MMR: Exercise 2 Report

Viktor Loreth
JKU AI Student
viktor.loreth@pm.me

Tobias Katsch
JKU AI Student
tobias.katsch42@gmail.com

Harikrishnan Kesevan
JKU AI Student
kesavanharikrishnan@gmail.com

ABSTRACT

In this report, we introduce a song recommender framework using various features from the Music4All-Onion dataset. The evaluation is done by finding similar genres.

CCS Concepts

• Music Information Retrieval • Music Recommender Systems• Audio Features • Natural Language Processing • Video• Image • K-Nearest Neighbars

1. Introduction

Music Recommender System are an important part of the music industry. YouTube, Spotify and other music apps are trying to optimize the listening experience of users by recommending the best songs. Several different approaches exist to retrieve songs and evaluate the recommendations. In this paper, we focus on using K-Nearest Neighbors on extracted features from songs.

2. Dataset: Music4All-Onion

In our framework we are using features provided from the Music4All-Onion dataset. In our evaluated framework 1) lyrics preprocessed by tf-idf, 2) lyrics preprocessed by word2vec 3) lyrics preprocessed by BERT 4) Block-Level Features, 5) Image data extracted from music videos using Res-Net, 6) spectral features a 7) Mel Frequency Cepstral Coefficients. We choose those 6 features to evaluate different aspects of a music video, focusing on lyrics, audio and images.

2.1 Preprocessing the dataset

The Music4All Dataset includes 76.115 tracks and most extracted feature sets have more than 1.000 features. Calculating the similarity metrics for every track has a very high runtime. Therefore we are using LSA to reduce the dimensionality to k=10 features. We are losing accuracy through this step but considering the long runtime we do not have other options except renting a high-performance server.

2.2 Data analysis: Song genres

How similar 2 songs are and how much a individual person likes 2 songs is highly subjective. Collaborative filtering can give a reasonable result on how similar 2 songs are. However, in this method we are using the genre of a song to evaluate the similarity. If 2 songs do have the same genre, they are similar. Of course, some songs do have several genres in which case the “main” genre is considered. (see Evaluation metrics)

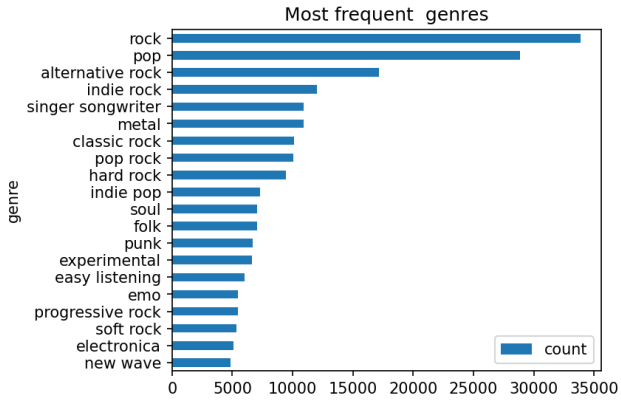

Figure1: Most frequent genres

In the above figure one can see the most similar genre rock. Using basic probability theory, one can achieve a 47% accuracy baseline by always recommending rock songs. In the following methods, we are trying to achieve better results.

In total, 1.670 genres do exist. About 800 genres do have less than 10 songs. Using KNN or other supervised machine learning models do have a hard time learning those underrepresented samples. Additionally, many genres are very similar in style, e.g.: progressive rock and rock. Achieving a high accuracy across the whole dataset is very difficult. A good target is about 70-80 % precision, considering the evaluation method chosen.

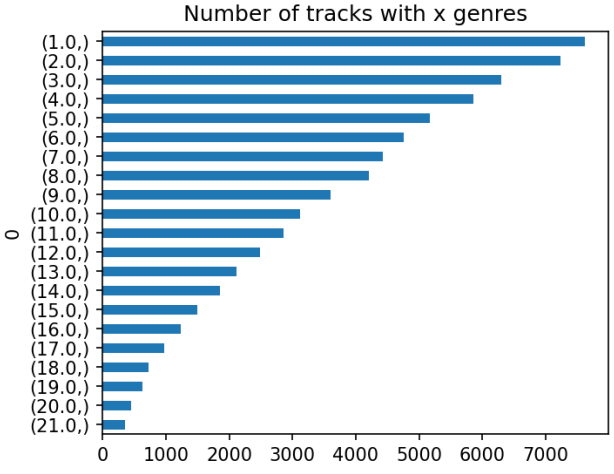

Figure2: Genre Distribution

For the nDCG and the MMR metric the position of the genre is important. Therefore, we analyzed how rare multiple genres are. Without going in-depth into this analysis, one can quickly see that

many songs only do have 1-5 genres, which will negatively impact our accuracy, specially in MMR and nDCG.

3. Methodic

3.1 Processing steps.

After having done the preprocessing we have suitable labeled data to train a supervised model. Still, having to calculate the kNN metrics for every single song more than 67000 times would result in a very long runtime. To counteract this problem, we are using a KDTree. For the recommendations itself we are running a k=[10-100] recommender to get a reasonable sample size. The top k most similar songs according to the features will be taken as recommendation.

