A MACHINE LEARNING APPROACH FOR CONTENT-BASED MUSIC RECOMMENDER SYSTEMS

Alan Francisco Höng

February 19, 2016

Bachelor's Thesis presentation, UFRGS

Struture of Content

This presentation is structured accordingly to the work it presents:

- Motivation and goals
- Background on music recommendations and similarity
- The Approach
  - Data Acquisition
  - Feature Extraction
  - Clustering
  - Recommender
- Evaluation
- Conclusions

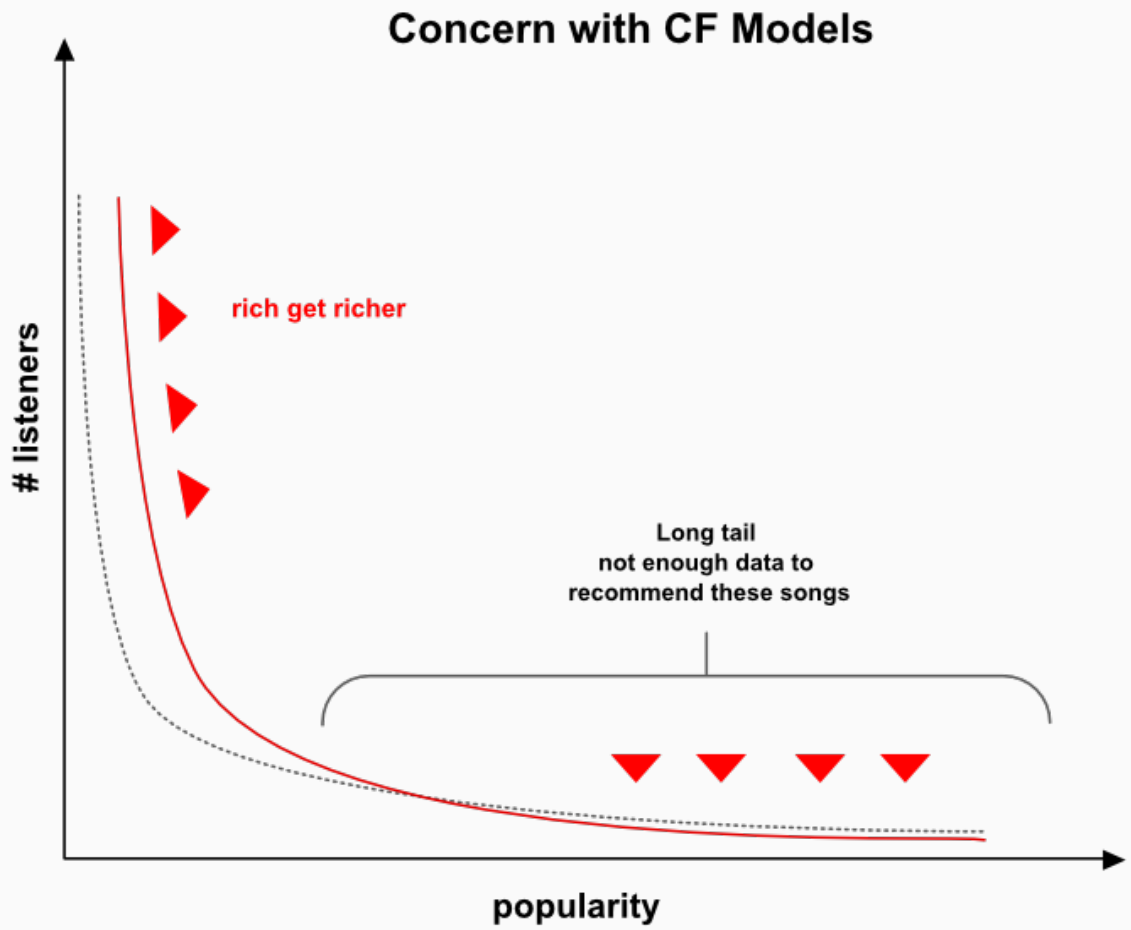
MOTIVATION AND GOALS

Motivation

Motivations

- Traditional collaborative filtering approaches fail to recommend lesser known artists
- Rapidly growing online available music databases
- Little effort was made yet to use music psychology research to choose features from audio

Motivation


Concern with CF Models
rich get richer
listeners
Long tail
not enough data to
recommend these songs
popularity

Goals

Goals

- Prototype recommender system based on audio analysis
- Use as less user data as possible
- Be able to recommend each track in database
- Introduce a similarity metric based on audio content
- Evaluate this system

BACKGROUND ON MUSIC RECOMMENDATIONS AND SIMILARITY

Collaborative Filtering

Likematrix used for Collaborative Filtering contains all known likes in a system. Items are represented as column vectors and users as row vectors.

