

# Hiërarchische zelflerende modellen voor muziekbeveiling

Erik Vandeputte

Promotor: prof. dr. ir. Benjamin Schrauwen  
Begeleiders: Sander Dieleman, Philémon Brakel

Masterproef ingediend tot het behalen van de academische graad van  
Master in de ingenieurswetenschappen: computerwetenschappen

Vakgroep Elektronica en Informatiesystemen  
Voorzitter: prof. dr. ir. Jan Van Campenhout  
Faculteit Ingenieurswetenschappen en Architectuur  
Academiejaar 2012-2013





# Hiërarchische zelflerende modellen voor muziekbeveiling

Erik Vandeputte

Promotor: prof. dr. ir. Benjamin Schrauwen  
Begeleiders: Sander Dieleman, Philémon Brakel

Masterproef ingediend tot het behalen van de academische graad van  
Master in de ingenieurswetenschappen: computerwetenschappen

Vakgroep Elektronica en Informatiesystemen  
Voorzitter: prof. dr. ir. Jan Van Campenhout  
Faculteit Ingenieurswetenschappen en Architectuur  
Academiejaar 2012-2013



# Toelating tot bruikleen

De auteur geeft de toelating deze masterproef voor consultatie beschikbaar te stellen en delen van de masterproef te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze masterproef.

Erik Vandeputte, mei 2013

# Voorwoord

Muziek en computers zijn altijd 2 grote passies van mij geweest. Vandaar dat ik tijdens het beschikbaar komen van de lijst met thesisonderwerpen begonnen ben met zoeken op het trefwoord 'muziek'. Ongeveer een jaar later kan ik zeggen dat de combinatie van computerwetenschappen en muziek voor mij een leuke en verrassende uitdaging is gebleken.

Ik wil eerst en vooral mijn promotor prof. dr. ir. Benjamin Schrauwen bedanken. Daarnaast wil ik in het bijzonder ook mijn begeleiders Philemon Brakel, Sander Dieleman, Pieter-Jan Kindermans en Aäron van den Oord bedanken voor hun advies en feedback.

Ten slotte wil ik ook mijn ouders bedanken voor hun steun en vertrouwen.

# Music recommendation based on listening history and acoustic models

Erik Vandeputte

Supervisor(s): prof. dr. ir. Benjamin Schrauwen, ir. Sander Dieleman, ir. Philémon Brakel

**Abstract**—In this article, we try to come up with a successful solution to the cold start problem by using a recommender system that combines the benefits of collaborative filtering techniques and content-based techniques.

**Keywords**—Music Recommendation, Collaborative Filtering, Cold start problem

## I. INTRODUCTION

SHIFTING from physical sales to digital sales during recent years, the online music industry is booming. Users are virtually overwhelmed with the available music content. Both from the perspective of the users, who want to discover relevant new music, as from the perspective of the content providers, who want to match their content with the right consumer group, the interest in recommendation engines has risen. Collaborative Filtering techniques (CF) are based on analyzing the large amount of information on user behavior such as rating information or listening/viewing history. Based on this information they are able to learn new relations between users and items so that they are able to predict what user will like based on their similarity to other users or items they've liked. CF is known to suffer from the cold start problem, where a system cannot recommend items for which it doesn't have any ratings for. Content-based techniques attempt to overcome this problem by extracting features out of the actual content and matching them with known user preferences for those features. We attempt to build a hybrid music recommender system that preforms well on song for which we have a lot of information for, but is able to recommend new relevant songs to users also.

### A. Subset

The dataset consists of a subset of the Million Song Dataset containing listening information for 20.000 users and 10.000 songs. This means that the input for the recommender system is a form of implicit feedback. Additionally, there is an 30s audio file available that will be used for extracting features out of the audio. We can represent this data in a user-item matrix  $R$ . One dimension corresponds to a user  $u$  while the other dimension corresponds to the song  $i$ . The entry  $r_{ui}$  represents the number of times user  $u$  has listened to song  $i$ .

## II. COLLABORATIVE FILTERING

Our CF approach is based on a latent factor model that takes implicit feedback into account. Latent factor models try to discover latent features that explain the observed ratings. Our model tries to decompose a user-item matrix  $R$  into a product of a user-factor matrix  $X$  with a song-factor matrix  $Y$ . If the number of latent features is  $k$  then the observed score  $r_{ui}$  in the

user-item matrix is approximated by an innerproduct between a user-factor vector  $x_u \in \mathbb{R}^k$  and a song-factor vector  $y_i \in \mathbb{R}^k$  i.e.,  $r_{ui} \approx x_u^T y_i$ . The main difference between recommender systems based on explicit feedback datasets and those based on implicit feedback datasets is that a score of 0 in an implicit feedback dataset doesn't mean that the user doesn't like the item. In our case it just means that he hasn't listened to that song before. A suitable loss function for a recommender system for implicit feedback datasets can look like this:

$$\min_{x^*, y^*} = \sum_{u, i} c_{ui} (p_{ui} - x_u^T y_i)^2 + \lambda \left( \sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right) \quad (1)$$

$c_{ui}$  is a confidence parameter for a certain rating that is 1 when user  $u$  hasn't listened to item  $i$  but that increases lineary or logarithmically with the number of play counts.  $p_{ui}$  is a binarized form of the user-item matrix where  $p_{ui} = 1$  means that user  $u$  has listened to item  $i$ .  $\lambda$  is used for regularizing the model. Because the user-item matrix cannot be considered sparse, Alternating Least Squares (ALS) optimization is used for equation 1. We alternate between re-computing the user-factors and the song-factors by considering the former as fixed and solve for the latter. In each step we are guaranteed to lower the value for the cost function. [1] describes an efficient implementation of this process, scaling lineary in the size of the dataset.

### A. Results

The results were compared to two trivial recommendation algorithms and one neighborhood-based model. A first baseline system, called *popularity*, recommends each song in descending order of its popularity, regardless of the user's preferences. The *same artist* algorithm recommends songs in descending order of popularity as well, taking into account only the artists that the user has played before. The neighborhood-based model is based on [2].

The evaluation metrics used are the mean average precision (mAP) and the recall when 50 recommendations are made.

As expected, the neighborhood-based model and the latent factor model outperform the trivial recommender systems. An important advantage of the latent factor model is that we obtain a representation of each song in  $k$  factors.

## III. CONTENT-BASED FILTERING

Motivated by the idea that some of the song factors could correspond to various acoustical attributes of the song, an alternative approach is for songs for which we don't have any listening

Recommender system	mAP@50	Rec@50
<i>Popularity</i>	0.01295	0.02739
<i>Same artist</i>	0.03063	0.04723
<i>Neighborhood-based model</i>	0.12405	0.25469
<i>Latent factor model</i>	0.12830	0.27362

TABLE I

THE EVALUATION FOR DIFFERENT RECOMMENDER SYSTEMS EVALUATED ON THE SUBSET OF THE MSD

information to predict the song factors. Predicting the song factors directly out of the raw audio files is computationally expensive. We present an alternative approach based on the clustering of Mel Frequency Cepstral Coefficients.

#### A. Feature Learning

MFCC's [3] originated in the context of speech recognition, but have already successfully been used in the field of music data mining, f.e. in genre recognition. MFCC's present a compact representation of how humans perceive sound. Each raw audio file was converted to a sequence of 2905 MFCC-vectors. As input for a feature learning algorithm, a sequence of 5 MFCC-vectors was grouped into one frame. Motivated by [4], k-means clustering was chosen as feature learning algorithm, operating directly on frames. K-means is extremely fast, has no hyperparameters to tune other than the number of clusters and is very easy to implement. After learning a number of cluster means, we consider two choices for the feature mapping function  $f$ . The standard formula for cluster assignment is *hard clustering*:

$$f_k(x) = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \|c^{(j)} - x\|_2^2 \\ 0 & \text{else} \end{cases} \quad (2)$$

An alternative approach is to assign each frame to multiple cluster means by formula:

$$f_k(x) = \max \{0, \mu(z) - z_k\} \quad (3)$$

The latter we will refer to as *soft clustering*.

#### B. Predicting Song Factors

After performing the feature learning algorithm, we can now represent a song by its clustered MFCC-representation. To be able to predict the song-factors out of this representation, we make use of regression analysis. First we splitted the songfactormatrix into a 80/20 training- and testset. By using ridge regression we estimated the song factors, based on the training data. The hyperparameter  $\lambda$  is obtained through 10-fold cross-validation. The evaluation on the testset for different frame sizes, different number of features and for *hard* and *soft clustering* is shown in figure 1.

### IV. MAKING RECOMMENDATIONS

The goal is to estimate the predicted song factors for new songs. To test whether this content-based approach is feasible, different experiments were conducted.

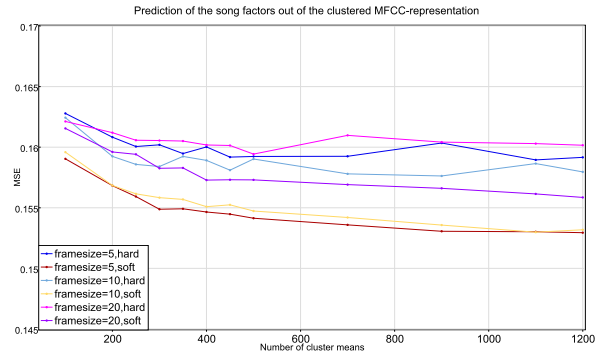


Fig. 1. Evolution of the number of cluster means on the MSE between the predicted and the true song factors for different types of clustering and frame sizes.

### V. EXPERIMENTS

Starting from the original user-item matrix, the dataset is divided into two parts as described in figure 2. A latent factor model is trained on  $X_{train}$ , resulting in a songfactormatrix  $V_{train}$  and a userfactormatrix  $U$ . The songfactormatrix  $V_{train}$  is now used to train a linear model that estimates songfactors based on the clustered MFCC-representation described in III-A. This model is now used to predict the songfactors for the songs in  $X_{test}$ . We will refer to those songfactors as  $V_{test\_audio}$ .

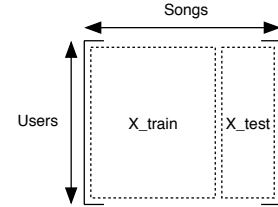


Fig. 2. Initial splitting of the dataset.

To simulate an actual cold start problem, we now remove listening information for the songs in  $X_{test}$ . Concretely, we split the matrix  $X_{test}$  randomly into 2 disjunct collections called  $X_{test\_A}$ , that will serve as our new testset, and  $X_{test\_B}$ , that will be used to calculate a new songfactormatrix  $V_{test\_mf}$ , using the userfactormatrix  $U$  and one iteration out of the latent factor model described in III where we take the userfactors  $U$  to be constant. The parameter  $\alpha$  is used to determine the percentage listening information that is removed and is thus in  $X_{test\_A}$ . The matrix  $X_{test}$  consisted out of 264.767 nonzero entries. When  $\alpha = 0.5$ , both matrices  $X_{test\_A}$  and  $X_{test\_B}$  contain about 132.000 nonzero entries.

For different values of  $\alpha$ , we can generate our prediction scores for each song for the two types of recommender systems as follows:

$$pred\_audio = U * (V_{test\_audio})^T \quad (4)$$

$$pred\_mf = U * (V_{test\_mf})^T \quad (5)$$

Furthermore, the content-based technique (4) and the social-based technique (5) could easily be combined into a hybrid recommender system:

$$pred\_hybrid = pred\_audio + pred\_mf \quad (6)$$

The results of this experiment for multiple values of  $\alpha$  are presented in tables II and III.

TABLE II  
AUC FOR DIFFERENT RECOMMENDER SYSTEMS AND DIFFERENT VALUES OF  $\alpha$

	$\alpha$						
	0.5	0.8	0.9	0.95	0.99	0.995	0.999
pred_mf	0.8580	0.8354	0.8129	0.7844	0.7022	0.6669	0.6451
pred_audio	0.6630	0.6629	0.6628	0.6625	0.6626	0.6625	<b>0.6580</b>
pred_hybrid	0.8520	0.8259	0.7987	0.7668	0.7019	<b>0.6877</b>	0.6808

TABLE III  
MAP@500 FOR DIFFERENT RECOMMENDER SYSTEMS AND DIFFERENT VALUES OF  $\alpha$

	$\alpha$						
	0.5	0.8	0.9	0.95	0.99	0.995	0.999
pred_mf	0.1367	0.1446	0.1353	0.1174	0.0572	0.0368	0.0228
pred_audio	0.0174	0.0203	0.0214	0.0219	0.0224	0.0224	0.0226
pred_hybrid	0.1278	0.1292	0.1091	0.0792	0.0365	0.0302	<b>0.0277</b>

We can tune the importance of each technique in the hybrid approach by using an extra parameter  $\beta$ . Equation 6 then becomes:

$$pred\_hybrid = (1-\beta)*pred\_audio + (\beta)*pred\_mf \quad \beta \in [0, 1] \quad (7)$$

We now can search for the optimal value of  $\beta$  for different values of  $\alpha$ :

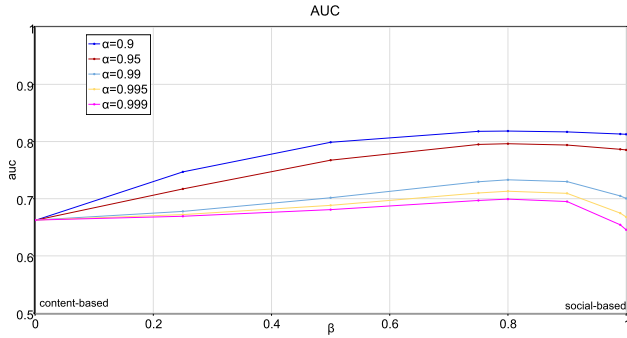


Fig. 3. Evolution of the number of cluster means on the MSE between the predicted and the true song factors for different types of clustering and frame sizes.

We can clearly see that for different values of  $\alpha$ , the optimal value for  $\beta$  lies between 0.8-0.9, but then decreases when we only take the social-based system into account. We can conclude that in the tuned hybrid recommender system the social-based technique has a significant bigger contribution than our own content-based technique, but that the correct combination of both can lead to an improved recommender system in the case of a cold start.

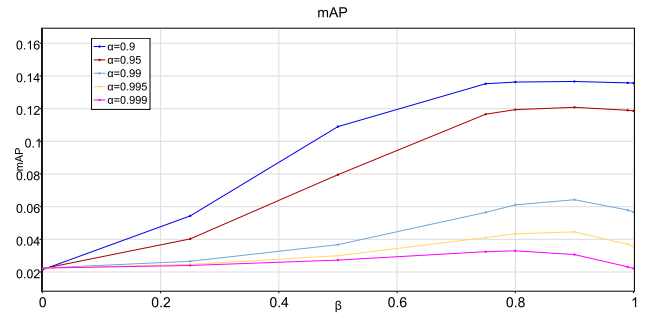


Fig. 4. Evolution of the number of cluster means on the MSE between the predicted and the true song factors for different types of clustering and frame sizes.

## VI. CONCLUSION & FUTURE WORK

In this paper, a solution to the cold start problem in music recommendation is proposed by using a latent factor model that allows to predict songfactors based on MFCC-vectors rather than calculating the factors out of the listening history. If no listening information is available for certain songs, this approach works better than randomly recommend those songs, but the latent factor model tends to be superior even when few listening information is available. The main reason why the prediction of the songfactors is poor is that not all song factors can be predicted out of MFCC's because they can correspond to other non-acoustical features. A possible solution to this problem is to incorporate other, non-acoustical attributes while predicting song factors.

Furthermore, a hybrid recommendation system that correctly combines the social-based and content-based technique outperforms the latent factor model when few listening information is available.

## REFERENCES

- [1] Yifan Hu, Yehuda Koren, and Chris Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, Washington, DC, USA, 2008, ICDM '08, pp. 263–272, IEEE Computer Society.
- [2] Fabio Aiolli, "A preliminary study on a recommender system for the million songs dataset challenge," 2012.
- [3] S. Davis and P. Mermelstein, "Experiments in syllable-based recognition of continuous speech," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 28, pp. 357 – 366, Aug. 1980.
- [4] Adam Coates, Andrew Y. Ng, and Honglak Lee, "An analysis of single-layer networks in unsupervised feature learning.," *Journal of Machine Learning Research - Proceedings Track*, vol. 15, pp. 215–223, 2011.