Furthermore, we added a song comparison matrix. If 2 songs share a genre, they both are considered as successful recommendation to each other. These speeds up computation time.

3.2 Evaluation Methods

Before discussing our results, the evaluation metrics will be explained.

Avg. Precision compares the genres of the song with any genre of the recommendation. Due to the forgiving nature of this method a high accuracy of 80% would be a reasonable goal. MMR and nDCG punishe less accurate recommendations. Achieving 60% is good enough.

4. Results

We sampled our results by choosing 1 % of the data as evaluation data randomly. Our average accuracy is about 27% not dependent on the method used. The variance is 14%, which is quite high.

4.1 Precision Recall

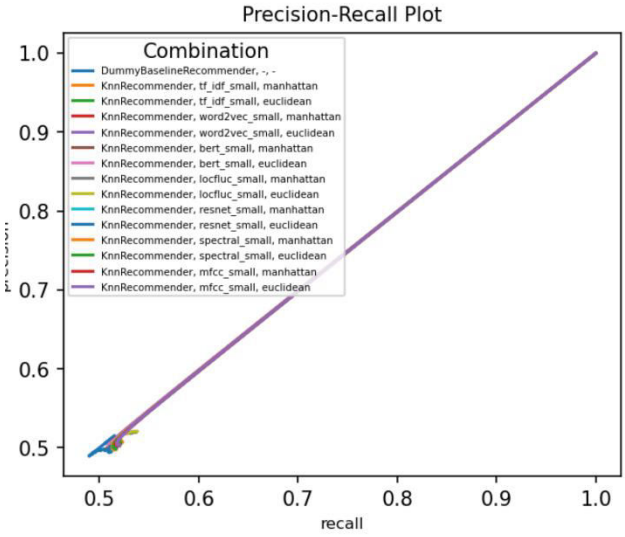

Figure 4: Precision-Recall Plot

4.2 Discussion of Results

We believe that our intuitive wrong. Having the same accuracy for any feature set required is not realistic. Compared to report 1, our numbers got worse, but there is more certainty in the calculations.

The Precision/Recall plot shows a linear relationship between Precision and Recall.

For the next project stage, we will try to investigate our code to get to reasonable numbers (50-70% accuracy).

5. REFERENCES

[1] Marta Moscati, Emilia Parada-Cabaleiro, Yashar Deldjoo, Eva Zangerle, and Markus Schedl. 2022. Music4All-Onion — A Large-Scale Multi-Faceted Content-Centric Music Recommendation Dataset. In Proceedings of the 31st ACM Int'l Conference on Information and Knowledge Management (CIKM'22), Oct. 17–21, 2022, Atlanta, GA, USA. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557656

