POLITECNICO
MILANO 1863

RECOMMENDER SYSTEM 2017 CHALLENGE

Simone Boglio 10394623

Simone Brunitti 10434309

Team: I_Taz

TABLE OF CONTENTS

INTRODUCTION

OVERVIEW

In this presentation we will show our work done for the Recommender System 2017 Challenge Polimi hosted on Kaggle.

After some initial considerations, we analyse the basic recommender algorithms. Finally we show the final solution that allowed us to reach the first position in the competition.

POLITECNICO
MILANO 1863

kaggle

TOOLS AND LIBRARIES

- Language used: Python
- Tools used: Jupyter Notebook, PyCharm
- External libraries used: Implicit (just the function all_pairs_knn())

From the Implicit libraries we use only the function all_pairs_knn() that receives as input a matrix and a value K, it calculates the dot product of the matrix with its transpose and returns it keeping only the top K values.

This function is written in Cython (and compiled in C), because of that it is much faster than an equivalent function written directly in Python.

Example:

similarity = all_pairs_knn(urm_matrix, K = 100)

DATA USED

For each data file we show what data we use (green) and which not (red)

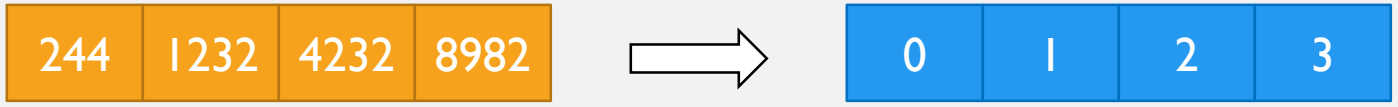
- train_final
  - playlist_id - ID of the playlist
  - track_id - ID of the track
- tracks_final
  - track_id - ID of the track
  - artist_id - ID of the artist
  - duration - duration of the track in seconds
  - playcount - number of times the track has been played
  - album - ID of the album in which the track is contained
  - tags - list of significant keywords related to the track

DATA USED

- playlists_final
  - created_at - timestamp of playlist creation
  - playlist_id - ID of the playlist
  - owner - ID of the user who created the playlist
  - title - encoding of the words in the title of the
  - duration - total duration of the playlist in seconds

DATA PREPROCESSING

- First problem: deal with a relatively big amount of data in a reasonable time
- Solution: use a dictionary to align IDs (tracks, artist, album, tag) in order to reduce the size of the all the matrices we use in the computations
- At the end of all the computation, the IDs are then translated back to their original values
- This led to a huge time and space improvement (e.g. apply mask, filter recommendations)
- URM without preprocessing → sparse matrix with millions of rows and columns
- URM with dictionary →sparse matrix with hundreds of thousands of columns



RECOMMENDER ALGORITHMS

CONTENT BASED RECOMMENDERS

CB recommenders using information about tracks:

1. ICM with artist IDs
2. ICM with album IDs
3. ICM with both album IDs and artist IDs
4. ICM with tag IDs

CB recommenders using information about playlists:

1. UCM with owner IDs
2. UCM with title's words IDs

CONTENT BASED RECOMMENDERS

Once the ICM has been built, all the content based filters work in the same manner:

- Calculate a similarity matrix between tracks as $S = ICM * ICM^T$
- Apply KNN to the similarity matrix to keep only the most relevant k values for each track
- Calculate the rating matrix as $R = URM * S^T$

(similar operations for UCM)

COLLABORATIVE FILTERING RECOMMENDERS

Two main types of CF recommenders:

- User based:
  - Symmetric similarity
  - Asymmetric similarity
- Item based:
  - Symmetric similarity
  - Asymmetric similarity

COLLABORATIVE FILTERING RECOMMENDERS

The item based filter works in the following way:

- Calculate the similarity matrix:
  - Symmetric similarity: $S = URM^T * URM$
  - Asymmetric cosine similarity $s_{ij} = \frac{\sum r_{ui} \cdot r_{uj}}{(\sum {r_{ui}}^2)^{\alpha} + (\sum {r_{uj}}^2)^{1-\alpha}}$

    (with $\alpha = \frac{1}{2}$ we have the normal cosine similarity)
- Apply KNN to the similarity matrix to keep only the most relevant k values for each track
- Calculate the result matrix as $R = URM * S^T$

(similar operations for CF user based)

OTHER RECOMMENDERS

- SLIM BPR
- SSLIM BPR (add side information of artists and albums)