# Inhoudsopgave

<b>Toelating tot bruikleen</b>	<b>i</b>
<b>Voorwoord</b>	<b>ii</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>x</b>
<b>Lijst met afkortingen</b>	<b>xi</b>
<b>1 Inleiding</b>	<b>1</b>
1.1 Wat zijn aanbevelingssystemen? . . . . .	1
1.1.1 Verschillende soorten aanbevelingssystemen . . . . .	2
1.1.2 De lange staart . . . . .	4
1.1.3 Evaluatie van een aanbevelingssysteem . . . . .	4
1.1.3.1 mean Average Precision . . . . .	5
1.1.3.2 Novelty . . . . .	7
1.1.3.3 Recall . . . . .	7
1.1.3.4 Area Under Curve . . . . .	8
<b>2 Collaboratieve Filtertechnieken</b>	<b>10</b>
2.1 Buurgebaseerde modellen . . . . .	10
2.1.1 Algoritme . . . . .	11
2.2 Verborgene Factormodellen . . . . .	11
2.2.1 Algoritme . . . . .	13
2.3 Resultaten . . . . .	14
<b>3 Het voorspellen van emotie in muziek</b>	<b>17</b>
3.1 Data voorverwerking en feature extractie . . . . .	17
3.1.1 Echonest features . . . . .	19
3.1.2 Songtekst features . . . . .	22
3.2 Classificatie en Experimenten . . . . .	23
3.2.1 Classificatie . . . . .	23
3.2.2 Experimenten . . . . .	23
<b>4 Het voorspellen van muziekfactoren</b>	<b>26</b>
4.1 Inleiding . . . . .	26
4.1.1 Mel Frequency Cepstral Coefficients (MFCCs) . . . . .	26
4.2 Feature Learning . . . . .	28

---

4.2.1	Feature encoding . . . . .	29
4.3	Voorspellen van muziekfactoren . . . . .	31
4.3.1	Ridge-regression . . . . .	31
4.3.2	Resultaten . . . . .	32
4.4	Het genereren van aanbevelingen . . . . .	33
4.4.1	Resultaten . . . . .	35
4.4.1.1	Experiment 1 . . . . .	35
4.4.1.2	Experiment 2 . . . . .	37
4.4.1.3	Experiment 3 . . . . .	41
<b>5</b>	<b>Besluit</b>	<b>43</b>
<b>A</b>	<b>Dataset</b>	<b>45</b>
A.1	Million Song dataset . . . . .	45
A.2	MusiXmatch dataset . . . . .	45
A.3	Last.fm dataset . . . . .	46
	<b>Bibliografie</b>	<b>47</b>

# Lijst van figuren

1.1	Een voorbeelddistributie van de populariteit van de nummers in een muziekcollectie	4
1.2	Een voorbeelddistributie van de populariteit van een muziekcollectie . . . . .	9
2.1	De recall op de testset voor de verschillende aanbevelingssystemen wanneer 50 aanbevelingen werden gegenereerd . . . . .	15
2.2	de mAP op de testset voor de verschillende aanbevelingssystemen wanneer 50 aanbevelingen werden gegenereerd . . . . .	15
2.3	taginformatie van tracks die een sterk negatieve waarde versus een sterk positieve waarde hadden voor factor 5 en voor factor 7. . . . .	16
2.4	De relatie tussen de coëfficiënten van de 5e en 7e factor. . . . .	16
3.1	Opbouw van de dataset . . . . .	18
3.2	De 12 basis functies van de timbre vector . . . . .	20
3.3	De relatie tussen de gemiddelde luidheid en de snelheid van aanzet voor nummers met een positieve en negatieve emotie . . . . .	20
4.1	Opsplitsing van het akoestisch signaal in vensters . . . . .	27
4.2	Het samennemen van MFCC-vectoren in vensters . . . . .	31
4.3	overzicht van het trainingsproces van de muziekfactoren uit een geclusterde MFCC- voorstelling . . . . .	33
4.4	De invloed van het aantal clustercentra op de MSE voor verschillende venster- groottes . . . . .	34
4.5	Het aanbevelingsproces met voorspelde muziekfactoren . . . . .	34
4.6	ROC-curve voor alle nummers in de testset voor een bepaalde gebruiker . . . . .	36
4.7	ROC-curve voor voorspelde nummers in de testset voor een bepaalde gebruiker .	36
4.8	gemiddelde AUC waarden voor alle nummers . . . . .	36
4.9	gemiddelde AUC waarden voor voorspelde nummers . . . . .	37
4.10	Opdeling van de data in training en testset . . . . .	38
4.11	Opdeling van de data in training en testset . . . . .	38
4.12	Een voorbeeld van hoe de dataset $X_{test}$ wordt opgesplitst in $X_{test-A}$ en $X_{test-B}$ wanneer $\alpha = 0.5$ . . . . .	39
4.13	AUC voor verschillende waarden van $\alpha$ bij experiment 2 . . . . .	40
4.14	mAP voor verschillende waarden van $\alpha$ bij experiment 2 . . . . .	40
4.15	mAP voor verschillende waarden van $\beta$ . . . . .	41
4.16	AUC voor verschillende waarden van $\beta$ . . . . .	42
A.1	Een voorbeeldfragment uit de dataset. Rechts bevindt zich het gebruikers id, in het midden het songid en rechts het aantal afspeelbeurten. . . . .	46
A.2	Een voorbeeldfragment uit de mxm dataset. . . . .	46

---

A.3 Een voorbeeldbestand uit de last.fm dataset. . . . .	46
--	----

# Lijst van tabellen

1.1	de berekening van de gemiddelde precisie voor een bepaalde gebruiker wanneer $\tau = 10$ nummers worden aangeraden en 3 nummers in de testset zitten . . . . .	7
2.1	De evaluatie van de verschillende aanbevelingssystemen . . . . .	15
3.1	de verzameling van akoestische features die gebruikt werd bij classificatie . . . .	21
3.2	De hyperparameters van de verschillende classifiers . . . . .	23
4.1	AUC voor de verschillende technieken bij experiment 2 . . . . .	39
4.2	mAP voor de verschillende technieken bij experiment 2 . . . . .	40

# Lijst met afkortingen

<b>ALS</b>	Alternating Least Squares, alternerende kleinste kwadratenmethode
<b>AUC</b>	Area Under the Curve, de oppervlakte onder een curve
<b>API</b>	Application Programming Interface
<b>CF</b>	Collaboratieve Filtertechnieken
<b>DFT</b>	Discrete Fouriertransformatie
<b>DCT</b>	Discrete Cosinustransformatie
<b>FPR</b>	Fals Positive Rate, vals positief ratio
<b>MSD</b>	Million Song Dataset
<b>mAP</b>	mean Average Precision, gemiddelde precisie
<b>MFCC</b>	Mel Frequency Ceptral Coefficients
<b>MSE</b>	Mean Squared Error, gemiddelde kwadratische fout
<b>PCA</b>	Principal Component Analysis, principale-componentenanalyse
<b>RSS</b>	Residual Sum of Squares, niet-verklaarde kwadraatsom
<b>SVD</b>	Singular Value Decomposition, singuliere waarden decompositie
<b>TPR</b>	True Positive Rate, echt positief ratio

# Hoofdstuk 1

## Inleiding

In dit hoofdstuk wordt het begrip 'aanbevelingssysteem' uitgelegd waarna de verschillende klassen van aanbevelingssystemen kort toegelicht worden. Daarna wordt het lange staartprobleem uitgelegd en de aanpak die deze masterproef voorstelt om dit probleem te verhelpen. Ten slotte worden de gebruikte metrieken en de dataset waarvan vertrokken werd toegelicht.

### 1.1 Wat zijn aanbevelingssystemen?

Aanbevelingssystemen hebben de afgelopen jaren enorm aan populariteit gewonnen op het internet. De lijst van aanbevolen video's op Youtube, een webshop zoals Amazon die steeds relevante producten probeert te promoten of de online streaming videodienst Netflix die relevante films en series aanraadt zijn allemaal voorbeelden van aanbevelingssystemen.

Deze systemen gaan mensen helpen door gepersonaliseerde informatie te tonen waar de gebruiker in eerste instantie nog niet noodzakelijk aan heeft gedacht. Contentleveranciers proberen vaak een zo groot mogelijk publiek te bereiken zodat de eindgebruiker vaak overspoeld wordt door de enorme hoeveelheid informatie. In de plaats van de gebruiker te laten zoeken wat hij precies nodig heeft uit die informatie, filtert het systeem de informatie en presenteert hij enkel de informatie waarvan vermoed wordt dat die van belang kan zijn voor de gebruiker. Aanbevelingen vormen dus eigenlijk een brug tussen contentleveranciers en eindgebruikers en zijn interessant voor beide partijen. Enerzijds moet de eindgebruiker niet meer op zoek gaan naar de juiste informatie en de aanbieder kan automatisch de juiste doelgroep bereiken.

Terwijl men vroeger nog naar de platenzaak ging om een CD is het tegenwoordig mogelijk om met enkele klikken een enorme hoeveelheid aan muziek terug te vinden en te beluisteren via het internet. Ook hier zit men nog steeds met de kloof tussen de eindgebruiker met zijn specifieke voorkeuren en de muziekindustrie die natuurlijk zoveel mogelijk van zijn muziek wil verkopen. Wat een muzikaanbevelingssysteem doet is het voorzien van de juiste muziek voor de juiste mensen op het juiste moment.

Aanbevelingssystemen kunnen pas gebouwd worden wanneer er reeds een hoeveelheid invoerdata voorhanden is. Deze informatie wordt typisch voorgesteld in een gebruiker-item matrix waar één dimensie correspondeert met de gebruikers en de andere dimensie met de verschillende items. Wanneer een gebruiker een score heeft toegekend aan een bepaald item, spreken we van een systeem met expliciete feedback. Enkele voorbeelden van expliciete feedback zijn de scores die een gebruiker toekent aan een bepaalde film op Netflix, of de “ik vind dit leuk” en “ik vind dit niet leuk” knoppen van Youtube. Een gebruiker heeft typisch slechts een klein percentage van alle items beoordeeld, wat resulteert in een vrij ijle gebruiker-item matrix.

Wanneer er geen expliciete scores voorhanden zijn kan de invoerdata bestaan uit de zoekgeschiedenis, luistergeschiedenis, zoekopdrachten,... . Dit zijn allemaal vormen van impliciete feedback. Wanneer de invoerdata bestaat uit de luistergeschiedenis van verschillende gebruikers, kan elke positie in de gebruiker-item matrix corresponderen met het aantal keer dat een gebruiker een bepaald nummer heeft afgespeeld.

### 1.1.1 Verschillende soorten aanbevelingssystemen

In het algemeen kunnen we 3 grote klassen van aanbevelingssystemen onderscheiden, namelijk inhoudsgebaseerde, sociaal-gebaseerde en hybride systemen [1]. Inhoudsgebaseerde aanbevelingssystemen vormen de meest intuïtieve klasse. Hierbij probeert men een bepaald profiel van de eindgebruiker en het item samen te stellen aan de hand van externe informatie. Een profiel van een gebruiker kan bestaan uit verschillende attributen zoals leeftijd, woonplaats, voorkeur voor een bepaald muziekgenre,... Wanneer men muzieknummers wil aanbevelen kan men attributen verzamelen zoals artiest, tempo, emotie, muziekgenre,... Het aanbevelingsproces bestaat er dan in om beide profielen op elkaar af te stemmen. De internetradio Pandora is een commerciële toepassing van dit type aanbevelingssystemen voortgevloeid uit het Music Genome Project. In dit project heeft men gTPRrobeerd om elk muzieknummer voor te stellen als een gen. Een gen kan op zijn beurt worden voorgesteld als een vector met ongeveer 400 componenten die corresponderen



met o.a. het geslacht van de artiest, toonbereik van het refrein,... De luisteraar kan specifieke voorkeuren opgeven waarna beide profielen op elkaar afgestemd worden. Deze informatie hoeft niet noodzakelijk manueel ingevoerd te worden. Bij een inhoudsgebaseerd aanbevelingssysteem voor muziek kan mijn bijvoorbeeld door akoestische analyse van het muzieknnummer automatisch een goeie schatting maken van attributen zoals tempo en genre. Een voorbeeld van deze automatische detectie wordt besproken in hoofdstuk 2.

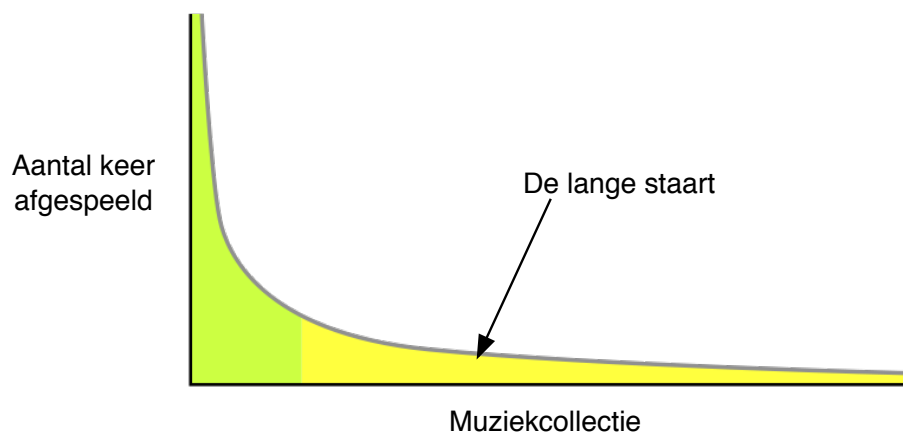
In de praktijk moeten bedrijven zoals Amazon, IMDB en Last.fm het echter vaak stellen met een bepaald gedrag dat de gebruikers vertonen op hun website zoals bijvoorbeeld zoekgeschiedenis of luistergeschiedenis. Dit brengt ons bij de sociaal-gebaseerde aanbevelingssystemen. Het basisidee van sociaal-gebaseerde aanbevelingssystemen of collaboratieve filtertechnieken (CF) is vrij eenvoudig. Wanneer een gebruiker de items A,B,C en D goed vond en er veel gebruikers zijn die items A,B,C,D en E goed vonden dan kan je met vrij hoge zekerheid stellen dat die gebruiker item E ook goed zal vinden. De relaties tussen gebruikers en items worden geanalyseerd om nieuwe associaties te leren tussen gebruikers en items. Een overzicht van de belangrijkste CF wordt gegeven in [2].

Een belangrijk voordeel dat collaboratieve filtertechnieken hebben ten opzichte van inhoudsgebaseerde technieken is dat ze domeinvrij zijn. Dit betekent dat dezelfde technieken kunnen worden gebruikt om verschillende soorten items aan te bevelen. Daar tegenover staat dan wel dat deze technieken onderhevig zijn aan problemen zoals het *koude startprobleem*, waarbij een aanbevelingssysteem er niet in slaagt om items aan te bevelen waarvoor nog geen gebruikersinformatie beschikbaar is. Indien er in het bovenstaande voorbeeld geen enkele gebruiker item E reeds geconsumeerd heeft, dan kan het aanbevelingssysteem onmogelijk weten of dat item van nut zal zijn voor een bepaalde gebruiker. Een ander probleem waarmee CF te maken hebben is dat ze de voorkeur geven om globaal populaire items aan te bevelen ten koste van niche items.

In een poging om het beste van de twee klassen te gaan gebruiken kan men ook inhoudsgebaseerde en sociaal-gebaseerde systemen gaan combineren. In dit geval spreken we van een hybride aanbevelingssysteem. Wanneer er te weinig invoerdata is om CF op te doen kan een inhoudsgebaseerde voorspelling bijvoorbeeld een oplossing bieden zijn.

### 1.1.2 De lange staart

Door de populariteit van het internet is ook de manier waarop mensen muziek consumeren sterk veranderd. Door technologische vooruitgang in netwerken, opslagcapaciteit en afspeelmedia bezit men nu vaak een grotere muziekcollectie dan vroeger. Deze fenomenen leiden echter niet noodzakelijk tot het feit dat mensen meer verschillende muzieknummers beluisteren. Vaak is het zo dat veel muzieknummers zelden of nooit worden afgespeeld. Dit verschijnsel heet de lange staart [3]. Huidige aanbevelingssystemen focussen zich voornamelijk op een hoge graad van nauwkeurigheid. Dit leidt vaak tot het probleem dat deze systemen de voorkeur geven aan populaire items, terwijl items die zich in de lange staart bevinden misschien meer nut hebben voor de gebruiker. Een goed aanbevelingssysteem moet hiermee rekening houden en moet gepersonaliseerde aanbevelingen genereren die de gebruikers helpen om nieuwe, relevante nummers te ontdekken. Een bekend probleem bij sociaal-gebaseerde aanbevelingssystemen is dat ze moeilijker om items aan te bevelen die zich in de lange staart bevinden. De bedoeling van deze masterproef is het bouwen van modellen voor het verbeteren van muziekaanbevelingssystemen en onderzoeken hoe een hybride aanbevelingssysteem op basis van een sociaal-gebaseerde en een inhoudsgebaseerde methode een oplossing kan bieden voor dit probleem en het koude startprobleem.



FIGUUR 1.1: Een voorbeelddistributie van de populariteit van de nummers in een muziekcollectie

### 1.1.3 Evaluatie van een aanbevelingssysteem

Vooraleer men aan muziekaanbeveling kan doen is het belangrijk om een geschikte manier te vinden om de aanbevelingen te evalueren. Voor een aanbevelingssysteem dat expliciete feedback

bevat, is het mogelijk om de gemiddelde kwadratische fout (MSE) te bepalen tussen de score die het systeem voorspelt en de effectieve score die een gebruiker gaf:

$$MSE = \frac{1}{n} \sum_{i=1}^n (p_i - a_i)^2 \quad (1.1)$$

Waarbij  $n$  het aantal voorspelde scores bedraagt,  $p_i$  de voorspelde score voor de gebruiker op item  $i$ , en  $a_i$  de effectieve score die de gebruiker toekende aan item  $i$ .

In deze masterproef werd echter met impliciete feedback gewerkt, namelijk het aantal keer dat een gebruiker een muzieknnummer heeft afgespeeld (zie appendix A). Het aantal afspelbeurten correspondeert niet noodzakelijk met de score die een gebruiker zou geven aan het muzieknnummer. Dit zorgt ervoor dat andere evaluatiemetriecken dienen te worden gebruikt.