Figure 3: Evaluation results

	recommender	embedding	metric	k	avg. precision	mmr	ndcg
0	DummyBaselineRecommender	-	-	10	μ=0.223, σ=0.146	μ=0.648, σ=0.379	μ=0.494, σ=0.319
1	DummyBaselineRecommender	-	-	50	μ=0.242, σ=0.144	μ=0.653, σ=0.372	μ=0.498, σ=0.293
2	DummyBaselineRecommender	-	-	100	μ=0.246, σ=0.144	μ=0.653, σ=0.372	μ=0.502, σ=0.284
3	KnnRecommender	tf_idf_small	manhattan	10	μ=0.272, σ=0.123	μ=1.0, σ=0.0	μ=0.618, σ=0.256
4	KnnRecommender	tf_idf_small	manhattan	50	μ=0.258, σ=0.138	μ=1.0, σ=0.0	μ=0.547, σ=0.269
5	KnnRecommender	tf_idf_small	manhattan	100	μ=0.257, σ=0.139	μ=1.0, σ=0.0	μ=0.539, σ=0.265
6	KnnRecommender	tf_idf_small	euclidean	10	μ=0.271, σ=0.121	μ=1.0, σ=0.0	μ=0.612, σ=0.253
7	KnnRecommender	tf_idf_small	euclidean	50	μ=0.257, σ=0.138	μ=1.0, σ=0.0	μ=0.547, σ=0.268
8	KnnRecommender	tf_idf_small	euclidean	100	μ=0.256, σ=0.139	μ=1.0, σ=0.0	μ=0.537, σ=0.264
9	KnnRecommender	word2vec_small	manhattan	10	μ=0.268, σ=0.118	μ=1.0, σ=0.0	μ=0.608, σ=0.244
10	KnnRecommender	word2vec_small	manhattan	50	μ=0.255, σ=0.133	μ=1.0, σ=0.0	μ=0.542, σ=0.258
11	KnnRecommender	word2vec_small	manhattan	100	μ=0.254, σ=0.135	μ=1.0, σ=0.0	μ=0.533, σ=0.255
12	KnnRecommender	word2vec_small	euclidean	10	μ=0.268, σ=0.117	μ=1.0, σ=0.0	μ=0.603, σ=0.242
13	KnnRecommender	word2vec_small	euclidean	50	μ=0.254, σ=0.134	μ=1.0, σ=0.0	μ=0.54, σ=0.258
14	KnnRecommender	word2vec_small	euclidean	100	μ=0.252, σ=0.134	μ=1.0, σ=0.0	μ=0.53, σ=0.254
15	KnnRecommender	bert_small	manhattan	10	μ=0.264, σ=0.124	μ=1.0, σ=0.0	μ=0.597, σ=0.259
16	KnnRecommender	bert_small	manhattan	50	μ=0.252, σ=0.139	μ=1.0, σ=0.0	μ=0.536, σ=0.271
17	KnnRecommender	bert_small	manhattan	100	μ=0.252, σ=0.14	μ=1.0, σ=0.0	μ=0.53, σ=0.267
18	KnnRecommender	bert_small	euclidean	10	μ=0.267, σ=0.123	μ=1.0, σ=0.0	μ=0.605, σ=0.259
19	KnnRecommender	bert_small	euclidean	50	μ=0.253, σ=0.14	μ=1.0, σ=0.0	μ=0.538, σ=0.271
20	KnnRecommender	bert_small	euclidean	100	μ=0.252, σ=0.14	μ=1.0, σ=0.0	μ=0.531, σ=0.266
21	KnnRecommender	locfluc_small	manhattan	10	μ=0.271, σ=0.117	μ=1.0, σ=0.0	μ=0.614, σ=0.244
22	KnnRecommender	locfluc_small	manhattan	50	μ=0.26, σ=0.137	μ=1.0, σ=0.0	μ=0.553, σ=0.265
23	KnnRecommender	locfluc_small	manhattan	100	μ=0.261, σ=0.138	μ=1.0, σ=0.0	μ=0.551, σ=0.261
24	KnnRecommender	locfluc_small	euclidean	10	μ=0.269, σ=0.117	μ=1.0, σ=0.0	μ=0.61, σ=0.246
25	KnnRecommender	locfluc_small	euclidean	50	μ=0.26, σ=0.137	μ=1.0, σ=0.0	μ=0.552, σ=0.265
26	KnnRecommender	locfluc_small	euclidean	100	μ=0.261, σ=0.138	μ=1.0, σ=0.0	μ=0.551, σ=0.261
27	KnnRecommender	resnet_small	manhattan	10	μ=0.269, σ=0.118	μ=1.0, σ=0.0	μ=0.605, σ=0.244
28	KnnRecommender	resnet_small	manhattan	50	μ=0.255, σ=0.134	μ=1.0, σ=0.0	μ=0.54, σ=0.26
29	KnnRecommender	resnet_small	manhattan	100	μ=0.252, σ=0.136	μ=1.0, σ=0.0	μ=0.53, σ=0.26
30	KnnRecommender	resnet_small	euclidean	10	μ=0.27, σ=0.118	μ=1.0, σ=0.0	μ=0.609, σ=0.245
31	KnnRecommender	resnet_small	euclidean	50	μ=0.255, σ=0.134	μ=1.0, σ=0.0	μ=0.541, σ=0.261
32	KnnRecommender	resnet_small	euclidean	100	μ=0.252, σ=0.136	μ=1.0, σ=0.0	μ=0.529, σ=0.26
33	KnnRecommender	spectral_small	manhattan	10	μ=0.27, σ=0.121	μ=1.0, σ=0.0	μ=0.612, σ=0.252
34	KnnRecommender	spectral_small	manhattan	50	μ=0.255, σ=0.135	μ=1.0, σ=0.0	μ=0.542, σ=0.261
35	KnnRecommender	spectral_small	manhattan	100	μ=0.253, σ=0.136	μ=1.0, σ=0.0	μ=0.532, σ=0.256
36	KnnRecommender	spectral_small	euclidean	10	μ=0.27, σ=0.12	μ=1.0, σ=0.0	μ=0.61, σ=0.251
37	KnnRecommender	spectral_small	euclidean	50	μ=0.256, σ=0.135	μ=1.0, σ=0.0	μ=0.544, σ=0.261
38	KnnRecommender	spectral_small	euclidean	100	μ=0.254, σ=0.136	μ=1.0, σ=0.0	μ=0.533, σ=0.255
39	KnnRecommender	mfcc_small	manhattan	10	μ=0.27, σ=0.118	μ=1.0, σ=0.0	μ=0.61, σ=0.245
40	KnnRecommender	mfcc_small	manhattan	50	μ=0.257, σ=0.135	μ=1.0, σ=0.0	μ=0.545, σ=0.261
41	KnnRecommender	mfcc_small	manhattan	100	μ=0.255, σ=0.136	μ=1.0, σ=0.0	μ=0.536, σ=0.257
42	KnnRecommender	mfcc_small	euclidean	10	μ=0.268, σ=0.119	μ=1.0, σ=0.0	μ=0.609, σ=0.248
43	KnnRecommender	mfcc_small	euclidean	50	μ=0.257, σ=0.135	μ=1.0, σ=0.0	μ=0.546, σ=0.261