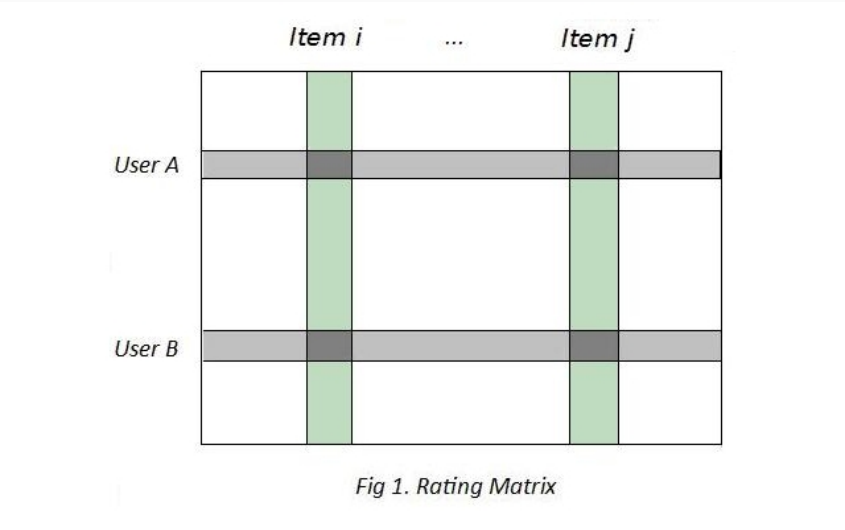


Fig 1. Rating Matrix

Collaborative Filtering

The Algorithm

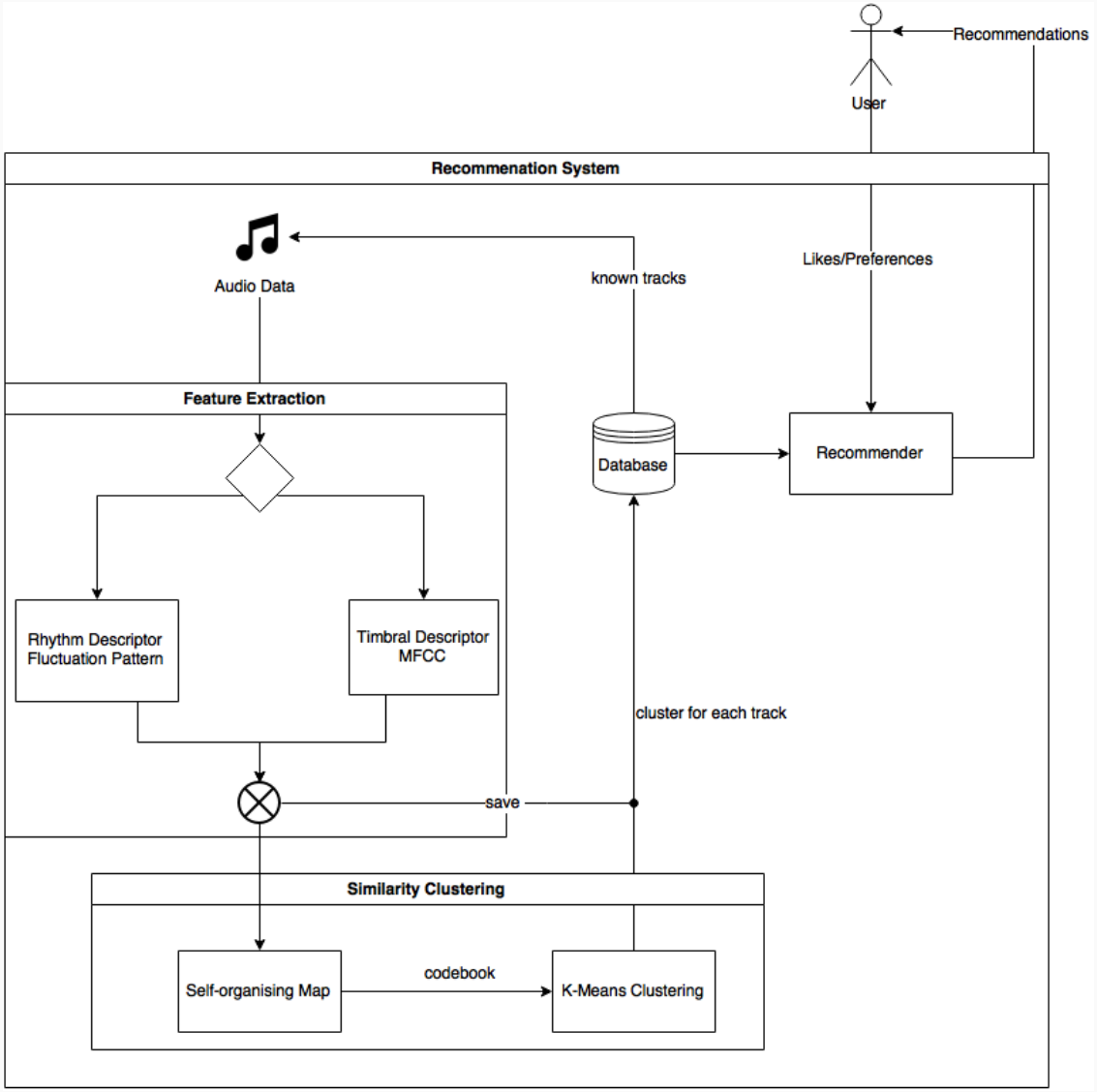
Select subset S of similar tracks using for example cosine similarity $s : T \times T \to \mathbb{R}$

Compute average weighted by similarity to infer new prediction of user preferences

$$p_{u,i} = \frac{\sum_{t \in S} s(t, t') l_{u,t'}}{\sum_{t \in S} |s(t, t')|} \tag{1}$$