Een dataset wordt in een eerste fase altijd opgesplitst in een trainingset en een testset. De trainingset bestaat uit de informatie die gebruikt wordt om de CF te bouwen waarna ze in staat moeten zijn om aanbevelingen voor elke gebruiker te genereren. Een goed aanbevelingssysteem zal erin slagen om aanbevelingen te genereren die effectief in de testset voorkomen. Het systeem zal als het ware de testset moeten voorspellen hoewel het enkel de informatie uit de trainingset te verwerken kreeg.

Allereerst kan worden opgemerkt dat een goede evaluatiemaat voor een aanbevelingssysteem met impliciete feedback de volgende principes in acht neemt:

- Een realistisch aanbevelingssysteem zal maximum  $x$  items aanbevelen.
- Bij het genereren van aanbevelingen zullen ook telkens  $x$  items worden aanbevolen gezien foute aanbevelingen typisch niet worden afgestraft, ze beperken enkel het aantal 'goede' aanbevelingen die kunnen voorkomen.
- De volgorde van aanbevelingen is van belang, dus het systeem zal eerst de items aanbevelen waar het het meest zeker van is.

### 1.1.3.1 mean Average Precision

Een eerste evaluatiemaat waarvan gebruikt gemaakt werd is de gemiddelde precisie (mAP). De mAP is een metriek die de nadruk legt op de eerste items die aanbevolen worden. Er wordt

bovendien een drempelwaarde opgegeven voor het maximum aantal aanbevelingen dat voor elke gebruiker gegenereerd mag worden zodat we kunnen spreken van een afgeknotte mAP. We veronderstellen dat de drempelwaarde  $\tau = 50$ .

Om tot de mAP te komen wordt vertrokken vanuit formule 1.2. De matrix  $M \in \{0,1\}^{m \times n}$  is een binaire gebruiker-item matrix, waarbij  $M_{u,i} = 1$  betekent dat de gebruiker  $u$  naar item  $i$  geluisterd heeft. De lijst van aanbevelingen wordt voorgesteld door  $y$  waarbij  $y(j) = i$  betekent dat item  $i$  voorkomt op positie  $j$  in de lijst. Verder veronderstellen we dat er in een lijst  $y_u$  voor een gebruiker  $u$  geen nummers voorkomen die hij reeds beluisterd heeft. We willen immers nieuwe nummers aanbevelen voor elke gebruiker.

$$P_k(u, y) = \frac{1}{k} \sum_{j=1}^k M_{u,y(j)} \quad (1.2)$$

In bovenstaande formule wordt de precisie op positie  $k$  voor een gebruiker  $u$  en een lijst van aanbevelingen  $y$  berekend. Dit kan gezien worden als de proportie van juist voorspelde items binnen de top- $k$  van de aanbevelingen.

Voor elke gebruiker wordt nu de gemiddelde precisie berekend:

$$AP(u, y) = \frac{1}{n_u} \sum_{k=1}^{\tau} P_k(u, y) \cdot M_{u,y(j)} \quad (1.3)$$

Hierbij stelt  $n_u$  het minimum voor van het aantal aangerade muzieknnummers en het aantal nummers voor die gebruiker die in de testset voorkomen.  $M_{u,y(j)}$  is opnieuw een binaire matrix die waarde 1 bevat als gebruiker  $u$  geluisterd heeft naar het item dat zich bevindt in de aanbevelingslijst op positie  $k$ . Een voorbeeld van de berekening van de gemiddelde precisie wordt weergegeven in tabel 1.1.

In een laatste stap wordt dan uitgemiddeld over alle  $m$  gebruikers om de mAP te bekomen:

$$mAP = \frac{1}{m} \sum_u AP(u, y_u) \quad (1.4)$$

Gebruiker u		
j	relevant?	$P_k(u, y)$
1	ja	1.00
2	nee	0.5
3	nee	0.33
4	nee	0.25
5	ja	0.40
6	nee	0.33
7	nee	0.29
8	nee	0.25
9	ja	0.33
10	nee	0.30
$AP(u, y) = \frac{1.00+0.40+0.33}{3} = 0.58$		

TABEL 1.1: de berekening van de gemiddelde precisie voor een bepaalde gebruiker wanneer  $\tau = 10$  nummers worden aangeraden en 3 nummers in de testset zitten

### 1.1.3.2 Novelty

Men kan zich de vraag stellen of een metriek enkel gebaseerd op relevantie de beste metriek is. Wanneer men bijvoorbeeld het nieuwste album van een zeer bekende popgroep opzoekt op Amazon, wordt men overstelpd met aanbevelingen die de andere albums bevatten van diezelfde popgroep. Voor een webwinkel zoals Amazon zal dit wellicht zijn nut hebben maar de kans is echter vrij groot dat de gebruiker reeds weet heeft van deze albums. Bij het luisteren van muziek is men echter ook vaak op zoek naar nieuwe muziek. Een zelf ontworpen metriek geeft de *nieuwheid* van de aanbevelingen weer:

$$Nov = \frac{1}{m} \sum_u \sum_{i \in y(i)} \frac{\tau}{\log(total\_plays_i)} \quad (1.5)$$

$total\_plays_i$  geeft het totaal aantal afspeelbeurten van een nummer  $i$  terug, berekend over de testset.

### 1.1.3.3 Recall

Een derde metriek die van pas komt bij aanbevelingssystemen is de recall. Recall is de verhouding tussen het aantal relevante aangeraden items en het totaal aantal relevante items voor die gebruiker. Wanneer we voor iedere gebruiker telkens 50 aanbevelingen genereren kunnen we de recall als volgt formuleren:

$$Rec = \frac{1}{m} \sum_u \frac{1}{r} \sum_{j=1}^{\tau} M_{u,y(j)} \quad (1.6)$$

Hierin stelt  $r$  het aantal nummers voor gebruiker  $u$  die in de testset voorkomen.

#### 1.1.3.4 Area Under Curve

Een alternatieve manier om aanbevelingen te evalueren is om niet te kijken naar de  $\tau = 50$  beste aanbevelingen alleen, maar de volgorde van aanbevelingen te gaan bestuderen. In deze masterproef werd gebruik gemaakt van de *Receiver Operating Characteristic curve* (ROC-curve). De ROC-curve vergelijkt de true positive rate (TPR) met de false positive rate (FPR) bij verschillende drempelwaarden. Om een ROC-curve op te bouwen vertrekt men van een *confusiematrix*. Een *confusiematrix* is een tabel met 2 rijen en 2 kolommen die het aantal echt positieve, vals positieve, echt negatieve en vals negatieve gevallen meet tijdens testfase.

	echte waarde	
voorspelling	Echt positief (EP)	Vals positief (VP)
	Vals negatief (VN)	Echt negatief (EN)

Ook de evaluatie van een lijst met aanbevelingen kan men opdelen in 4 disjuncte gevallen:

		echte waarde	
voorspelling	aanbevelen + in testset	aanbevelen + niet in testset	
	niet aanbevelen + in testset	niet aanbevelen + niet in testset	

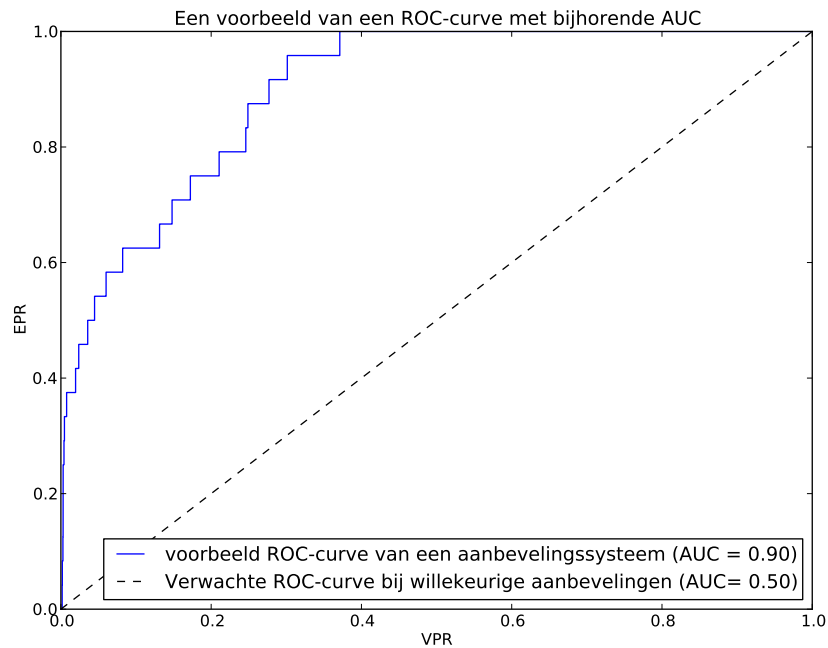
De true positive rate en false positive rate bekomt men aan de hand van volgende formules:

$$TPR = \frac{EP}{EP + VN} \quad (1.7)$$

$$FPR = \frac{VP}{VP + EN} \quad (1.8)$$

Bij evaluatie van het aanbevelingssysteem worden nu alle aanbevelingen in beschouwing genomen. Men start met de hoogste score als drempelwaarde waarna men stap voor stap de drempelwaarde verlaagt tot uiteindelijk alle scores aangeraden worden. In elke stap kan men de TPR en FPR berekenen en uitzetten tegenover elkaar. Wanneer men deze punten nu met

elkaar verbindt, bekomt men een ROC-curve. Een ROC-curve begint altijd in het punt (0,0) en eindigt in het punt (1,1). Een goed aanbevelingssysteem zal snel een hoge TPR bereiken in verhouding tot de FPR. Vandaar dat de oppervlakte onder de curve (AUC) een goede manier is om een aanbevelingssysteem te evalueren. De AUC kan dus bepaald worden door de ROC-curve te integreren en kan een maximale waarde van 1.0 aannemen. Wanneer alle nummers in een willekeurige volgorde worden aanbevolen is de verwachte AUC = 0.5.



FIGUUR 1.2: Een voorbeelddistributie van de populariteit van een muziekcollectie

Een metriek die gebaseerd is op gebruikerstevredenheid is uiteraard de beste optie. Het nadeel van deze soort metrieken is echter dat deze vrij omslachtig zijn aangezien een eindgebruiker telkens de aanbevelingen moet evalueren.

## Hoofdstuk 2

# Collaboratieve Filtertechnieken

In dit hoofdstuk wordt de klasse van collaboratieve filtertechnieken uitvoerig besproken. Vervolgens worden enkele technieken samen met eenvoudige algoritmes vergeleken wanneer men aanbevelingen wil genereren op wanneer een deel van de dataset gebruikt wordt als invoerdata. Het overige deel vormt dan de testset.

### 2.1 Buurgebaseerde modellen

Een eerste soort van collaboratieve filtertechnieken zijn de buurgebaseerde modellen. De voorspelde score voor een bepaalde gebruiker en een bepaald item wordt bepaald op basis van de score van de buren. Deze buren kunnen zowel items als gebruikers zijn. In het eerste geval spreekt men van een gebruikers-georieënteerd model. In deze masterproef werd echter een item-georieënteerd model geïmplementeerd. Dit houdt in dat men voor een bepaalde gebruiker items zal aanbevelen die als *buren* beschouwd kunnen worden van eerder positief bevonden items. Dit vereist natuurlijk dat men eerst a priori voor elk item zijn belangrijkste *buren* kan definiëren. Dit gebeurt door eerst tussen alle items de similariteit te meten op basis van de beschikbare invoerdata met behulp van een similariteitsmaat. Er zijn verschillende gekende similariteitsmaten zoals de Pearson correlatie of de cosinus similariteit. Het zoeken naar een geschikte similariteitsmaat is een belangrijk probleem op zich. Wanneer we nu voor een bepaalde gebruiker een voorspelling moeten doen voor een item gaan we kijken naar de similariteit tussen het nieuwe item en de items waarvoor de gebruiker reeds een rating heeft toegekend. Dit noemt men de



scorefunctie. Ook hier bestaan opnieuw vele varianten. Men kan bijvoorbeeld de gemiddelde score nemen van de 5 beste burens of de betere burens een hoger gewicht toekennen.

### 2.1.1 Algoritme

De implementatie die in deze masterproef werd gebruikt steunt grotendeels op de winnende oplossing van de Million Song Dataset Challenge ([4]). We beschikken over een set  $U$  van  $m$  gebruikers en een set  $I$  van  $n$  items. De gebruikers-item matrix  $R = \{r_{ui}\} \in \mathbb{R}$  wordt voor de eenvoud voorgesteld door een binaire matrix waarvoor geldt dat  $r_{ui} = 1$  als gebruiker  $u$  reeds geluisterd heeft naar nummer  $i$ . De scorefunctie van een gebruiker  $u$  voor een bepaald item  $i$  is als volgt neer te schrijven:

$$h_{ui} = \sum_{j \in I(u)} f(w_{ij}) \quad (2.1)$$

De scorefunctie is dus proportioneel tussen de similariteiten van het item  $i$  en de items die eerder al beluisterd worden door de gebruiker  $u$  ( $j \in I(u)$ ).

Een ander aspect is dat van de similariteitsfunctie. De cosinus similariteit is een populaire keuze. De cosinus similariteit is symmetrisch en door het feit dat we enkel met binaire waarden in de gebruikers-item matrix te maken hebben kan ze dus bovendien snel berekend worden als volgt:

$$w_{ij} = \frac{|U(i) \cap U(j)|}{|U(i)|^{\frac{1}{2}} |U(j)|^{\frac{1}{2}}} \quad (2.2)$$

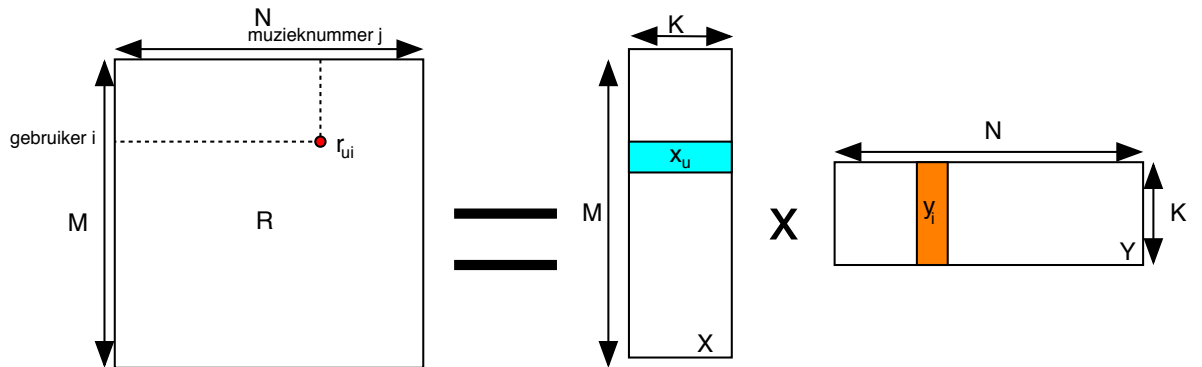
Voor elke gebruiker wordt nu voor de nummers die hij nog niet beluisterd heeft een score bepaald met behulp van vergelijking 2.1. Deze scores worden in dalende volgorde gesorteerd en eerst worden de nummers aangeraden met de hoogste berekende scores.

## 2.2 Verborgene Factormodellen

Een alternatieve aanpak ten opzichte van buurtegebaseerde modellen is gebruik te maken van factormodellen. Hier probeert men de scores tussen gebruikers en items te verklaren door verborgen variabelen bloot te leggen die deze scores verklaren. Binnen deze klasse bestaan verschillende

technieken, maar in deze masterproef werd de focus gelegd op modellen die een matrixfactorisatie genereren uit de gebruiker-item matrix. Een alternatieve aanpak bestaat erin om een singuliere waarden decompositie (SVD) uit te voeren op de gebruikers-item matrix. Standaard SVD-technieken werken echter in op een matrix waarvan alle waarden gekend zijn. In een realistische dataset zijn er echter veel ontbrekende waarden aangezien de gebruikers niet elk item beoordeeld hebben. Dit zorgt ervoor dat er enige aanpassingen nodig zijn. Bovendien zou een SVD oplossing erg te leiden hebben onder overfitting. Overfitting is een gekend probleem binnen het domein van machinaal leren en statistiek. Bij overfitting gaat het model relaties leren die specifiek zijn voor de trainingsdata maar die zich niet veralgemenen over alle data. Een typisch verschijnsel dat optreedt bij overfitting is dat het model zeer goed presteert op de trainingsdata. Wanneer hetzelfde model echter geëvalueerd wordt op nieuwe data zal de prestatie plots veel slechter zijn. Een belangrijk voordeel van matrixfactorisatietechnieken zijn hun hoge nauwkeurigheid en hun goede schaalbaarheid.

