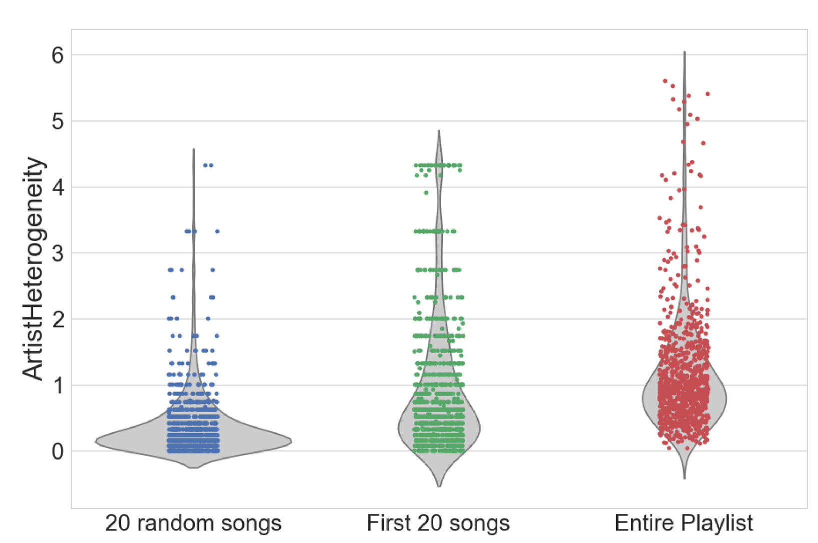

Figure 1: Artist Heterogeneity for *1K* long playlists. The gray area shows the ArH distribution over three sampling strategies.

- Playlists sometimes exhibit a common underlying structure due to the way a user fills them. We define a new measure to capture the **heterogeneity of artists** in a playlist.

$$ArH_p = \log_2\left(\frac{|\ uniqueTracks_p\ |}{|\ uniqueArtists_p\ |}\right)$$

Algorithms

Our model is made of five well known algorithms:

1. Collaborative Filtering - Track based
2. Collaborative Filtering - Playlist based
3. Content Based Filtering - Track Based
4. Content Based Filtering - Playlist based
5. Personalized Top Popular

Ensemble

Different algorithms are better suited for subsets of playlists with specific characteristics. The final model is a **weighted sum** of score predictions of our algorithms, taking into account the length of the playlist and the position of the tracks. we take advantage of the diversity in the predictions.

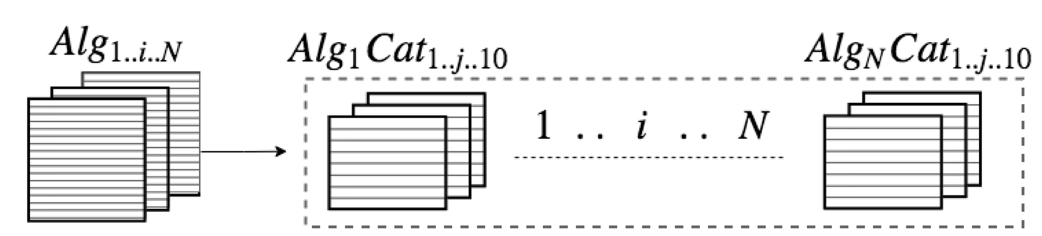

Figure 2: Category splitting of each of the *N* different algorithms.

Postprocessing

We improve our score leveraging on **domain-specific patterns** of the dataset, developing ad-hoc boost techniques. Starting from a list of K predicted tracks for a playlist p and for each $k \in K$ they boost the precomputed $Score_{p_k}$:

$$Score_{p_k} = Score_{p_k} + Boost_{p_k}$$

Gap Boost: an heuristic which applies to playlists with known tracks not in order that tries to increase weight of tracks which seems to better "fit" between all the gaps of the playlist.

Private Leaderboard Scores

Creamy Fireflies team ranked **2nd** in the Creative track and **4th** in the Main track. The following tables show for each metric the final score and the relative rank.

Main track			Creative track		
R-prec	0.2201	3rd	R-prec	0.2197	2nd
NDCG	0.3856	3rd	NDCG	0.3845	2nd
Clicks	1.9335	7th	Clicks	1.9252	4th

External Datasets

We tried several external datasets to enrich the *Million Playlist Dataset*. At last, we used **Spotify API** to retrieve tracks popularity and audio features such as: acousticnes, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence, popularity.

Dataset Name	Data Type	Year
#nowplaying music	Listening behavior	2018
#nowplaying playlists	Playlist	2015
MLHD	Listening behavior	2017
FMA	Audio Features	2017
MSD	Audio features	2011
Spotify API	Audio features, pop	2018

Table 1: External datasets explored for the creative track. Listening behaviour refers to timestamps of listening events.

Creative Track

CBF using ten additional features:

1. For each feature, divide the tracks into 4 clusters with equal number of elements.
2. Considering feature clusters as a 3rd dimension, split the dense ICM into 4 sparse layers.
3. Concatenate 4 layers horizontally in order to create a final sparsified ICM and apply CBF.

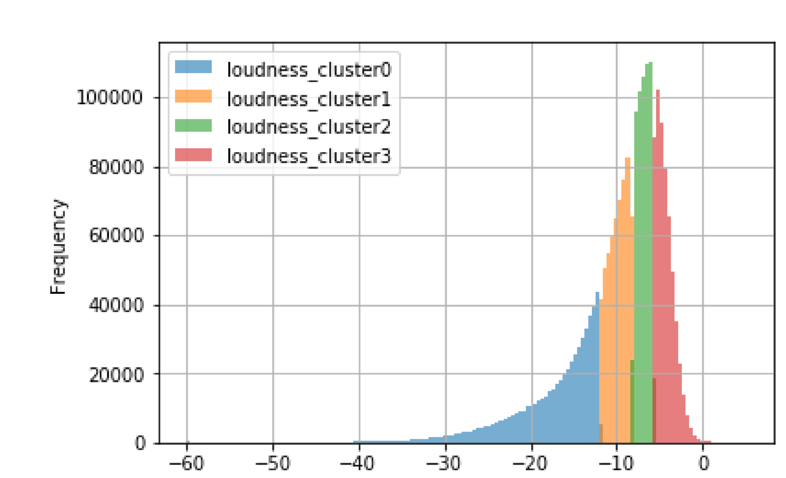

Figure 3: Layered ICM over *loudness* feature (200 artists).

Conclusion

Our architecture is built in a **simple** and **modular** way. It can be easily extended with additional features coming from different datasets and new techniques can be implemented with no impact on the pre-existent work flow. Furthermore our architecture relies on an efficient **Cython** implementation of the most computationally intensive tasks, which allows to keep the time and space complexity under a low threshold.

Computational Requirements

To run the entire model we use a AWS memory optimized cr1.8xlarge VM with 32 vCPU and 244 GiB of RAM.

Step	Time	RAM
Model Creation	1.5h	80GB
Bayesian Optimization	16h	~15GB
Ensemble	5m	<8GB
Postprocessing	8m	<8GB

Table 2: Computational requirements for each step of the recommendation process.

Acknowledgements

The authors would like to thank Prof. Paolo Cremonesi for his continuous support and for inspiring us to choose the team's name.

Contact Information

- Email: creamy.fireflies@gmail.com
- Github: github.com/MaurizioFD/spotify-recsys-challenge

POLITECNICO MILANO 1863

CREAMY FIREFLIES