THE APPROACH

Overview

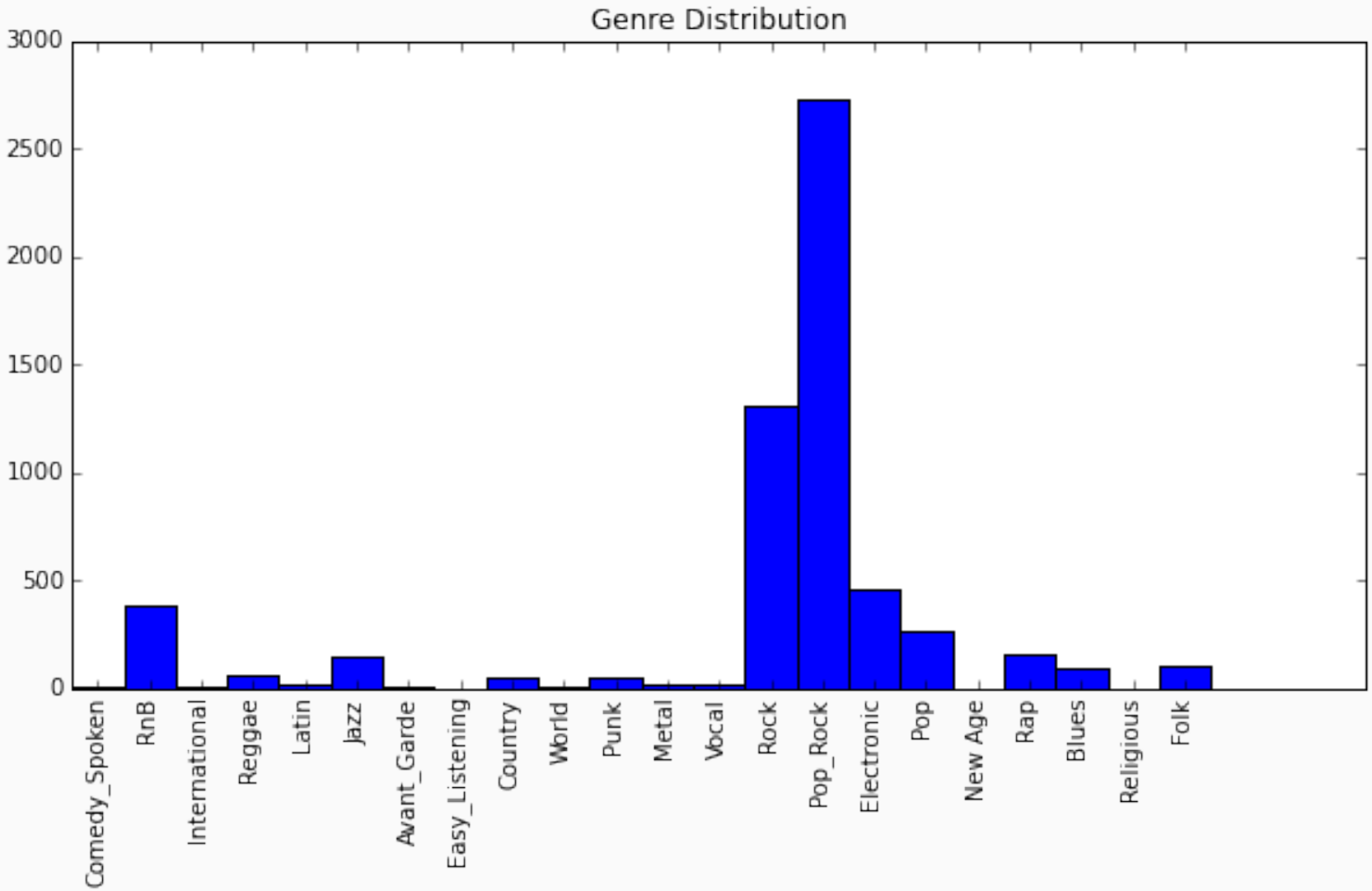

Recommendations
User
Recommenation System
Audio Data
known tracks
Likes/Preferences
Feature Extraction
Database
Recommender
Rhythm Descriptor
Fluctuation Pattern
Timbral Descriptor
MFCC
cluster for each track
save
Similarity Clustering
Self-organising Map
codebook
K-Means Clustering

Data Acquisition

Constructing the dataset

- Metadata is provided from **Million Song Dataset**
- Userdata is provided by **ThisIsMyJam** Archive dump
- Audio is fetched from music download service **7digital**
- **250** Users with **13891** Likes over **8003** tracks