Concreet wordt er opnieuw vertrokken van de gebruiker-item matrix  $R$  die we zullen proberen te benaderen als het product van 2 andere matrices  $X$  en  $Y$  die we respectievelijk de gebruikersfactormatrix en de item- of muziekfactormatrix noemen. Indien de gebruikers-item matrix dimensie  $M \times N$  heeft, dan komen de dimensies van  $X$  en  $Y$  overeen met  $M \times K$  en  $K \times N$ , waarbij  $K$  dus vrij te bepalen is. De waarde van  $K$  correspondeert met het aantal verborgen variabelen dat gebruikt wordt om de scores te verklaren.



$$r_{ui} = \sum_k x_{uk} y_{ik}$$

### 2.2.1 Algoritme

Een score is telkens gebaseerd op het inwendig product van de gebruikersfactoren en de muziekfactoren. Wanneer we een gebruiker  $u$  met bijhorende gebruikerfactoren  $x_u$  en een muzieknnummer  $y$  met bijhorende muziekfactoren  $y_i$  voorstellen, dan wordt de voorspelde score  $r_{ui} = x_u^T * y_i$ .

Indien we met expliciete scores werken kunnen we de kostfunctie als volgt neerschrijven:

$$\min_{x^*, y^*} = \sum_{r_{u,i} \text{ is known}} (r_{ui} - x_u^T * y_i)^2 + \lambda(||x_u||^2 + ||y_i||^2) \quad (2.3)$$

Deze kostfunctie minimaliseert het verschil tussen de voorspelde scores en de geobserveerde scores. De parameter  $\lambda$  is de regularisatieparameter en voorkomt dat het model overfit op de trainingdata. Deze kostfunctie neemt dus enkel maar de observeerbare scores in beschouwing. Deze kostfunctie kan geoptimaliseerd worden met behulp van stochastische gradient descent.

Vertrekkend vanuit deze kostfunctie kan men nu een kostfunctie afleiden die overweg kan met een dataset waar enkel impliciete feedback aanwezig is.

Allereerst observeren we dat een waarde  $r_{ui} = 0$  nu niet overeenkomt met een zeer slechte score, maar gewoon met het feit dat de gebruiker het betreffende nummer nog niet beluisterd heeft. Indien een gebruiker een nummer nog niet heeft afgespeeld, kunnen we maar moeilijk bepalen of de gebruiker het nummer goed zal vinden. Hier zijn verschillende redenen voor. De gebruiker kan nog nooit gehoord hebben van het muzieknnummer of hij kan het nummer bijvoorbeeld onmogelijk afspelen. Bovendien is het zo dat indien een nummer slechts enkele malen is afgespeeld ook geen sterke indicatie is dat hij het een goed nummer vindt. De gebruiker kan het nummer beluisterd hebben omdat het toevallig in zijn afspeellijst opdook. We kunnen wel stellen dat naarmate een gebruiker het nummer meerdere keren heeft afgespeeld, we met grotere zekerheid kunnen stellen dat hij het een goed nummer vindt.

We voeren nu een nieuwe variabele  $c_{ij}$  in die het vertrouwen in een bepaalde score modelleert. Deze zal laag zijn wanneer het nummer nog niet werd afgespeeld door de gebruiker maar zal lineair of logaritmisches stijgen volgens het aantal afspelbeurten.  $r_{ui}$  zelf wordt dan vervangen door de binaire waarde 0 of 1, naargelang de gebruiker het item heeft afgespeeld. Dit brengt ons uiteindelijk tot de volgende kostfunctie:

$$\min_{x^*, y^*} = \sum_{u,i} c_{ui} (p_{ui} - x_u^T * y_i)^2 + \lambda \left( \sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right) \quad (2.4)$$

Een belangrijk verschil van deze kostfunctie ten opzichte van 2.3 is dat onze nieuwe kostfunctie nu  $M \times N$  termen heeft, wat in een typische dataset problemen kan geven m.b.t. de schaalbaarheid van ons model.

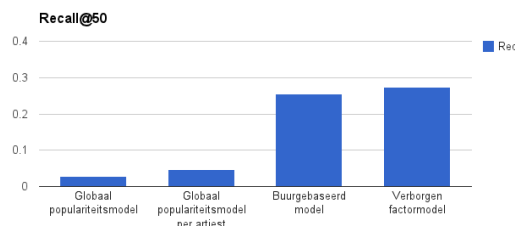
Vergelijking 2.4 kan men opnieuw proberen te optimalizeren met stochastic gradient descent, maar dit zal echter veel te traag convergeren. Een alternatieve oplossingsmethode is om gebruik te maken van Alternating Least Squares(ALS). Wanneer we de gebruikersfactoren of muziektfactoren berekend hebben, dan wordt de kostfunctie wel kwadratisch zodat we het globale optimum snel kunnen berekenen. Wat we in ALS doen is telkens 1 van beide types factoren als constant beschouwen en telkens het andere type factoren optimalizeren in functie van de constante. In elke stap zullen we immers een lagere totale kostfunctie verkrijgen.

In [5] wordt er een efficiënte manier voorgesteld om dit probleem op te lossen waarbij het optimalisatieproces lineair schaalt in functie van de grootte van de dataset.

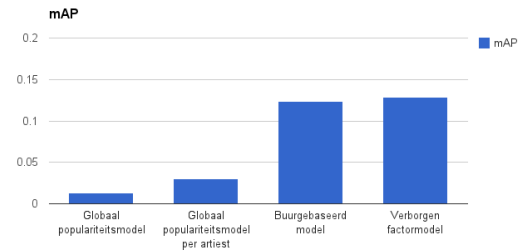
Het product van de muziektfactormatrix en de gebruikers-itemmatrix vormt een benadering voor de originele gebruikers-item matrix  $R$ . We kunnen deze benaderde matrix  $R'$  nu gebruiken om relevante aanbevelingen te genereren. Concreet zullen we voor elke gebruiker  $i$  de nummers aanbevelen met de hoogste score in  $R'$  voor die gebruiker, op voorwaarde dat hij nog niet eerder geluisterd heeft naar dit nummer.

## 2.3 Resultaten

In een eerste fase werden enkele eenvoudige aanbevelingssystemen gebouwd. Het eerste aanbevelingssysteem was gebaseerd op de globale populariteit van alle nummers. Het gaat voor elke gebruiker de nummers aanraden die het meest afgespeeld werden over de volledige dataset op voorwaarde uiteraard dat de gebruiker nog niet naar dat nummer geluisterd heeft. Een iets geavanceerder aanbevelingssysteem gaat eerst gaan kijken welke artiesten een gebruiker reeds beluisterd heeft, en zal enkel de meest globaal populaire nummers aanraden waarvoor de gebruiker minstens al 1 nummer van die artiest eerder heeft beluisterd.



FIGUUR 2.1: De recall op de testset voor de verschillende aanbevelingssystemen wanneer 50 aanbevelingen werden gegenereerd



FIGUUR 2.2: de mAP op de testset voor de verschillende aanbevelingssystemen wanneer 50 aanbevelingen werden gegenereerd

Zoals weergegeven wordt in figuren 2.1 en 2.2, presteerden de geavanceerde aanbevelingssystemen duidelijk beter.

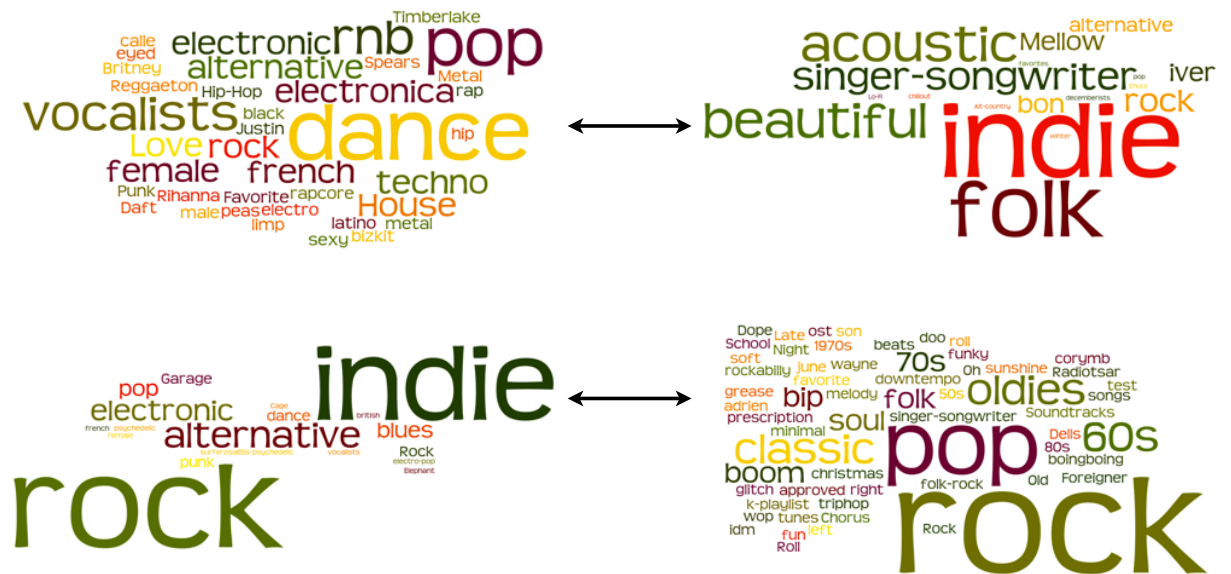
Aanbevelingssysteem	mAP	Nov	Rec
Globaal populariteitsmodel	0.01295	3.90522	0.02739
Globaal populariteitsmodel per artiest	0.03063	4.28761	0.04723
Buurgebaseerd model	0.12405	4.36076	0.25469
Verborgen Factormodel	0.12830	4.37244	0.27362

TABEL 2.1: De evaluatie van de verschillende aanbevelingssystemen

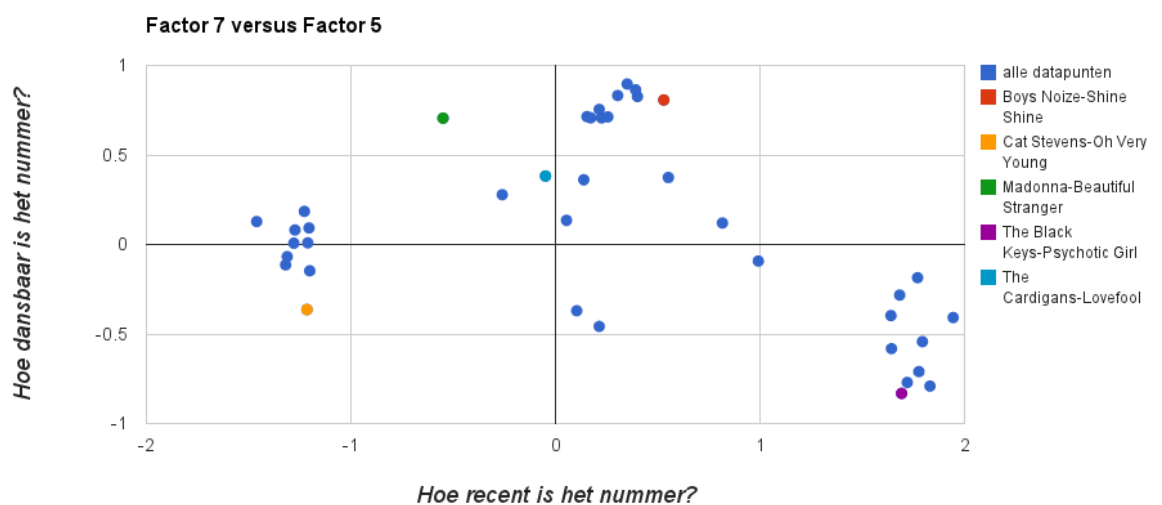
Het grote voordeel van het verborgen factormodel is dat we, vertrekkend vanuit de gebruiker-item matrix, nu een abstracte voorstelling hebben van elk muzieknummer en elke gebruiker in enkele factoren. Deze muziekfactoren kunnen we zien als een soort automatisch gegenereerd profiel gelijkaardig aan wat men bekomt bij een inhoudgebaseerde aanpak. Een bepaalde dimensie kan bijvoorbeeld corresponderen met de geografische locatie van de artiest, de luidheid van een bepaald nummer, dansbaarheid van het nummer,... . De gebruikersfactoren geven aan wat de precieze voorkeuren voor die dimensies zijn. Om dit te verifiëren kan men met behulp van regressie-analyse nagaan of de muziekfactoren te voorspellen zijn uit externe informatie. Een diepere bespreking hiervan volgt in hoofdstuk 4, maar eenvoudige lineaire regressie waarbij de te voorspellen muziekfactoren uit vrij beschikbare akoestische kenmerken van de bijhorende muzieknummers werden geschat leerde ons reeds heel wat. Het viel op dat de features of kenmerken die de muziek het meest verklaren, zoals timbre-features en pitch-informatie de grootste coëfficiënten bevatten in absolute waarde na lineaire regressie.

Om nog een betere indicatie van die muziekfactoren te krijgen werden de de nummers met de meest extreme waarden voor een factor bepaald waarna bijhorende tag informatie werd opgehaald uit een Last.fm dataset. Enkele factoren kunnen duidelijk gelinkt worden aan een bepaald kenmerk van het muzieknummer zoals genre en tijdsgeest. Het resultaat hiervan is gevisualiseerd

in figuur 2.3. In figuur 2.4 werden de waarden voor 2 factoren ten opzichte van elkaar uitgezet. Belangrijk hierbij op te merken is dat de muziekfactoren niet absoluut moeten geïnterpreteerd worden maar relatief. Wanneer een muzieknummer A bijvoorbeeld een grotere waarde bezit voor factor 7 dan muzieknummer B, dan kan verwacht worden dat nummer A recenter werd uitgebracht dan nummer B. Om een exact tijdsverschil tussen beide nummers te bepalen is de factor echter te onnauwkeurig.



FIGUUR 2.3: taginformatie van tracks die een sterk negatieve waarde versus een sterk positieve waarde hadden voor factor 5 en voor factor 7.



FIGUUR 2.4: De relatie tussen de coëfficiënten van de 5e en 7e factor.

## Hoofdstuk 3

# Het voorspellen van emotie in muziek

Muziek en emotie zitten dicht bij elkaar. Mensen luisteren vaak naar verschillende muziekstijlen op basis van hun stemming. Automatische detectie van emotie van een bepaald nummer kan dus een belangrijke component in muziekaanbeveling zijn. Naast de traditionele collaboratieve filtertechnieken kan de herkenning van emotie in muziek een oplossing bieden voor het *koude startprobleem*. Emotieherkenning van muziek kan gebaseerd zijn op akoestische features, songteksten of metadata zoals taginformatie van een bepaald nummer. In dit hoofdstuk wordt onderzocht of men aan de hand van akoestische features en songtekstinformatie emotie kan herkennen in muziek. Er wordt vertrokken van een gelabelde dataset. Dit is een dataset met al deze informatie over de muzieknummers en telkens de aanduiding van de stemming. Uit deze dataset gaat het algoritme emoties leren herkennen. Het resultaat is dat het algoritme in staat moet zijn om emotie te herkennen in nieuwe muzieknummers.

### 3.1 Data voorverwerking en feature extractie

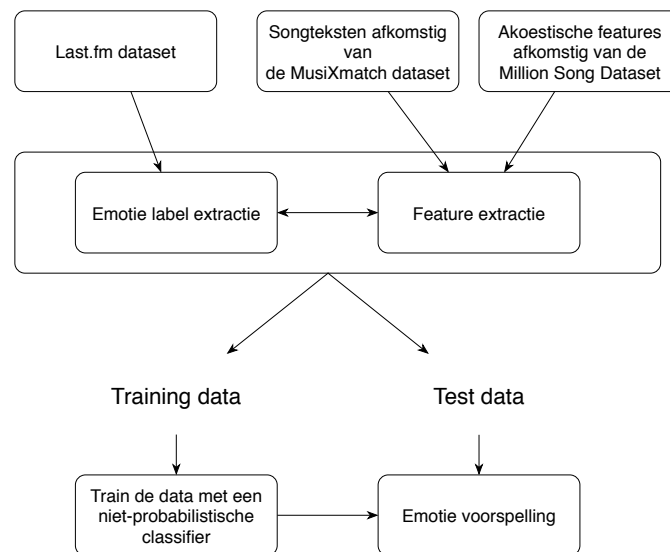
Een eerste overweging die werd gemaakt is hoeveel emoties er in rekening gebracht moeten worden. Er bestaan vele soorten emoties, de ene al gemakkelijker te herkennen dan de andere. Hoe meer emoties we in rekening brengen, hoe geavanceerder het algoritme zal moeten zijn om het onderscheid te kunnen maken. Daarnaast is het aantal emoties ook een belangrijk aspect tijdens het opstellen van de gelabelde dataset. Het is eenvoudig om het verschil te herkennen

tussen blijdschap en verdriet, maar het verschil tussen verdriet en woede is vaak veel subjectiever en dus moeilijker om te classificeren. In een eerste fase werd enkel onderscheid gemaakt tussen een positieve emotie en een negatieve emotie. In een later stadium werd een negatieve emotie nog opgesplitst tussen verdriet en woede.

De dataset waarvan vertrokken werd bestond uit een subset van de Million Song Dataset (MSD) [6]. Aangezien het manueel annoteren van emotie bij elk nummer tijdsrovend en subjectief kan zijn en in Levy et al.[7] aangetoond wordt dat sociale taginformatie een goede semantische descriptor is van het muzieknnummer werd de initiële dataset gelabeld aan de hand van taginformatie. Concreet werd taginformatie van een Last.fm dataset gebruikt [6]. Muzieknummers waarvan de taginformatie woorden bevatte die corresponderen met een positieve of negatieve emotie werden geselecteerd en gekoppeld met de MSD. Songtekstinformatie werd verkregen door de MSD te koppelen met de MusiXmatch dataset [6].