IMPROVEMENTS

ICM/UCM/URM MATRIX

Before calculating the similarity matrix, we adjust the values of the ICM (same for UCM and URM) with one of these techniques:

- None (matrix of 1 and 0)
- Normalization on row or column
- TF-IDF
- BM25 (Okapi BM25, ranking function used in search engines)

SIMILARITY MATRIX

After having calculated the similarity matrix, we apply one of these techniques on the matrix:

- None
- Keep the top K nearest neighbors
- Remove the diagonal

(The last operations could seem tricky, but we will see that this difference is important)

RATING MATRIX

After having calculated the ratings ($R = URM * S^T$) we apply one of these techniques on the matrix:

- None
- Normalize on row (user)
- Normalize on column (track)

(In reference to the previous slide, keep or remove the diagonal on the similarity matrix can lead to different result)

EXAMPLE: CONTENT BASED WITH ARTIST

MAP@5 values of content based recommenders on artist.

At each step we add one technique to the algorithm:

1. Basic recommender: 0.0431
2. Apply BM25 on ICM: 0.0521
3. Normalize rating matrix on tracks (columns): 0.0619
4. Remove diagonal on the similarity matrix: 0.0701

FINAL ALGORITHM

OVERVIEW

- Final solution: combination of many different recommender systems
- Main technique → linear combination of the results obtained from different recommender algorithms

Recommender	KNN	ICM/UCM/URM	Remove diagonal similarity	Rating Matrix	MAP@5	Used in final solution
CB artist	55	BM25	yes	norm column	0.0701	yes
CB album	55	BM25	yes	norm column	0.0712	yes
CB artist + album	55	BM25	no	norm column	0.0790	yes
CB tags	45	BM25	yes	norm column	0.0400	yes
CB owners	65	BM25	no	norm column	0.0226	yes
CB title's words	40	BM25	no	norm row	0.0103	no
CF IB symmetric	100	BM25	no	norm column	0.0729	yes
CF IB asymmetric (α = 0.30)	120		no		0.0768	no
CF UB symmetric	60	BM25	no	norm row	0.0661	yes
SLIM	100		no		0.0541	no
SSLIM (artist and album side information)	100		no		0.0636	no

(Note: MAP@5 is calculated on an offline test set, it is not calculated on the online test set of Kaggle)

FINAL SOLUTION

This is the final formula that led us to the first place of the competition:

$$\mathrm{R} = 2.25 * CF_{itemBased} + 1.82 * CF_{userBased} + 1.02 * CB_{albumAndArtist} + 0.19 * CB_{tags} + 0.15 * CB_{album} + 0.12 * CB_{artist} + 0.72 * CB_{owners}$$

With this solution we reach a MAP@5 value of 0,10845 on the online test set of the competition.

FINAL SOLUTION

Reccomender	MAP
1.02 * $CB_{artist+album}$	0.0790
+ 2.25 * $CF_{itemBased}$	0.1004
+ 1.82 * $CF_{userBased}$	0.1034
+ 0.19 * CB_{tag}	0.1038
+ 0.15 * CB_{album} + 0.12 * CB_{artist}	0.1058
+ 0.72 * CB_{owner}	0.1084

Time to compute the rating matrix: 30.8 sec
Time for recommendations (10k playlists): 6.1 sec

CONCLUSION

CONSIDERATIONS

- Two important steps that increase the score of the single recommenders:
  - BM25 on ICM/UCM/URM
  - Normalize rating matrix

- In the final solution we use only recommenders with BM25 applied on the initial matrix, we discard for example asymmetric cosine because it led us to worst results when summing it with others recommenders, even though it has the better score among collaborative filtering.

DISCARDED APPROACHES

- All the approaches involving machine learning (SLIM, SSLIM, Matrix Factorization, Graph) didn't lead to good results
- All other ensemble techniques (picking alternatively recommendations from different results matrices and mix them, feeding the results of an algorithm to another one etc.) didn't work as well

FUTURE WORKS

These could be the possible approaches that might be used to improve our result:

- Filter dataset
- Clustering
- Measure confidence of recommendations

REFERENCES

- Competition: https://www.kaggle.com/c/recommender-system-2017-challenge-polimi
- Asymmetric cosine, from Million Song Dataset Challange: http://www.math.unipd.it/~aiolli/PAPERS/paperIIR.pdf
- BPR for implicit feedback: https://arxiv.org/ftp/arxiv/papers/1205/1205.2618.pdf