Genre Distribution



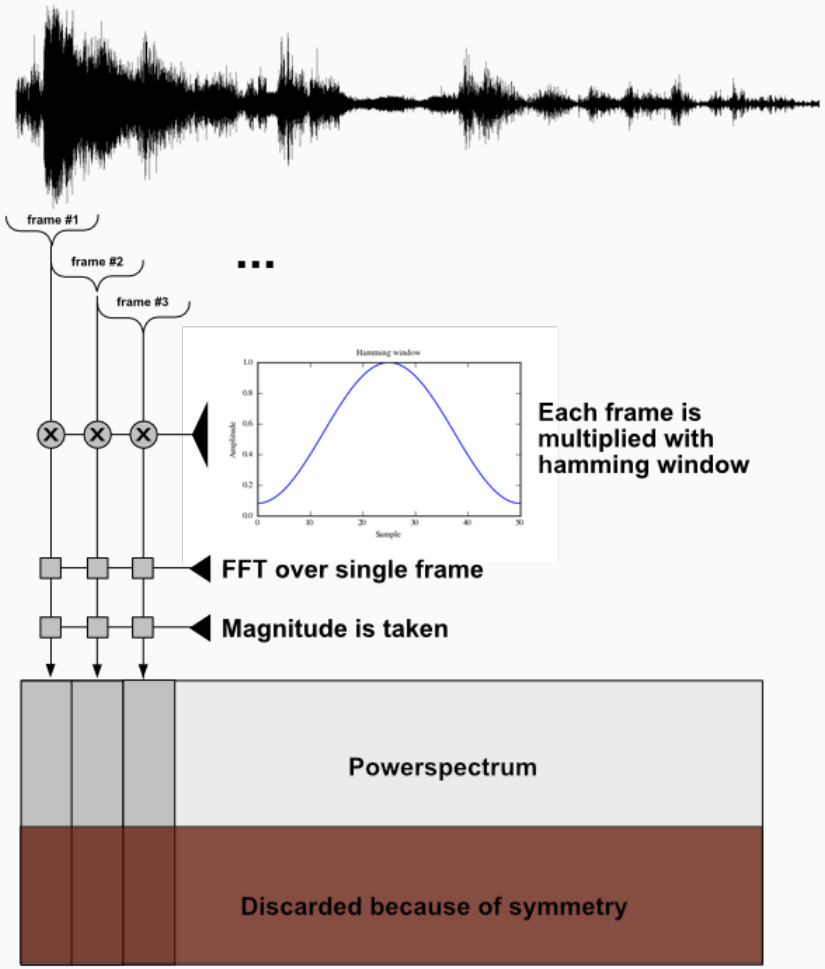
Feature Extraction

Music Descriptors

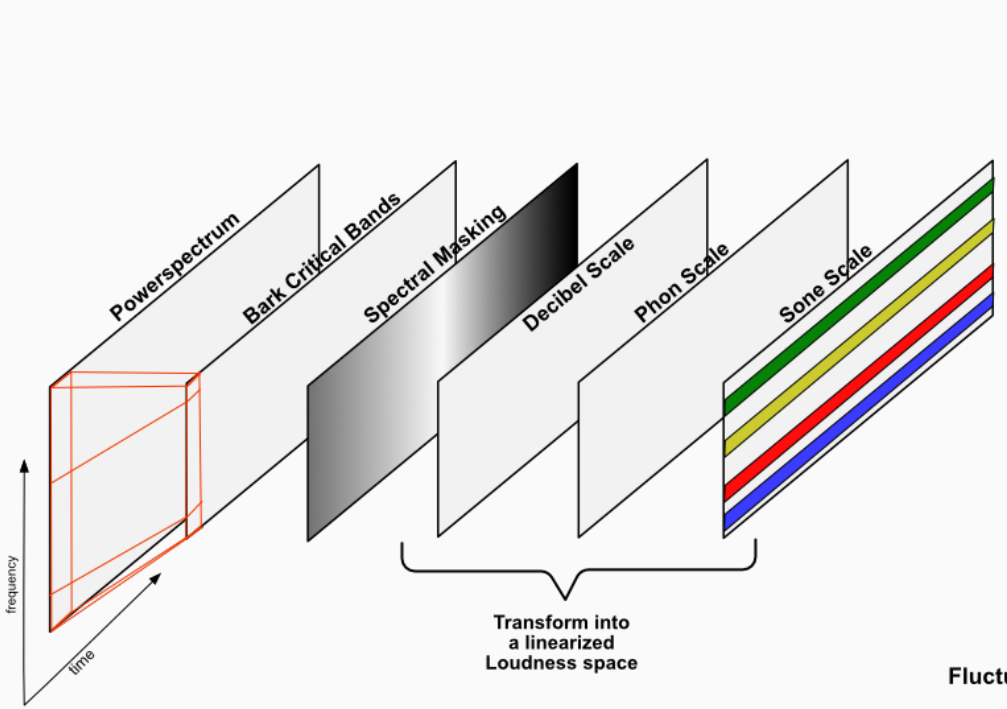
Two powerspectrum-based descriptors are extracted from the raw audio data. One for **rhythm** description and the other one for **timbre** description.

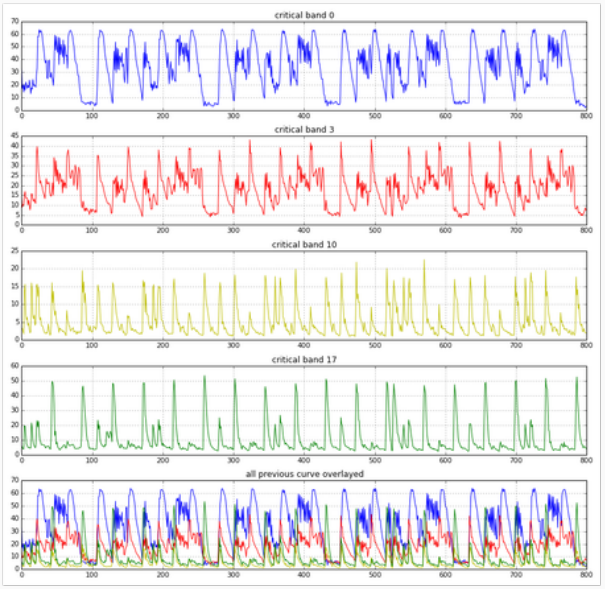
1. Fluctuation Pattern
2. Mel Frequency Cepstral Coeffcients