Concreet werd er vertrokken vanuit 2 verschillende datasets die respectievelijk 783 nummers en 5142 nummers bevatten met ongeveer evenveel nummers die gelabeld werden met een positieve of negatieve emotie.

Figuur 3.1 toont een overzicht van dit proces.



FIGUUR 3.1: Opbouw van de dataset



### 3.1.1 Echonest features

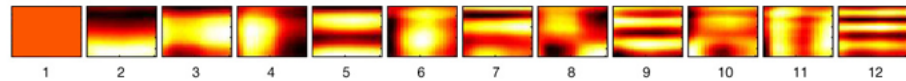
De akoestische features zijn afkomstig van de MSD. In het kader van de MSD-challenge<sup>1</sup> was heel wat informatie reeds offline beschikbaar, maar omdat bepaalde features zoals *danceability* en *energy* wellicht ook enige correlatie vertonen met de emotie van een muzieknummer, werd besloten om deze features nog eens afzonderlijk op te halen m.b.v. de Echonest API. Dit is een webservice die toegang geeft tot een enorme hoeveelheid aan akoestische informatie over muzieknummers. Helaas waren er verschillende ontbrekende waarden voor deze features. Een eenvoudige oplossing zou deze nummers gewoon verwijderen uit de dataset, maar de ontbrekende waarden vervangen door de gemiddelde waarde voor de feature bleek een beter alternatief.

Echonest deelt een muzieknummer op in verschillende segmenten. Een segment is een verzameling van opeenvolgende geluiden die uniform zijn in timbre en harmonie. De duur van een segment ligt typisch onder 1 seconde. Aan elk segment zijn vervolgens verschillende features gekoppeld. De *segment pitches* en *segments timbre* features worden door Echonest beschreven als toongebaseerde en MFCC-gebaseerde features. Een *segment pitch* feature bestaat uit een 12-dimensionele vector waarvan de verschillende componenten waarden bevatten tussen 0 en 1. Deze beschrijven de intensiteit van een bepaalde grondtoon in een bepaald segment van het nummer, er wordt dus geen onderscheid gemaakt tussen de verschillende harmonischen van deze grondtoon. De exacte implementatie van de *segment timbre* feature is onbekend, de enige informatie die gevonden kon worden is dat deze feature correspondeert met een projectie van het spectro-temporele oppervlak op een lagere deelruimte van 12 dimensies. De verschillende basisfuncties van deze projectie worden weergegeven in figuur 3.2. De horizontale as correspondeert met het tijdsverloop terwijl de verticale as overeenstemt met het frequentieverloop. De kleuren variëren van zwart tot wit en corresponderen met de amplitude. Het is moeilijk om de exacte relatie tussen de basisfunctie en het waargenomen geluid te bevatten, maar we kunnen bijvoorbeeld de eerste component van de timbre vector omschrijven als de gemiddelde luidheid over het volledige segment. De vierde component van de vector zal een grote waarde bevatten wanneer de segmenten een snellere aanzet hebben. De aanzet of *attack* van een toon is de wijze waarop een hoorbaar geluid begint en zijn maximale luidheid bereikt. Geluiden met een snelle aanzet zijn bijvoorbeeld geweerschoten of het toeslaan van een deur terwijl het traag openen van een piepende deur dan weer overeenkomt met een trage aanzet. Bij wijze van voorbeeld is in figuur 3.3 de 1e dataset gevisualiseerd in functie van de gemiddelde eerste en vierde component

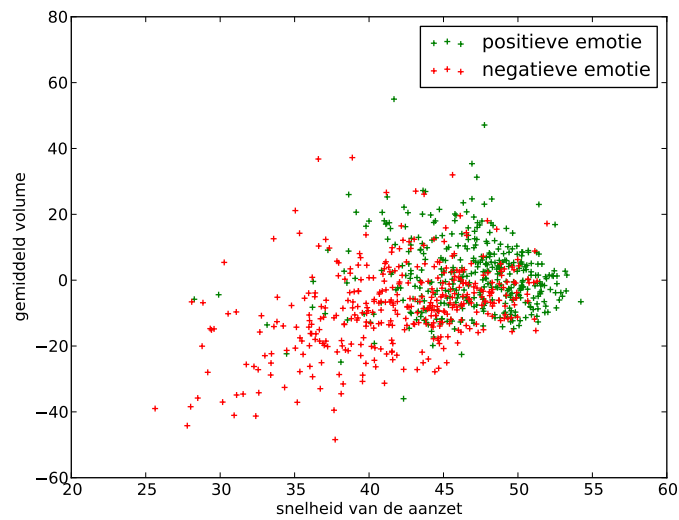
---

<sup>1</sup><http://www.kaggle.com/c/msdchallenge>

van elk nummer. We kunnen duidelijk zien dat muzieknummers met een positieve emotie eerder een luider volume en een snellere aanzet zullen hebben dan muzieknummers met een negatieve emotie.



FIGUUR 3.2: De 12 basis functies van de timbre vector



FIGUUR 3.3: De relatie tussen de gemiddelde luidheid en de snelheid van aanzet voor nummers met een positieve en negatieve emotie

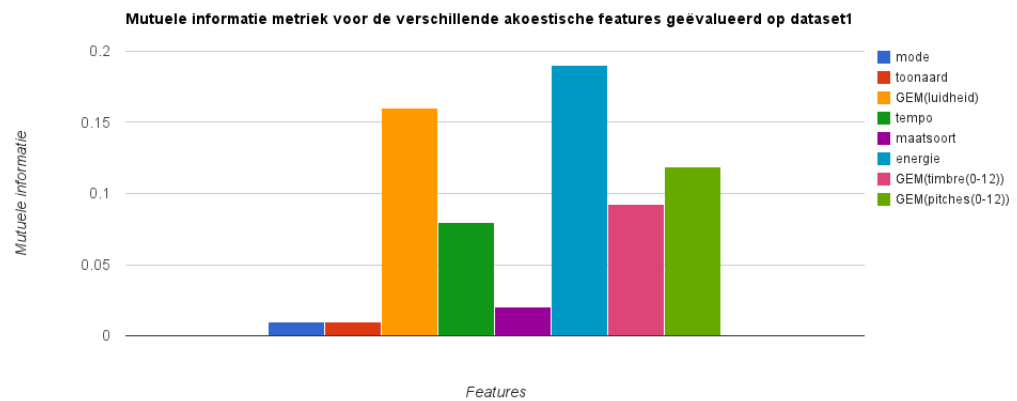
Verschillende muzieknummers hebben typisch een verschillende duur. Het aantal segmenten is dus verschillend per nummer. Om de verschillende features om te zetten naar een tijds-invariante representatie werd beslist om verschillende statistische metrieke van de *segments timbre* en *segment pitches* op te nemen. In de praktijk bleken het gemiddelde, de variantie, de mediaan en de minimale en maximale waarden het beste resultaat op te leveren.

Van de toonaard en het tempo van een muzieknummer zou men ook kunnen verwachten dat ze in sterke mate de emotie van het nummer beïnvloeden. Alvorens deze features op te nemen in de classifier, werd eerst op een gelijkaardige manier als in figuur 3.2 onderzocht of dit wel klopte. Verder werden ook verschillende features die te maken hadden met de luidheidsverdeling doorheen een muzieknummer in rekening gebracht.

Vooraleer er verschillende combinaties van features werden uitgetest leek het interessant om in een eerste fase een idee te krijgen welke features het belangrijkste zijn bij emotieherkenning van muziek. Daarvoor werd gebruik gemaakt van de mutuele informatie. De mutuele informatie is

de reductie in onzekerheid over een bepaalde variabele  $X$ , of de verwachte reductie in het aantal ja-nee vragen die gesteld moeten worden om de variabele  $X$  te bepalen na het observeren van  $Y$ . In dit geval kan de variabele  $X$  gelinkt worden aan de emotie van het nummer en de variabele  $Y$  komt dan overeen met de feature.

Features die een hoge mutuele informatie hebben m.b.t. de emotie zullen dus leiden tot een betere classificatie.

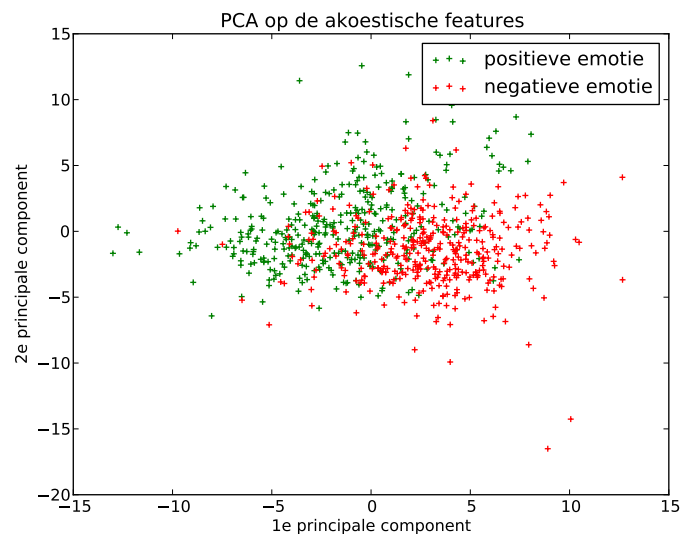


Zoals in bovenstaande figuur zichtbaar is, zal het incorporeren van de toonaard en de toonsoort (mode) wellicht niet leiden tot een betere classificatie. Tabel 3.1 bevat de combinatie van features die resulteerden in een optimale trade-off tussen de nauwkeurigheid van de classifier en de tijd die het duurde om de classifier te trainen.

No.	feature
0-11,	Gemiddelde van elke timbre component
12-23	Variantie van elke timbre component
24-35	Mediaan van elke timbre component
36-47	Minimum van elke timbre component
48-59	Maximum van elke timbre component
60-71,	Gemiddelde van elke pitch component
72-83	Variantie van elke pitch component
84-95	Mediaan van elke pitch component
96-107	Minimum van elke pitch component
108-119	Maximum van elke pitch component
120	Tempo
121	gemiddelde luidheid
122	variantie luidheid
123	hoogste-laagste luidheid
124	Energy

TABEL 3.1: de verzameling van akoestische features die gebruikt werd bij classificatie

Tenslotte werd gebruik gemaakt van principale-componentenanalyse (PCA) om de hoogdimensionale feature vector te kunnen visualiseren. PCA is een analysemethode die toelaat om een hoeveelheid gegevens te beschrijven met een kleiner aantal relevante grootheden. Dit zorgt ervoor dat een muziknummer met 125 kenmerken geschreven kan worden in functie van 2 kenmerken zodat het visueel geïnterpreteerd kan worden. Het resultaat is te zien in onderstaande figuur, waarbij reeds kan worden afgeleid dat het bouwen van een classifier die emoties kan onderscheiden een complex probleem is.



### 3.1.2 Songtekst features

Wegens het specifieke formaat van de songtekst dataset was het moeilijk om een geschikte feature representatie te vinden. De songteksten werden voorgesteld in een bag-of-words formaat: elk nummer werd beschreven aan de hand van het aantal keer dat een topwoord voorkwam. Een woord is een topwoord indien het in de top-5000 lijst voorkwam die gebaseerd was op het aantal keer dat het woord voorkomt over alle songteksten. Helaas waren de meeste topwoorden niet significant ('the, and, we, I') wanneer we aan emotieherkenning willen doen. Daarom werd een subset gevormd van deze topwoorden. Deze subset ontstond uit de doorsnede van de topwoorden met verschillende gevoelswoord datasets zoals General Inquirer [8] en AFINN [9]. Dit zorgde ervoor dat er slechts 2049 topwoorden overbleven. Daarna werd geëxperimenteerd met verschillende feature representaties zoals binair (komt een gevoelswoord voor in de songtekst?) of een tf-idf representatie, maar uiteindelijk werden de beste resultaten verkregen wanneer er gewoon met de frequentie van elk topwoord in het muziknummer werd gewerkt.

## 3.2 Classificatie en Experimenten

### 3.2.1 Classificatie

Voor muziekclassificatietaken worden vaak k-nearest neighbour (K-NN) en support vector machines (SVM) gebruikt [10]. Mede door hun recent toenemende populariteit werd gekeken of emotieherkenning aan de hand van Random Forests ook goeie resultaten opleverde. De implementatie van de verschillende classifiers is afkomstig uit scikit-learn 0.12.1 [11], een bibliotheek die verschillende machine-learning algoritmes bevat voor de Python programmeertaal. Er werd telkens gebruik gemaakt van 10-fold kruisvalidatie in combinatie met grid search om de hyperparameters voor de verschillende classifiers te bepalen. Kruisvalidatie houdt in dat de data wordt opgesplitst in verschillende gelijke delen waarna er vervolgens telkens 1 deel afgesplitst wordt om de classifier op te testen, terwijl de overige K-1 delen gebruikt worden tijdens het trainen. Dit proces wordt K keer herhaald waarna de uiteindelijke nauwkeurigheid wordt bepaald door het gemiddelde te nemen van de K nauwkeurigheden op de testsets. Grid search is een manier om de hyperparameters van een classifier te bepalen. Bij grid search wordt gewoon exhaustief gezocht in een subset van de hyperparameter ruimte. Tabel 3.2 bevat een overzicht van de verschillende hyperparameters en hun waarden voor de classifiers die enkel gebruik maken van akoestische features en de classifiers die enkel gebruik maken van songtekstinformatie.

classifier	hyperparameters	dataset 1		dataset 2	
K-Nearest Neighbors (KNN)	Aantal burens	k = 6		k = 8	
Support Vector Machines (poly)	Straf van de foutterm graad van de polynoom	C = 100 graad = 3		C = 10 graad = 1	
Support Vector Machines (RBF)	Straf van de foutterm kernel coëfficiënt	C = 1 $\gamma = 0.0001$		C = 10 $\gamma = 0.001$	
Random Forest	Aantal bomen	n = 400		n = 500	
	hyperparameters	alle words	gevoelswoorden	alle woorden	gevoelswoorden
Lineair SVM	Straf van de foutterm	C = 0.00001	C = 1.5	C = 0.001	C = 0.5
Naive Bayes	Smoothing parameter	$\alpha = 0.01$	$\alpha = 5$	$\alpha = 1.5$	$\alpha = 5$

TABEL 3.2: De hyperparameters van de verschillende classifiers

### 3.2.2 Experimenten

In een eerste experiment werd de nauwkeurigheid van de verschillende classifiers geëvalueerd. De nauwkeurigheid is gedefiniëerd als het percentage juist voorspelde emoties, gegeven de features van elk nummer. In eerdere studies zoals [12] werd aangetoond dat SVMs de beste resultaten opleveren m.b.t. emotieherkenning in muziek. Na het uitvoeren van het experiment bleek

dat Random Forests erin slagen om een gelijkaardige nauwkeurigheid te realiseren. In onderstaande tabel is een overzicht terug te vinden van de resultaten van de verschillende classifiers die bekomen werden na 10-fold kruisvalidatie op beide datasets.

classifier	nauwkeurigheid op dataset 1	nauwkeurigheid op dataset 2
K-Nearest Neighbors (KNN)	0.7842	0.8213
Support Vector Machines (poly)	0.7906	0.8394
Support Vector Machines (RBF)	<b>0.797</b>	0.8431
Random Forest	0.7906	<b>0.8454</b>

In een volgende experiment werd onderzocht hoe sterk de nauwkeurigheid zou afnemen indien er een extra emotie werd toegevoegd. Er werd vertrokken vanuit de eerste dataset waarbij een negatieve emotie verder werd opgesplitst in woede en verdriet. Zoals verwacht is er een sterke reductie in nauwkeurigheid.

classification	nauwkeurigheid
2-class	0.7906
3-class	0.6926

Om een verklaring te vinden voor deze sterke afname werd er onderzocht wat de voorspelling was voor elk muziknummer. Bij dit experiment werd gebruik gemaakt van de Random Forest classifier. De tabellen geven het percentage voorspelde emoties terug in vergelijking met de echte emoties die gekoppeld zijn aan de muziknummers.

				Voorspelde emotie							
				postief	negatief						
Echte emotie	postief	0.8111	0.1889	Echte emotie	postief	0.6045	0.2773	0.1182			
	negatief	0.1684	0.8316		droevig	0.1676	0.8113	0.0211			
								woede	0.3633	0.1490	0.4878

Op basis van deze resultaten blijkt het dat het moeilijker is om muziektracks gekoppeld aan een positieve emotie te onderscheiden dan muziektracks gekoppeld aan een woede emotie dan om het onderscheid te maken tussen een droevige en een woede emotie. Dit lijkt wat contra-intuïtief, maar als rekening gehouden wordt met het feit dat de voorspelling enkel gebeurde op basis van akoestische informatie, kan dit kloppen. Droevige muziek is vaak sereen terwijl nummers die

een positief gevoel of woede opwekken vaak veel meer actie bevatten. Verder onderzoek van de gelabelde dataset leidde tot de vaststelling dat muzieknnummers die het woedelabel opgeplakt kregen vaak gerelateerd waren aan elektronische muziek of het metal genre.

Als laatste experiment werd onderzocht of emotieherkenning op basis van songtekstinformatie mogelijk is en of de combinatie van akoestische en songtekstinformatie in een betere classifier resulteert. Helaas werd er niet voor elk nummer een bijhorende songtekst gevonden wat ervoor zorgde dat de dataset 1 gereduceerd werd van 783 naar 537 nummers, en dataset 2 van 5142 naar 2886 nummers.

methode	aantal woorden	classifier	nauwkeurigheid op dataset 1	nauwkeurigheid op dataset 2
all woorden	5000	Naive Bayes	0.6115	<b>0.7278</b>
		LinearSVC	0.589	0.7174
gevoelswoorden	2039	Naive Bayes	0.6723	0.7211
		LinearSVC	0.6462	0.6393

Deze resultaten tonen duidelijk aan dat muziekclassificatie op basis van akoestische features veel beter werkt dan op basis van songteksten. Enkel gevoelswoorden in beschouwing nemen lijkt enkel voordelig te zijn bij een kleinere dataset. Het classificeren van een nummer enkel op basis van zijn songtekst blijkt in de praktijk ook vaak moeilijk te zijn. Allereerst kan een nummer bestaan uit meerdere emoties, denk maar aan muzieknnummers die droevig beginnen maar eindigen met een positieve noot. Bovendien kunnen songteksten vaak vrij poëtisch zijn waarbij een gevoel of emotie beschreven wordt zonder expliciet gebruik van gevoelswoorden. Een laatste reden die aangewend kan worden is het feit dat in sommige nummers de melodie en songtekst tegenstrijdig met elkaar zijn. Een nummer over liefdesverdriet kan gekoppeld zijn aan een oppeppende melodie bijvoorbeeld.

Een classifier die zowel gebruik maakte van de songteksten en akoestische features haalde een nauwkeurigheid van 82.29% op dataset 2. Dit is dus lager dan een classifier enkel gebaseerd op akoestische features. In [10] wordt echter aangetoond dat deze hybride aanpak wel tot een betere classificatie kan leiden. Wellicht zit het probleem in de slechte voorstelling van de songteksten.

## Hoofdstuk 4

# Het voorspellen van muziekfactoren

### 4.1 Inleiding

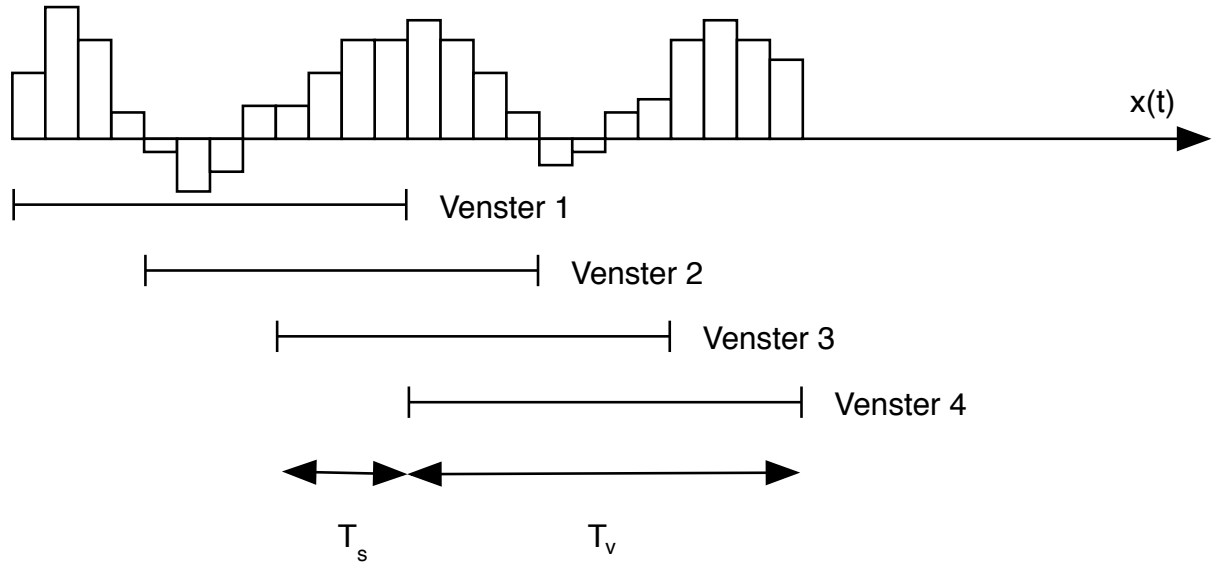
In dit hoofdstuk wordt geprobeerd om de muziekfactoren van een verborgen factormodel dat besproken werd in hoofdstuk 2 nu rechtstreeks uit het muzieknnummer zelf te gaan voorspellen. Geluid is een akoestische golf die correspondeert met drukveranderingen die zich in de lucht voortplanten. Wanneer men geluid opneemt met behulp van een microfoon, dan bekomt men een elektrisch signaal dat kan worden beschreven in functie van de tijd. Men noemt deze functie ook vaak het akoestisch signaal. Een computer kan geen continue functie voorstellen en zal dus dit akoestisch signaal gaan benaderen als een serie van gequantiseerde waarden. Een digitale CD heeft bijvoorbeeld een sampling rate van 44,100 Hz en een bereik van 16 bits. Dit betekent dat elke seconde geluid voorgesteld kan worden door 44,100 getallen of samples die elk voorgesteld kunnen worden door 16 bits. Rechtstreeks muziekfactoren gaan voorspellen op basis van deze getallen is uiteraard onbegonnen werk. Vandaar dat eerst geprobeerd wordt om deze getallen om te zetten naar een compactere representatie.

#### 4.1.1 Mel Frequency Cepstral Coefficients (MFCCs)

MFCCs [13] kennen hun oorsprong in spraakverwerking, maar zijn in het verleden reeds succesvol toegepast bij het modelleren van muziekfragmenten. Een MFCC-decompositie probeert een voorstelling te maken van het geluid m.b.t. hoe het menselijk gehoor dit geluid ervaart. Om over te gaan van een digitaal audiosignaal naar een MFCC-voorstelling zijn verschillende stappen nodig.



Eerst wordt het akoestisch signaal  $x(t)$  opgesplitst in verschillende overlappende vensters die elk afzonderlijk zullen worden geanalyseerd. In deze masterproef werd de lengte van een venster  $T_V$  vastgelegd op 25 ms en de verschuivingstijd  $T_S$  bedroeg 10ms.



FIGUUR 4.1: Opsplitsing van het akoestisch signaal in vensters

Daarna kan het signaal door een eenvoudige hoogdoorlaatfilter gestuurd worden. Deze filter modelleert het feit dat het gehoor lage tonen minder goed waarneemt dan hogere tonen. Elk venster wordt vervolgens omgezet vanuit het tijdsdomein naar het frequentiedomein via een discrete fouriertransformatie (DFT). We bezitten nu voor het signaal de intensiteitwaarden bij verschillende frequenties, maar dit komt niet overeen met de intensiteit die mensen waarnemen. Mensen kunnen o.a. het verschil tussen een klein verschil in frequentie of toonhoogte beter waarnemen bij lage frequenties dan bij hoge frequenties. Om dit probleem op te lossen wordt de output van de DFT geconverteerd naar de mel schaal. De mel schaal relateert de waargenomen frequentie van een zuivere toon aan de echte frequentie van die toon. Een populaire methode om van het frequentiedomein over te gaan naar de mel schaal is:

$$mel_{freq} = 2595 * \log_{10} \left[ 1 + \frac{f}{700} \right] \quad (4.1)$$

Een andere belangrijke eigenschap van het gehoor is dat het eerder gevoelig is voor relatieve dan absolute verschillen in intensiteit. Om het waargenomen volum van een geluid te verdubbelen moeten we 8 keer zoveel energie aan de geluidsbron aanleggen. Om dit te modelleren in de MFCC-decompositie is het aangewezen om de logaritme van de mel schaal te nemen.

Om uiteindelijk een discrete voorstelling te verkrijgen in een beperkt aantal coëfficiënten wordt in de laatste stap een discrete cosinustransformatie (DCT) uitgevoerd. Door de specifieke eigenschappen van de DCT zullen de eerste coëfficiënten de belangrijkste frequentie-eigenschappen van het signaal bevatten en de hogere coëfficiënten eerder de details van het spectrum. In deze masterproef werden enkel de eerste 12 coëfficiënten overgehouden om het log-mel-spectrum voor te stellen en vormen ze een MFCC-vector  $C = \langle c_0, c_1, \dots, c_{11} \rangle$ .

Net zoals de opeenvolgende samples waarvan vertrokken werd vaak met elkaar gecorreleerd zijn, zijn ook opeenvolgende MFCC-vectoren met elkaar gecorreleerd. Omdat we in een later stadium enkel MFCC's afzonderlijk gaan behandelen dreigt deze correlatie verloren te gaan. Een mogelijke oplossing bestaat erin om een MFCC van een venster  $n$  uit te breiden met informatie over de MFCC's van zijn omliggende vensters. Dit wordt gemodelleerd door de eerste-orde differenties  $\Delta C_{nk}, k = 0, \dots, 11$ :

$$\Delta C_{nk} = \frac{\sum_{m=1}^{m=2} m(X_{n+m,k} - X_{n-m,k})}{\sum_{m=1}^{m=2} m^2} \quad (4.2)$$

Elke MFCC-vector bestaat nu uit 24 componenten. In deze masterproef werd gebruik gemaakt van de bibliotheek TuneR [14] om de MFCC-vectoren voor verschillende audiosignalen te verkrijgen.

## 4.2 Feature Learning

Rechtstreekse voorspelling van de muziekfactoren op basis van de MFCC's zou niet werken goed werken omdat deze voorstelling nog steeds te ruw is om verbanden uit te leren. Daarom werd gekeken om MFCC's op een hoger niveau te gaan beschrijven. Dit is een typische voorbeeldtoepassing van *feature learning*, een verzameling van technieken binnen machinaal leren waarbij de kenmerken, vaak ook features genaamd, eerst omgezet worden naar een nieuwe ruimte die beter geschikt is voor een gesuperviseerde taak zoals bijvoorbeeld lineaire regressie.

Na het lezen van [15] werd besloten om het K-means algoritme te gebruiken als feature learning algoritme. [15] toont namelijk aan dat andere factoren zoals *whitening* van de data en het aantal nieuwe features een grotere impact hebben op de prestatie van de gesuperviseerde taak dan de complexiteit van het feature learning algoritme zelf. Een bijkomend voordeel van K-means ten

opzichte van andere feature learning algoritmes is dat het zeer snel is, enkel het aantal clusters heeft als hyperparameter en dat het eenvoudig zelf te implementeren is.

#### 4.2.1 Feature encoding

Vertrekkende vanuit een verzameling MFCC-vectoren wordt nu geprobeerd om een meer compacte voorstelling van elk muzieknummer te verkrijgen. Elk muzieknummer wordt voorgesteld door een sequentie van 2905 MFCC-vectoren met elk dimensie 24. Een allereerste stap in een machinaal leren algoritme is vaak dat er normalisatie van de invoerdata plaats vindt. Dit betekent dat elke component in de MFCC-vector vervangen wordt door de gemiddelde waarde van die component berekend over de volledige invoerdata af te trekken van de originele waarde en vervolgens te delen door de standaardafwijking op deze component. Op die manier heeft elke feature een gemiddelde waarde 0 en bedraagt de standaardafwijking van elke feature 1. Vergelijking 4.3 toont een voorbeeld van input normalisatie:

$$\begin{bmatrix} 1 & -1 & 2 \\ 2 & 0 & 0 \\ 0 & 1 & -1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & -1,22... & 1,33... \\ 1,22... & 0 & -0,26... \\ -1,22... & 1,22... & -1,06... \end{bmatrix} \quad (4.3)$$

Verder wordt in [15] ook aangetoond dat *whitening* van de data resulteert in betere resultaten. Een whitening transformatie transformeert een set van variabelen met covariantiematrix  $M$  in een set van nieuwe variabelen waarvan de covariantiematrix de eenheidsmatrix is, zodat alle variabelen ongecorrleerd zijn en allemaal variantie 1 hebben.

Een populaire methode om de geobserveerde data te whitenen is om gebruik te maken van de eigenwaardedecompositie van de covariantiematrix  $X$ .

Nadat deze voorverwerkingsstappen uitgevoerd zijn, is het tijd om nieuwe, geschiktere features te leren die de muziekfactoren kunnen voorspellen. Het K-means clusteralgoritme deelt de invoerdata op in verschillende clusters. Elke cluster bezit 1 centrum. Dat correspondeert met de gemiddelde waarde berekend over alle elementen die in de cluster aanwezig zijn. Het algoritme wordt gebruikt om centra te leren uit de inputdata waarna elke MFCC-vector toegewezen wordt aan een of meerdere centra. Het centrum van de cluster waaraan de MFCC-vector werd toegewezen kan nu worden gebruikt als nieuwe voorstelling. De enige parameter van het algoritme is het aantal clusters  $K$  dat gebruikt wordt om de dataset te clusteren.

De trainingsfase van het K-means algoritme werkt als volgt:

1. Initialiseer K verschillende clustercentra  $\mu_k$
2. Ken elke inputvector  $x_n$  toe aan het dichtsbijzijnde clustercentrum volgens deze formule:

$$r_{nk} = \begin{cases} 1 & \text{als } k = \operatorname{argmin}_j \|c^{(j)} - x_n\|_2^2 \\ 0 & \text{anders} \end{cases} \quad (4.4)$$

$r_{nk}$  is 1 enkel indien inputvector  $x_n$  werd toegekend aan cluster k met gemiddelde  $\mu_k$

3. Update de clustercentra:

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} \quad (4.5)$$

4. Indien er geen wijzigingen meer waren in stap 3 of aan een andere voorwaarde voldaan is, stop. In het andere geval keer je terug naar stap 2.

In de praktijk bleek het algoritme snel te convergeren in de eerste iteraties waarna convergentie veel trager verliep zoals werd aangetoond in [16]. 10 iteraties bleek voldoende te zijn om reeds een goede clustering te verkrijgen.

Wanneer de verschillende centra bepaald zijn, kunnen we de MFCC-vectoren nu toekennen aan de centra. Dit kunnen we op meerdere manieren doen. Enerzijds is er de eenvoudige *harde* toekenning, waarbij we elke vector toekennen aan juist 1 clustercentrum:

$$f_k(x) = \begin{cases} 1 & \text{als } k = \operatorname{argmin}_j \|c^{(j)} - x\|_2^2 \\ 0 & \text{anders} \end{cases} \quad (4.6)$$

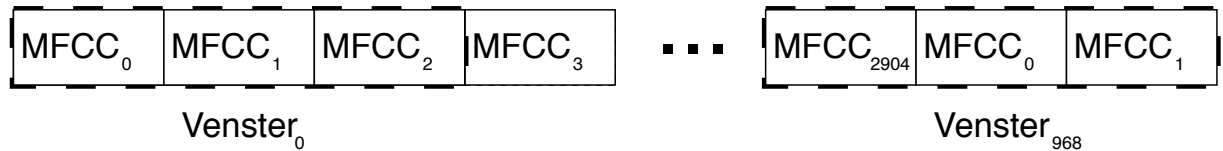
De functie  $f_{hard}(x) : \mathbb{R}^{24} \rightarrow \mathbb{R}^k$  bevat dus  $k - 1$  nulwaarden.

Een andere mapping die in [15] voorgesteld werd, is een *zachte* clustering:

$$f_k(x) = \max \{0, \mu(z) - z_k\} \quad (4.7)$$

waarbij  $z_k = \|x - c^{(k)}\|_2$  en  $\mu(z)$  het gemiddelde is van de elementen van  $z$ . Zachte clustering laat ons toe om iets meer informatie op te slaan dan enkel de dichtsbijzijnde cluster.

Bij zachte clustering is het dus zo dat, afhankelijk van het aantal clustercentra dat gekozen wordt, de dimensie van de geclusterde MFCC-vector kan toenemen. Om niet teveel data te genereren werd vaak een sequentie van  $l$  MFCC-vectoren eerst samengebracht in een *venster* alvorens deze aan te bieden aan het clusteralgoritme. Wanneer er bijvoorbeeld telkens 3 MFCC samen genomen worden, dan daalt het aantal *vensters* per muziknummer van 2905 naar 968. Dit zorgt ervoor dat verdere verwerking veel sneller kon gebeuren.