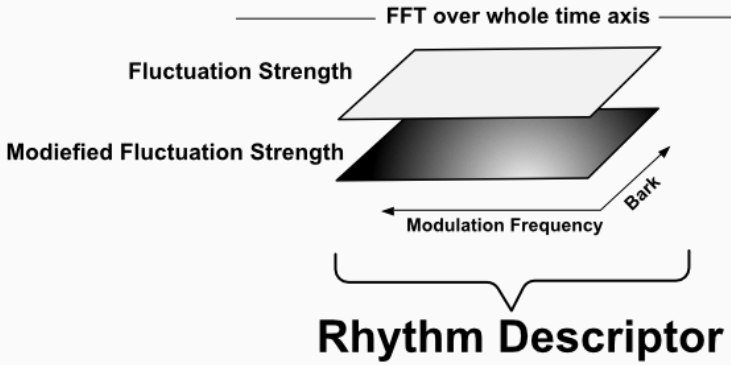
Powerspectrum



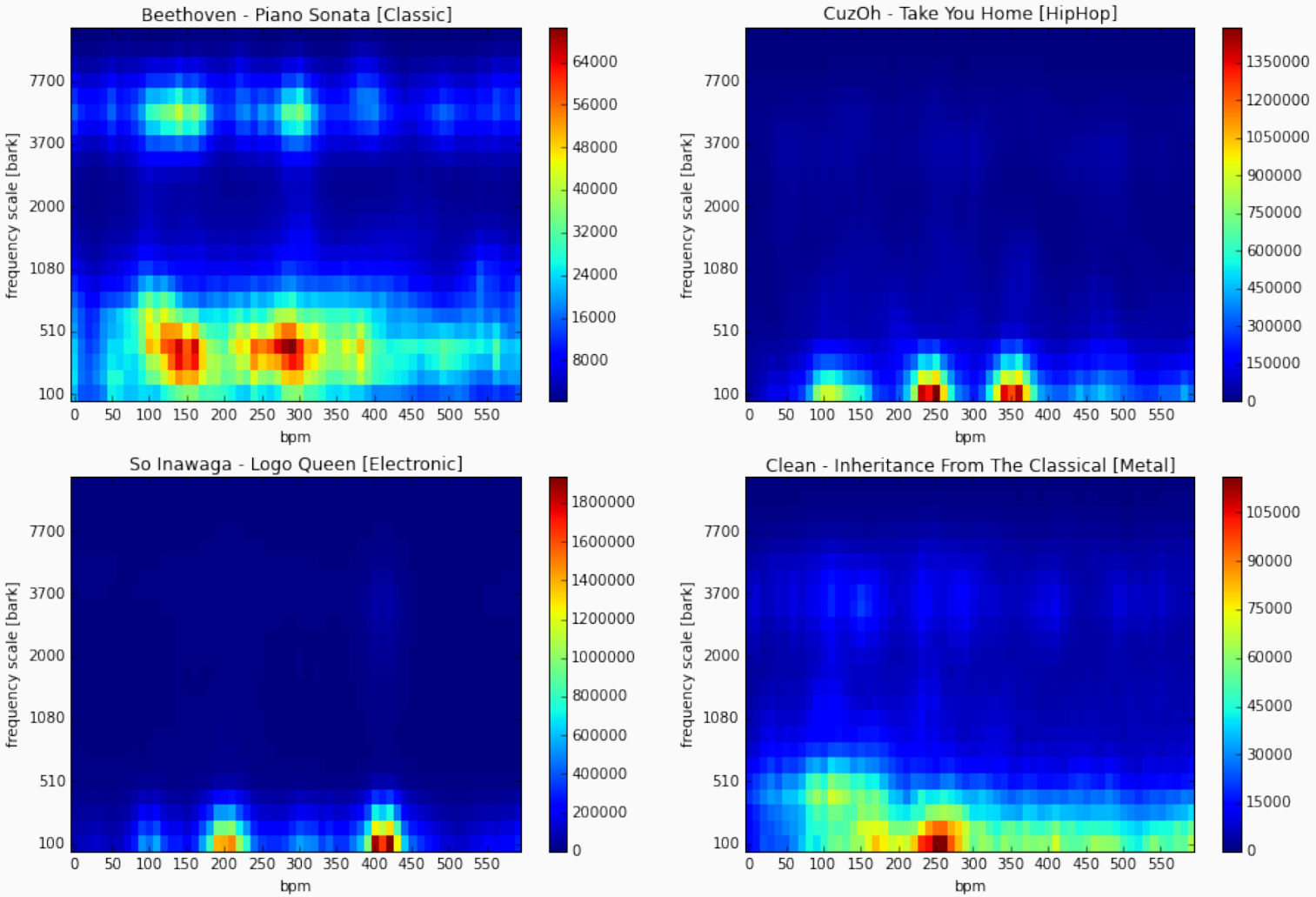
Fluctuation Pattern



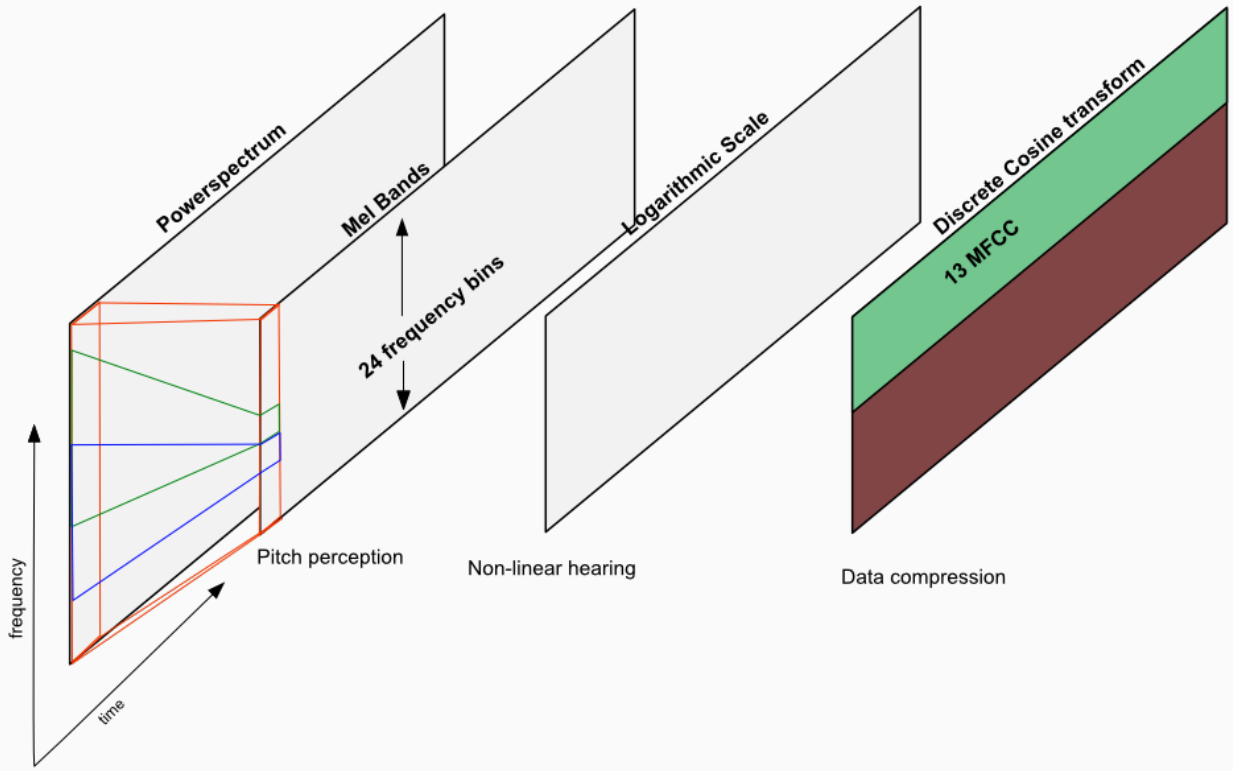




Fluctuation Pattern Examples



Mel Frequency Cepstral Coefficients



Feature Combinations

Table: Feature Vectors

Abb.	Combination	Dimensionality
A	Fluctuation Patterns Compressed + Gaussian Representation of Concatenated MFCC	90
B	Fluctuation Patterns Compressed + Concatenated MFCC Median	100
C	Fluctuation Patterns Compressed + Concatenated MFCC Delta Median	100
D	Fluctuation Patterns Compressed +(Concatenated MFCC + Concatenated MFCC Delta) Median	140
E	Fluctuation Patterns Compressed	60
F	Concatenated MFCC Medians	40
G	Concatenated MFCC Delta Median	40
H	Concatenated MFCC + Concatenated MFCC Delta Median	80