FIGUUR 4.2: Het samennemen van MFCC-vectoren in vensters

## 4.3 Voorspellen van muziekfactoren

### 4.3.1 Ridge-regression

Vertrekkende vanuit onze geclusterde MFCC-vectoren wordt nu onderzocht of het mogelijk is om de muziekfactoren te voorspellen. Dit wordt gedaan aan de hand van regressie-analyse. Regressie-analyse bestudeert of er een verband bestaat tussen een onafhankelijke variabele en een afhankelijke variabele, waarna de waarde van de onafhankelijke variabele gebruikt kan worden om de afhankelijke variabele te voorspellen:

$$Y = f(X) + U \quad (4.8)$$

$X$  vormt de onafhankelijke variabele en  $Y$  de afhankelijke variabele.  $U$  stelt de storingsterm voor, die onafhankelijk van  $X$  is. De functie  $f$  is onbekend, maar na toepassing van regressie-analyse op een verzameling trainingsvoorbeelden  $\{(x_i, y_i); i = 1, \dots, n\}$  is het mogelijk om  $f$  te karakteriseren aan de hand van enkele parameters.  $X$  zelf kan ook uit meerdere variabelen bestaan. In het bijzonder geval van lineaire regressie kan de functie  $f$  als volgt voorgesteld worden:

$$f(X) = \beta_0 + \sum_{j=1}^d \beta_j X_j \quad (4.9)$$

De onafhankelijke variabelen zijn nu  $X_1, \dots, X_d$ . De onbekende parameter vector  $\beta = (\beta_0, \dots, \beta_d)$  moet geschat worden uit de trainingsvoorbeelden  $\{(x_i, y_i); i = 1, \dots, n\}$ . Een populaire methode maakt gebruik van de kleinste kwadratenmethode, die ervoor zorgt dat de waarden van  $\beta$  bepaald worden door een foutterm te minimaliseren. In het geval van de kleinste kwadratenmethode wordt deze foutterm de residual sum of squares (RSS) genoemd:

$$\begin{aligned} RSS(\beta) &= \sum_{i=1}^n (y_i - f(x_i))^2 \\ &= \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^d \beta_j x_{ij})^2 \end{aligned} \quad (4.10)$$

Deze regressie coëfficiënten zijn optimaal voor de trainingsdata, maar het uiteindelijke doel is om aan de hand van deze regressiecoëfficiënten, nieuwe voorspellingen te genereren. Om het probleem van overfitting op de trainingsdata te vermijden kunnen we gebruik maken van verschillende inkrimpingsmethoden. Ridge regression is een van de bekendste inkrimpingsmethoden waarbij de regressiecoëfficiënten worden gekrompen door een strafterm toe te kennen die gerelateerd is aan de grootte van de coëfficiënten. In het geval van ridge regression wordt een sanctie opgelegd aan de kwadratensom van de regressiecoëfficiënten ( $L_2$ -straf). De RSS ziet er nu als volgt uit:

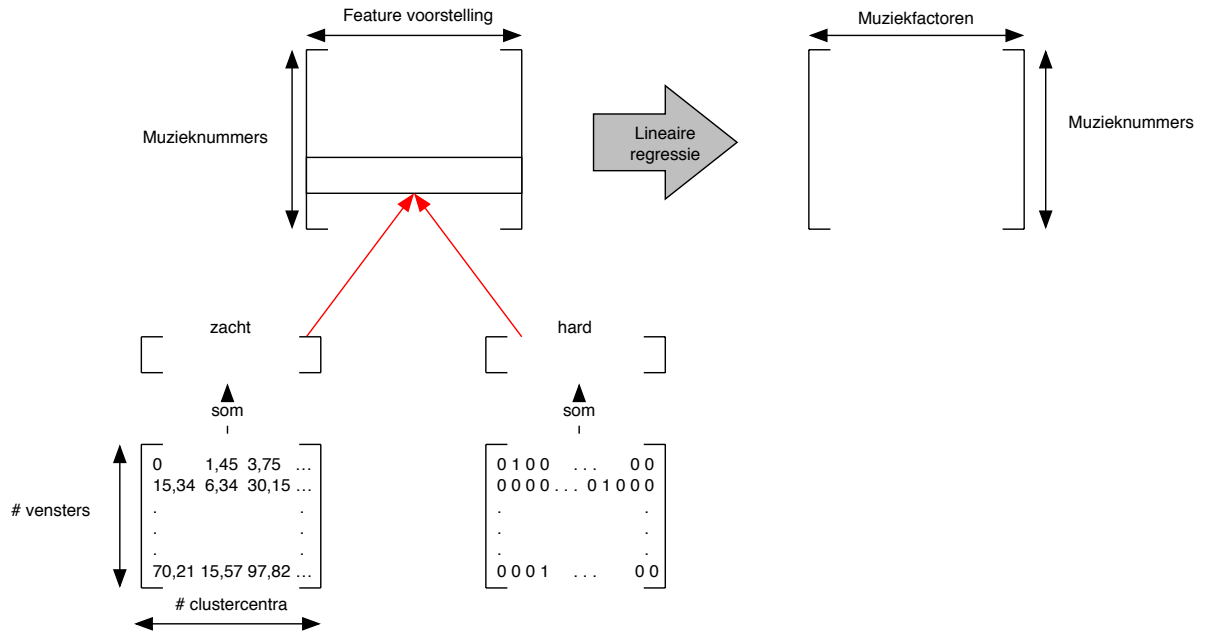
$$\beta_{ridge} = \arg \min_{\beta} \left\{ \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^d \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^d \beta_j^2 \right\} \quad (4.11)$$

De regularisatieparameter  $\lambda$  wordt bepaald aan de hand van kruisvalidatie.

In figuur 4.3 wordt nog eens een overzicht gegeven van het volledige proces dat gepaard gaat met het trainen van een model dat muziekfactoren kan voorspellen uit MFCC-vectoren. De geclusterde MFCC-vectoren van 1 nummer worden via een somfunctie ingevoegd in een designmatrix  $X$ .

### 4.3.2 Resultaten

De resultaten op de testset werden geëvalueerd aan de hand van de gemiddelde kwadratische fout (MSE). Als  $\hat{y}_i$  de voorspelde muziekfactoren van nummer  $i$  zijn en  $y_i$  de corresponderende echte muziekfactoren, dan wordt de MSE gegeven door:



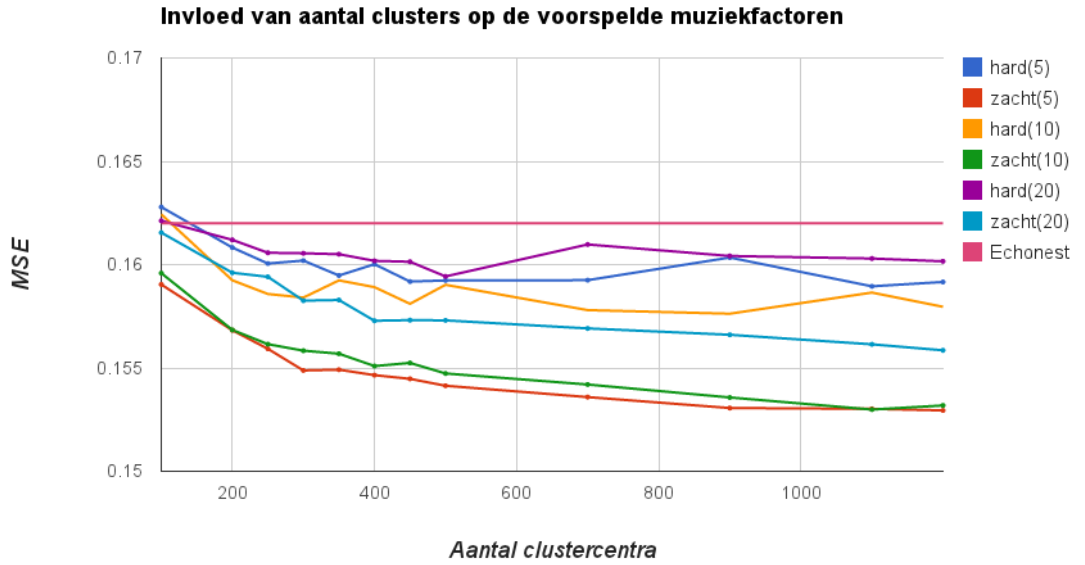
FIGUUR 4.3: overzicht van het trainingsproces van de muziekfactoren uit een geclusterde MFCC-voorstelling

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (4.12)$$

In figuur 4.4 wordt de invloed van het aantal clustercentra weergegeven in functie van de bekomen MSE voor harde en zachte clustering bij verschillende venstergroottes. De MSE die bekomen werd door de verschillende echonestfeatures uit tabel 3.1 te gebruiken werd als referentiewaarde genomen. We kunnen duidelijk zien dat de nauwkeurigheid van onze voorspellingen stijgt indien we kleinere vensters gebruiken en meer clustercentra. Er werd ook gekeken of het combineren van verschillende venstergroottes in een model tot een betere benadering zou leiden. De kleine reductie in MSE woog echter niet op tegen de extra rekenkracht die gepaard ging met het trainen van dit model.

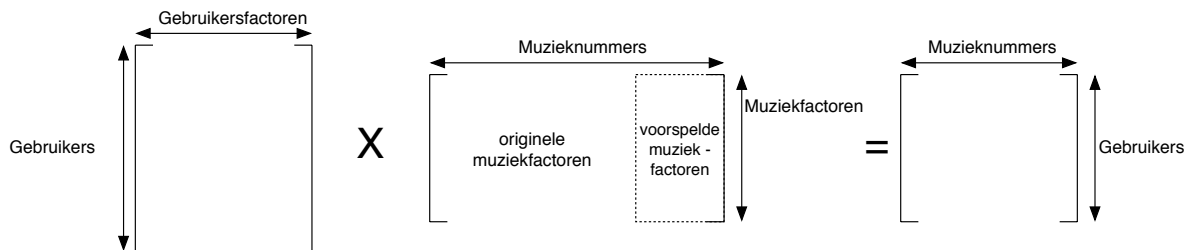
## 4.4 Het genereren van aanbevelingen

Het uiteindelijke doel van het voorspellen van de muziekfactoren is dat ze nauwkeurig genoeg moeten zijn om ook relevante aanbevelingen te maken. Concreet is het de bedoeling om voor nummers waarvoor geen luistergeschiedenis voorhanden is de muziekfactoren te laten voorspellen uit de MFCC-vectoren. Om dit te testen werden de voorspelde muziekfactoren achteraf opnieuw



FIGUUR 4.4: De invloed van het aantal clustercentra op de MSE voor verschillende venster-groottes

ingevoegd in de originele muziekfactormatrix en werden opnieuw aanbevelingen gegenereerd. Om aanbevelingen te genereren nemen we nog steeds het product van de gebruikersfactormatrix en de muziekfactormatrix. Dit product correspondeert met een benadering voor de originele gebruikers-item matrix  $R$ . Voor elke gebruiker  $i$  worden vervolgens de nummers aangeraden volgens dalende score in  $R'$ , op voorwaarde dat de gebruiker nog niet eerder geluisterd heeft naar dat nummer. Bij het genereren van deze aanbevelingen werd vertrokken vanuit zachte clustering van de MFCC-vectoren met 700 clustercentra. De resultaten met deels voorspelde muziekfactoren werden vergeleken met de originele matrixfactorisatiemethode en een willekeurig geval waar de muziekfactoren een onderlinge permutatie ondergingen vooralleer ze opnieuw ingevoegd werden in de muziekfactormatrix.



FIGUUR 4.5: Het aanbevelingsproces met voorspelde muziekfactoren



### 4.4.1 Resultaten

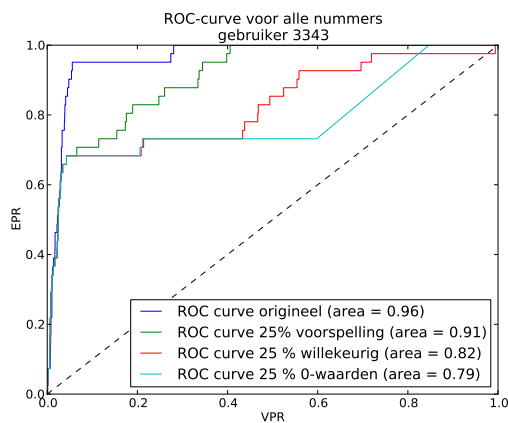
#### 4.4.1.1 Experiment 1

In het eerste experiment wordt de prestatie van een aanbevelingssysteem gebaseerd op een volledig verborgen factormodel vergeleken met een systeem waar een deel van de factoren voorspeld werden en met een semi-willekeurig systeem waar de voorspelde factoren eerst nog een willekeurige permutatie ondergingen vooraleer ze in de muziekfactormatrix werden ingevoegd. Eerst werden de resultaten geëvalueerd volgens de mAP (zie 1.1.3). Er werden telkens 50 aanbevelingen gegenereerd waarna de mAP berekend werd over alle gebruikers. Er werd vastgesteld dat de mAP gevoelig daalde in het geval de muziekfactoren volledig of zelfs deels voorspeld werden. Verder bleken deze resultaten niet significant beter te zijn dan in het geval van willekeurige aanbevelingen. Bovendien was het zo dat de nummers waarvoor de muziekfactoren voorspeld werden veel minder voorkwamen in de top 50 aanbevelingen in vergelijking met het originele aanbevelingsproces. Het vermoeden waarom dit zo is ligt bij het feit dat een regressietechniek elke waarde even goed gaat proberen te benaderen. Muzieknummers waarvan de muziekfactoren veel extreme waarden bevatten hebben een grotere kans om snel aanbevolen te worden. Indien een bepaalde component van de muziekfactor dus een extreme waarde bevat, is het veel belangrijker om die correct te kunnen voorspellen dan het geval waar die waarde rond de gemiddelde waarde voor die component schommelt.

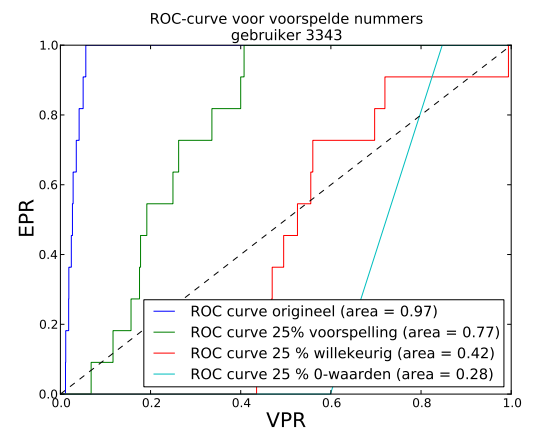
Een alternatieve metriek die werd voorgesteld in hoofdstuk 1 is de ROC-curve en de bijhorende AUC. In figuren 4.6 en 4.7 wordt de ROC-curve weergegeven wanneer alle nummers in de testset in beschouwing worden genomen en wanneer enkel gekeken wordt naar de nummers waarvan de muziekfactoren voorspeld werden en zich in de testset bevonden.

Wanneer we de AUC berekenen voor elke gebruiker en vervolgens uitmiddelen over alle gebruikers bekomen we de gemiddelde AUC voor het totale aanbevelingsproces. Zoals in figuur 4.8 te zien is, daalt de gemiddelde AUC sterk wanneer we een deel van de muziekfactoren gaan voorspellen. De voorspelling is in alle mogelijke gevallen echter nog steeds beter dan indien de factoren achterna willekeurig gepermuteerd worden en is bovendien nog steeds hoger dan 0.5, de verwachte waarde wanneer de nummers in totaal willekeurige volgorde aangeraden zouden worden.