I	Gaussian Representation of Concatenated MFCC	30
J	Fluctuation Patterns Raw	1440

Clustering: Self-organizing map algorithm

Kohonen's self organizing map

- Uses euclidean distance
- Performs unsupervised learning
- Robust against clusters of distinct sizes
- Provides good visualization of high dimensional data

Algorithm

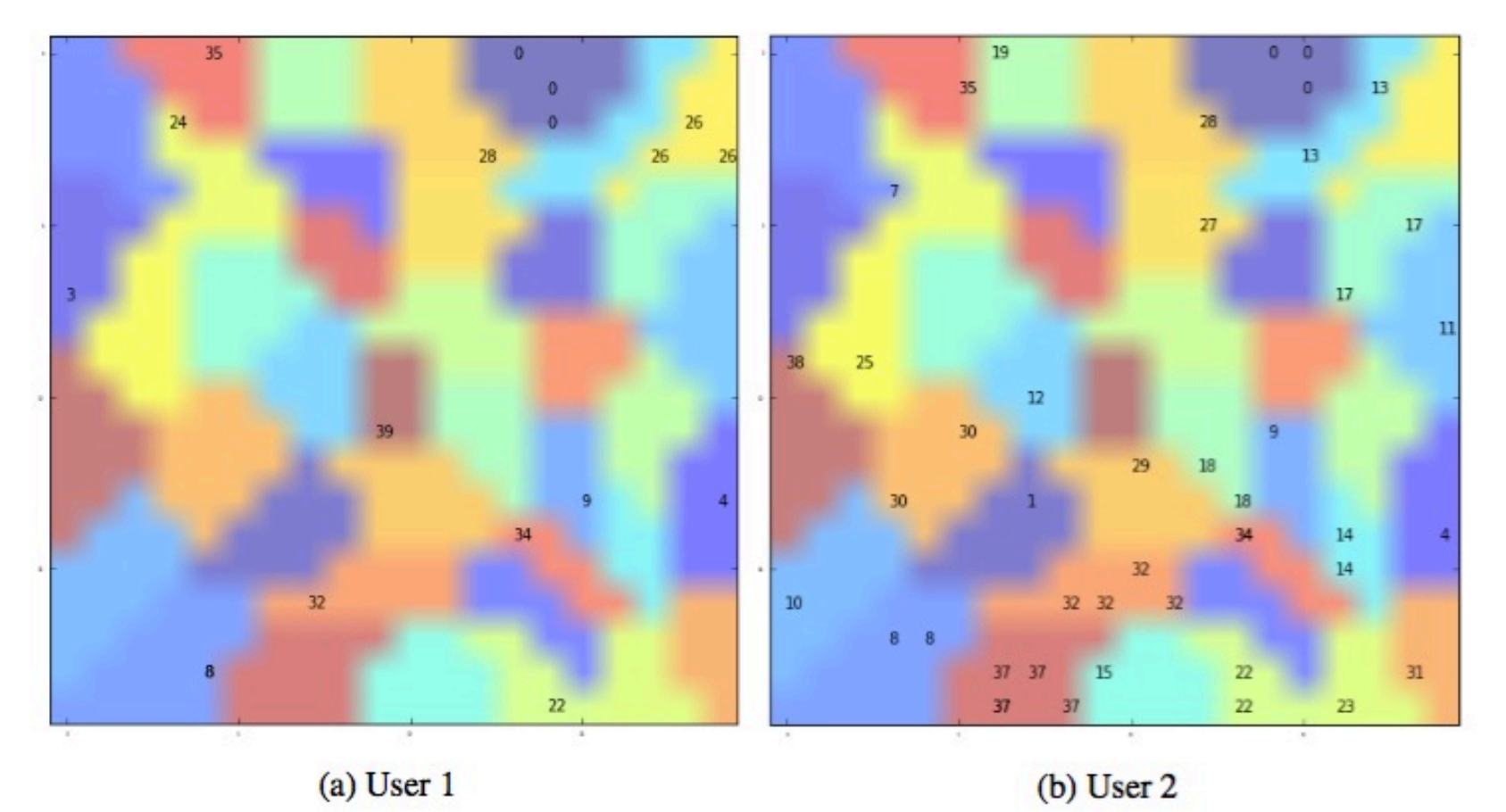
1. Rectangular grid of interconnected neurons, each one associated to a weight vector
2. Each datapoint can be projected to a best matching unit (neuron). In terms of minimal distance to it's weight vector.
3. The neurons weight vector and it's neighboring neurons vectors are adjusted to fit the input data.
4. Repeated until model fits the data sufficiently good.

Clustering

Clustering Parameters

- Tracks are clustered using a 20 by 20 SOM
- the resulting 400 weight vectors are further clustered using K-Means.
- this results in 40 clusters of similar music tracks.

Clustering



(a) User 1

(b) User 2

Recommender

Each user has a list of preferences known to the system

Selecting tracks

1. For a given user his preferences are mapped into the corresponding clusters. Resulting in a List of clusters.
2. Then this list of clusters is ordered by frequency of cluster's appearing the previous step.
3. Tracks are retrieved corresponding to above ordering

Recommender

As one cluster usually holds a big amount of tracks a score is introduced.

Score calculation

1. Select minimum distance to a liked tracks in a given cluster:

$$s'_u(\boldsymbol{t}) = \min(\{\|\boldsymbol{l}_i - \boldsymbol{t}\|_2 \mid l_i \in (C_t \cap L_u)\})$$

2. Invert this distance:

$$s_u(\boldsymbol{t}) = \frac{1}{s'_u(\boldsymbol{t})}$$

EVALUATION

Evaluation

Evaluation procedure

- To evaluate we need a truth to proof the algorithm against.
- In offline evaluation this truth can only be estimated
- To evaluate we hide 10% of each user's likes
- This hidden likes have to be guessed by the recommender.

Evaluation

- **Sign Test:** The sign test gives the probability that algorithm A is **not** truly better than algorithm B. It is counted how often Algorithm A outperforms algorithm B n_A and vice versa n_B, while $n = n_A + n_B$.

Table: Probabilities between our approach and random recommender

N	A	B	C	D	E	F	G	H	I	J
50	0.83	0.32	0.71	0.64	0.25	0.47	0.57	**0.07**	0.75	0.45
250	0.34	0.44	0.57	0.30	0.15	0.54	0.51	**0.02**	0.38	0.31
1000	0.41	0.18	0.23	**0.09**	**0.08**	0.41	0.59	**0.08**	0.27	0.13
2000	**0.06**	0.17	0.22	0.11	0.20	0.47	0.49	**0.09**	0.33	**0.04**

Evaluation

Table: Outcomes for unary ratings

	Recommended	Not Recommended
Used	True Positive	False Negative
Not Used	False Positive	True Negative

Evaluation

Usage Prediction Evaluation metrics based on the **4** possible **outcomes** for recommendations.

$$\text{Precision} = \frac{|tp|}{|tp| + |fp|} \tag{2}$$

$$\text{Recall(True Positive Rate)} = \frac{|tp|}{|tp| + |fn|} \tag{3}$$

$$\text{False Positive Rate} = \frac{|fp|}{|fp| + |tn|} \tag{4}$$

Evaluation

Table: Feature evaluation table

	Clusters used	Our Approach	CF Recommender	Random Recommender	Recommendation Size
A	1	0.87%(±4.37)	0.84%(±4.34)	0.47%(±3.62)	50
	5	3.86%(±9.36)	3.21%(±8.88)	3.28%(±8.93)	250
	20	13.89%(±17.77)	9.73%(±13.62)	12.02%(±13.96)	1000
	40	27.49%(±21.36)	18.26%(±19.34)	25.40%(±20.99)	2000
D	1	**1.20%(±5.82)**	0.84%(±4.34)	0.58%(±4.12)	50
	5	**4.16%(±11.04)**	3.21%(±8.88)	3.06%(±7.99)	250
	20	**17.35%(±19.28)**	9.73%(±13.62)	11.38%(±15.10)	1000
	40	**30.58%(±23.28)**	18.26%(±19.34)	24.78%(±21.42)	2000
E	1	1.12%(±4.89)	0.84%(±4.34)	0.60%(±3.11)	50
	5	3.80%(±9.17)	3.21%(±8.88)	2.93%(±9.00)	250
	20	14.36%(±16.25)	9.73%(±13.62)	12.02%(±14.59)	1000
	40	27.45%(±21.77)	18.26%(±19.34)	26.85%(±22.45)	2000
H	1	0.86%(±4.73)	0.84%(±4.34)	1.52%(±6.75)	50
	5	3.61%(±8.82)	3.21%(±8.88)	2.89%(±8.17)	250
	20	15.23%(±18.47)	9.73%(±13.62)	11.89%(±15.13)	1000
	40	29.74%(±22.68)	18.26%(±19.34)	24.77%(±21.52)	2000

Evaluation

Table: Usage Predictions

Feature Vector D			
N	Precision	Recall	FPR
50	0.104%(±0.479%)	1.204%(±5.823%)	0.625%(±0.003%)
250	0.091%(±0.196%)	4.163%(±11.044%)	3.123%(±0.006%)
1000	0.094%(±0.103%)	17.349%(±19.277%)	12.493%(±0.011%)
2000	0.084%(±0.071%)	30.575%(±23.279%)	24.988%(±0.014%)

Evaluation

Observations

- Difficult to evaluate and compare results as their doesn't exist a common used dataset yet.
- Offline evaluation is not an optimal method as it makes some risky assumptions about false positives
- Superior performance to the CF algorithm, due to sparse user data
- Random algorithm could only be outperformed at high recommendation lengths

Listening Evaluation

CONCLUSIONS

Conclusions

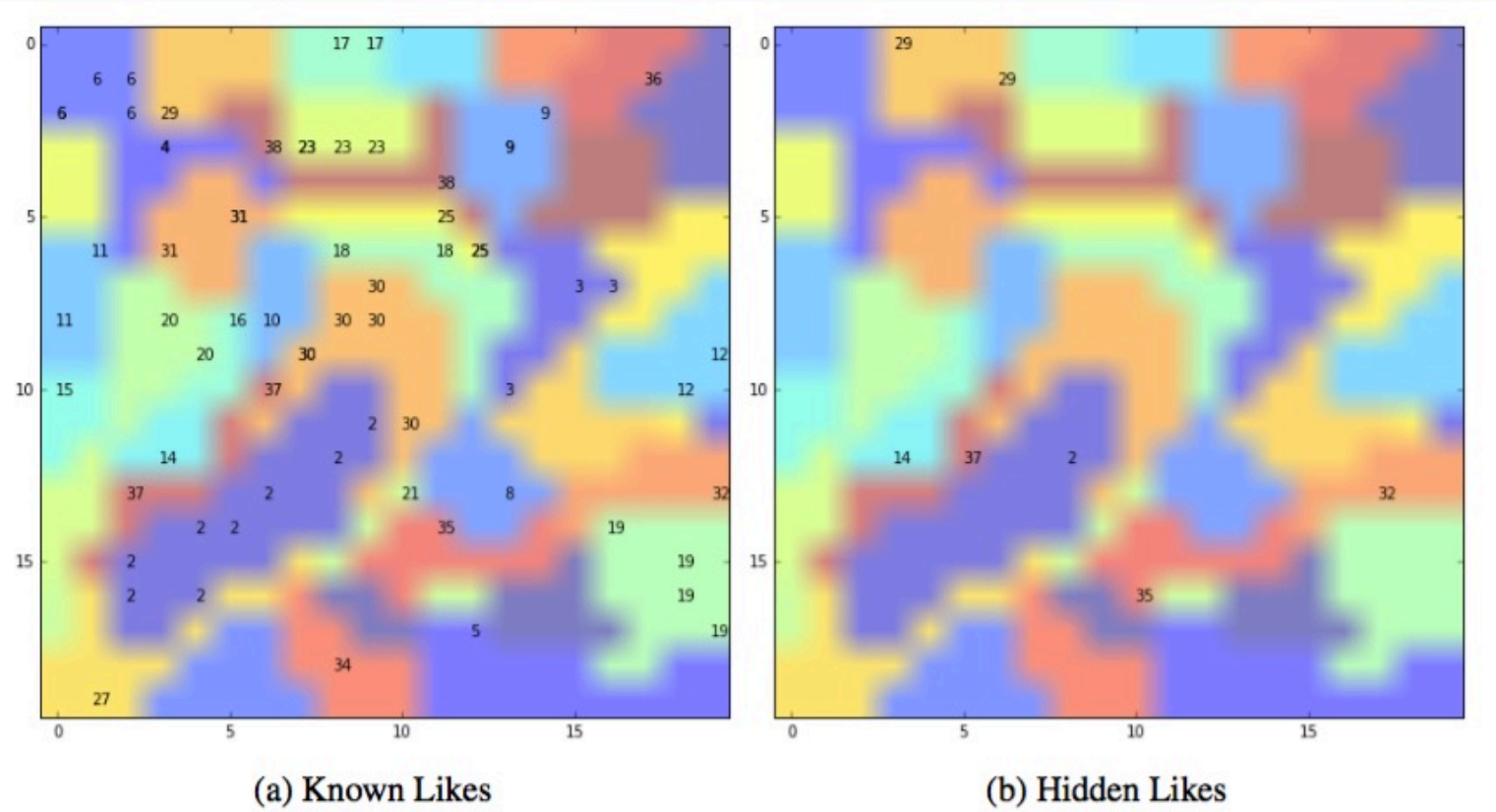
Summary

- An music similarity measure based on audio content could be build.
- An recommender system based on this similarity measure was prototyped.
- Human listening preferences don't always correlate with objective music similarity.
- SOM clustering might not be the best clustering method

Conclusions

The random split problem:

As user likes get split completly randomly it is hard for this recommender to guess isolated likes:



(a) Known Likes

(b) Hidden Likes

Conclusions

Future Work

- Consider subjective music similarity by learning a similarity function instead of using euclidean distances.
- Explore advanced clustering methods.
- Evaluate online with real users.

Questions?