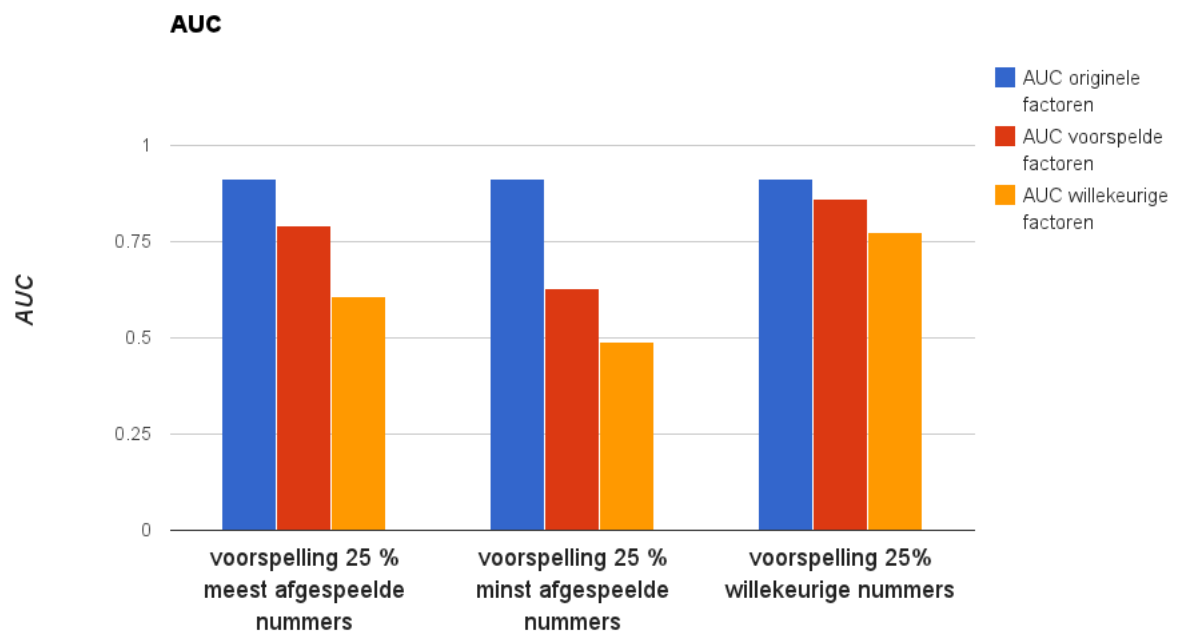
Uit figuren 4.8 en 4.9 kan dus worden afgeleid dat onze voorspellingsmethode voor de muziekfactoren wel degelijk zijn nut kan hebben indien er absoluut geen luistergeschiedenis voorhanden



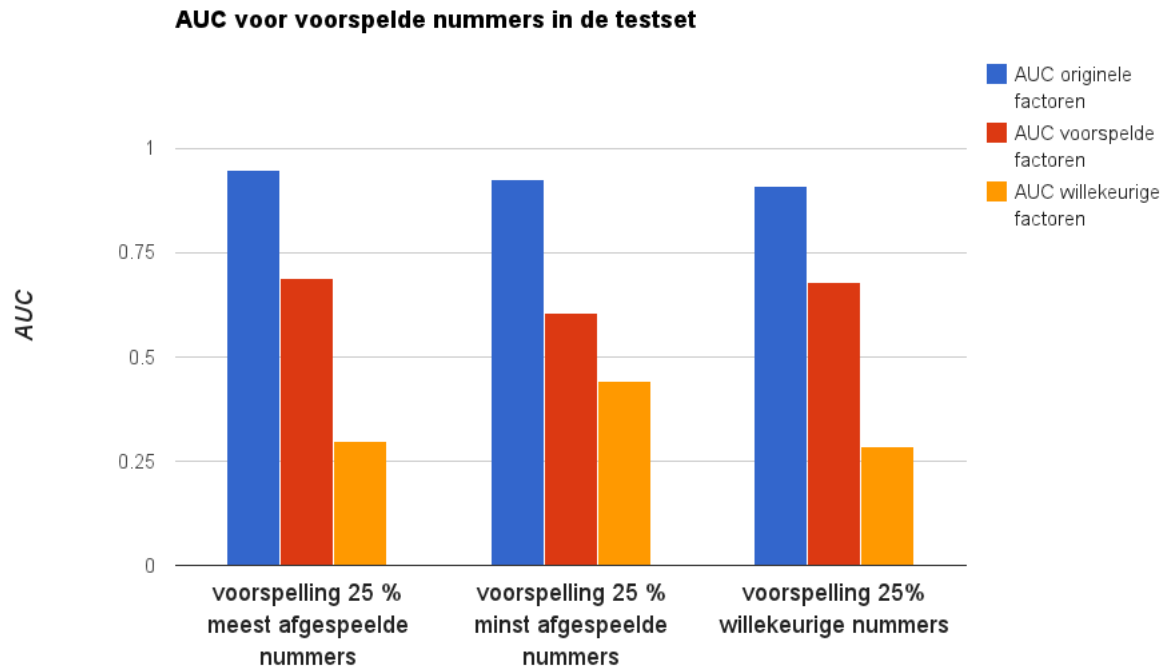
FIGUUR 4.6: ROC-curve voor alle nummers in de testset voor een bepaalde gebruiker



FIGUUR 4.7: ROC-curve voor voorspelde nummers in de testset voor een bepaalde gebruiker



FIGUUR 4.8: gemiddelde AUC waarden voor alle nummers



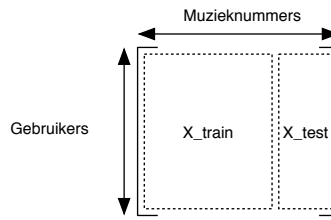
FIGUUR 4.9: gemiddelde AUC waarden voor voorspelde nummers

is. In een volgend experiment wordt er onderzocht of er een punt is voor de hoeveelheid luistergeschiedenis waar het voorspellen van de muziekfactoren uit de audio tot betere aanbevelingen zal leiden dan wanneer aanbevelingen worden gegenereerd enkel op basis van een collaboratieve filtertechniek.

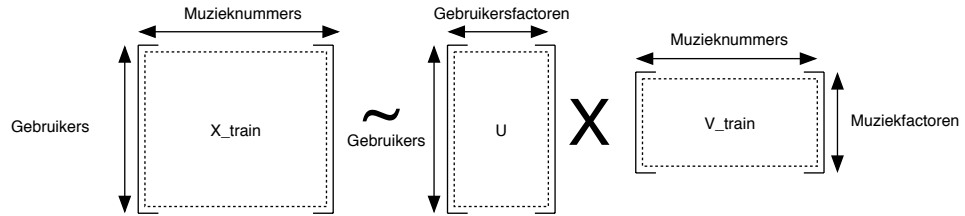
#### 4.4.1.2 Experiment 2

Er wordt opnieuw vertrokken van de initiële dataset (zie figuur 4.10). Daarna wordt een verbor-gen factormodel getraind op de trainingset zoals beschreven staat in hoofdstuk 2. Dit resulteert in een muziekfactormatrix  $V_{train}$  en een gebruikerfactormatrix  $U$  (zie figuur 4.11). Deze gebruikerfactormatrix blijft steeds dezelfde gedurende het vervolg van het experiment.

De muziekfactormatrix  $V_{train}$  wordt nu gebruikt om aan de hand van de geclusterde voorstel-ling van de MFCC-vectoren een lineair model te trainen dat in staat is om de muziekfactoren te voorspellen voor de muzieknummers die zich in de testset  $X_{test}$  bevinden. De uit de audio voorspelde muziekfactoren noemen we  $V_{test\_audio}$ .



FIGUUR 4.10: Opdeling van de data in training en testset



FIGUUR 4.11: Opdeling van de data in training en testset

Nu vertrekken we vanuit  $X_{test}$  en gaan we telkens informatie over de luistergeschiedenis verwijderen. Voor elk nummer wordt een percentage van de gebruikers die naar het nummer geluisterd hebben verwijderd. Dit percentage wordt bepaald door de parameter  $\alpha$ . Concreet betekent dit dat in de gebruiker-item matrix  $X_{test}$  het aantal luisterbeurten op 0 gezet wordt. Dit zorgt ervoor dat de gebruiker-item matrix opnieuw in een nieuwe training- en testset opgesplitst wordt. Het deel van de informatie dat wordt verwijderd noemen we  $X_{test\_A}$  en is de nieuwe testset. Deze matrix wordt omgezet naar een binaire gebruikers-item matrix omdat enkel zal gekeken worden of de voorspellingen effectief ooit al eens beluisterd werden door de gebruiker. Het aantal luisterbeurten doet er niet echt toe. Het andere deel  $X_{test\_B}$  zal opnieuw gebruikt worden om een nieuwe muziekfactormatrix  $V_{test\_mf}$  te bouwen met behulp van de gebruikerfactormatrix  $U$  en 1 iteratie uit het verborgen factormodel. In deze iteratie worden de gebruikersfactoren constant beschouwd en zullen de muziekfactoren geoptimaliseerd worden in functie van de gebruikersfactoren en de scores (zie vgl 2.4). Doordat deze matrixfactorisatie voor grote waarden van  $\alpha$  zeer weinig informatie heeft, wordt verwacht dat de resulterende muziekfactoren vrij slecht zullen zijn. Wanneer dan voorspellingen gegenereerd worden en deze geëvalueerd worden op  $X_{test\_A}$  moeten de resultaten vrij slecht zijn en wordt er hopelijk een waarde voor  $\alpha$  gevonden waarbij het beter is om de muziekfactoren te laten voorspellen uit de audio, dan op basis van de beperkte luistergeschiedenis. De oorspronkelijke matrix  $X_{test}$  bevatte 264.767 niet-nulwaarden. Wanneer  $\alpha = 0.5$  dan bevat  $X_{test\_B}$  dus ongeveer 132.000 niet-nulwaarden. De voorspellingen of aanbevelingen worden uiteindelijk als volgt bekomen:

$$\begin{bmatrix} 0 & 0 & 6 & 0 & 5 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 40 & 0 & 13 & 0 & 1 & 11 & 3 & 6 \\ 0 & 0 & 0 & 0 & 4 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 10 & 0 \\ 1 & 2 & 0 & 3 & 5 & 20 & 0 & 0 & 0 & 0 \\ 2 & 1 & 2 & 0 & 0 & 1 & 0 & 6 & 0 & 5 \\ 0 & 3 & 1 & 0 & 1 & 0 & 1 & 3 & 6 & 0 \\ 0 & 0 & 0 & 6 & 2 & 1 & 0 & 0 & 3 & 0 \\ 1 & 0 & 6 & 7 & 0 & 2 & 9 & 0 & 0 & 7 \\ 10 & 5 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 6 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 5 & 20 & 0 & 0 & 0 & 0 \\ 2 & 1 & 2 & 0 & 0 & 1 & 0 & 6 & 0 & 5 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 3 & 6 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 2 & 0 & 0 & 0 & 7 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (4.15)$$

FIGUUR 4.12: Een voorbeeld van hoe de dataset  $X_{test}$  wordt opgesplitst in  $X_{test\_A}$  en  $X_{test\_B}$  wanneer  $\alpha = 0.5$

$$pred\_audio = U * (V_{test\_audio})^T \quad (4.13)$$

$$pred\_mf = U * (V_{test\_mf})^T \quad (4.14)$$

Een andere vraag die men kan stellen is of een combinatie van een sociaal-gebaseerde techniek en een inhoudsgebaseerde techniek in bepaalde situaties kan leiden tot een beter aanbevelingssysteem. Een eenvoudige manier om dit na te gaan bestaat erin om de scores uit 4.13 en 4.14 op te tellen en vervolgens de resulterende scores gebruiken om aanbevelingen te genereren. Deze techniek zal in het bijzonder goed werken indien de 2 aanbevelingstechnieken complementair zijn. Dit betekent dat de inhoudsgebaseerde techniek erin slaagt om voorspellingen te maken die de sociaalgebaseerde techniek niet doet en vice versa:

$$pred\_hybrid = pred\_audio + pred\_mf \quad (4.16)$$

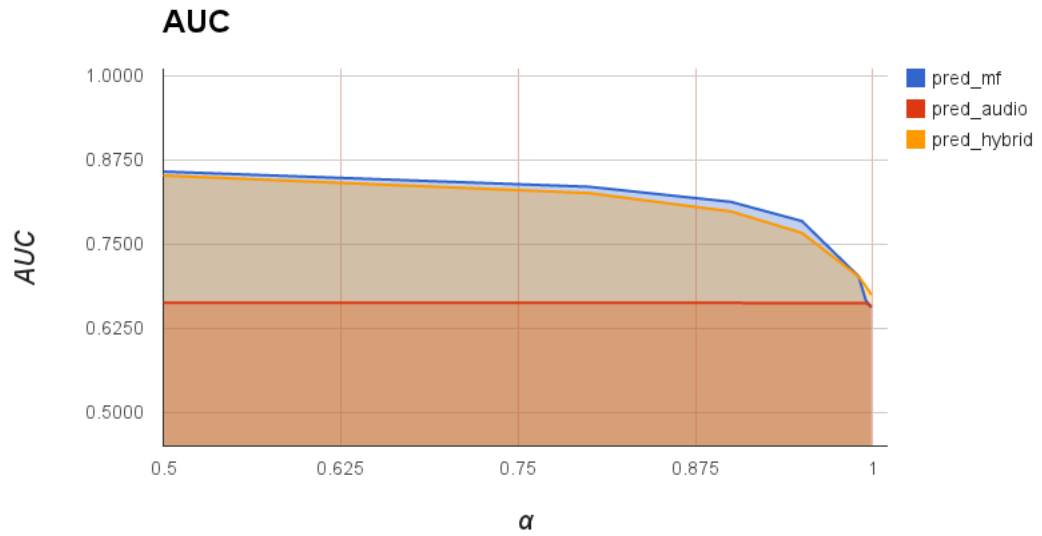
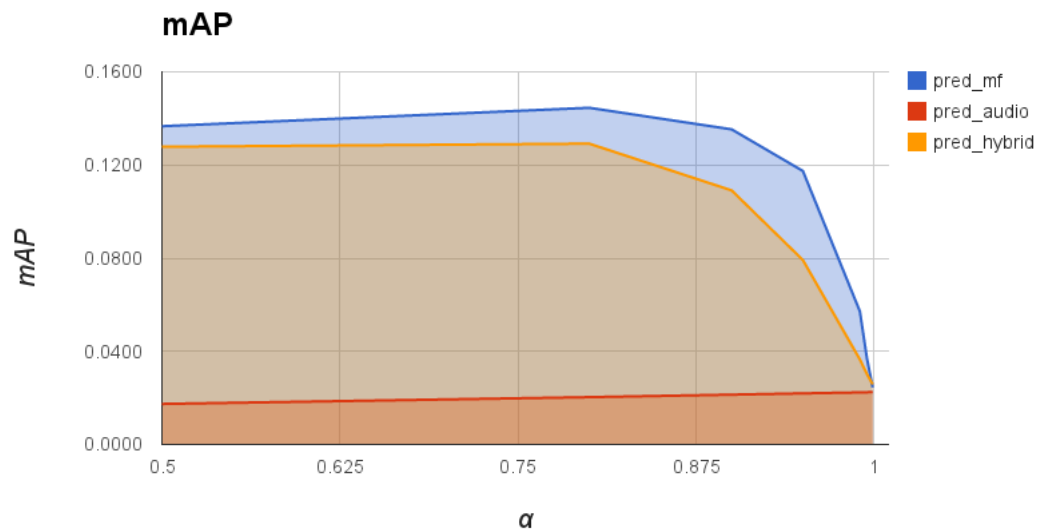
De resultaten werden bekomen door dit experiment 3 maal uit te voeren met telkens een verschillende  $X_{test\_A}$  en  $X_{test\_B}$  en vervolgens de resultaten uit te middelen.

TABEL 4.1: AUC voor de verschillende technieken bij experiment 2

	$\alpha$						
	0.5	0.8	0.9	0.95	0.99	0.995	0.999
pred_mf	0.8580	0.8354	0.8129	0.7844	0.7022	0.6669	0.6451
pred_audio	0.6630	0.6629	0.6628	0.6625	0.6626	0.6625	0.6580
pred_hybrid	0.8520	0.8259	0.7987	0.7668	0.7019	<b>0.6877</b>	0.6808

TABEL 4.2: mAP voor de verschillende technieken bij experiment 2

	$\alpha$						
	0.5	0.8	0.9	0.95	0.99	0.995	0.999
pred_mf	0.1367	0.1446	0.1353	0.1174	0.0572	0.0368	0.0228
pred_audio	0.0174	0.0203	0.0214	0.0219	0.0224	0.0224	0.0226
pred_hybrid	0.1278	0.1292	0.1091	0.0792	0.0365	0.0302	<b>0.0277</b>

FIGUUR 4.13: AUC voor verschillende waarden van  $\alpha$  bij experiment 2FIGUUR 4.14: mAP voor verschillende waarden van  $\alpha$  bij experiment 2

Zoals verwacht blijkt matrixfactorisatie superieur te zijn ten opzichte van voorspelling van de

muziekfactoren op basis van de MFCC-vectoren. Enkel wanneer zeer weinig luisterinformatie beschikbaar is, presteert het inhoudsgebaseerde aanbevelingssysteem op basis van MFCC-vectoren beter. Er werd een evenwichtspunt gevonden wanneer  $\alpha = 0.999$  was. In deze situatie bevatte de trainingset nog slechts een kleine 2000 niet-nulwaarden. Het gebruik van een hybride aanbevelingssysteem blijkt echter sneller de prestatie van de sociaalgebaseerde techniek te evenaren en zelfs te verbeteren wanneer weinig luistergeschiedenis voorhanden is.

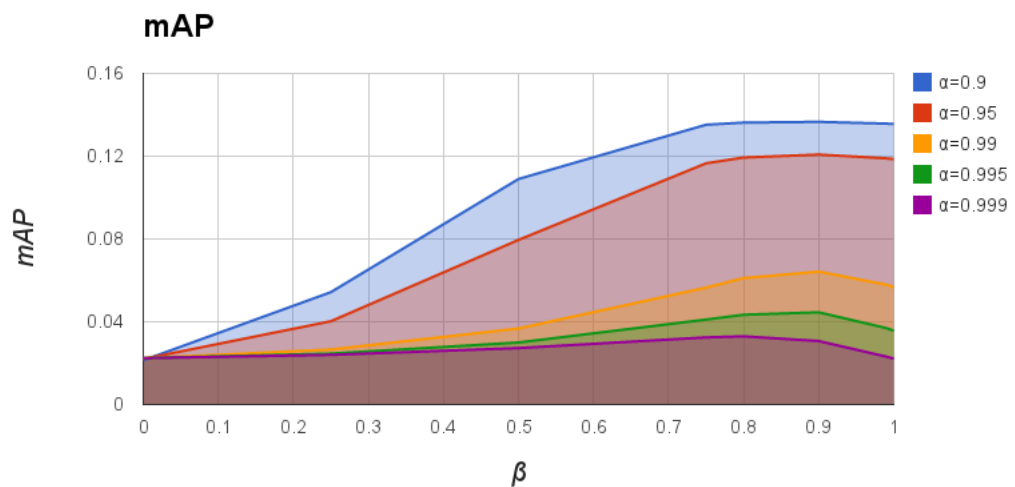
#### 4.4.1.3 Experiment 3

We kunnen tenslotte het belang van elke techniek in het hybride aanbevelingssysteem wijzigen door een extra parameter  $\beta$  in te voeren. Vergelijking 4.16 wordt dan uitgebreid tot:

$$pred_{hybrid} = (1 - \beta) * pred_{audio} + (\beta) * pred_{mf} \quad \beta \in [0, 1] \quad (4.17)$$

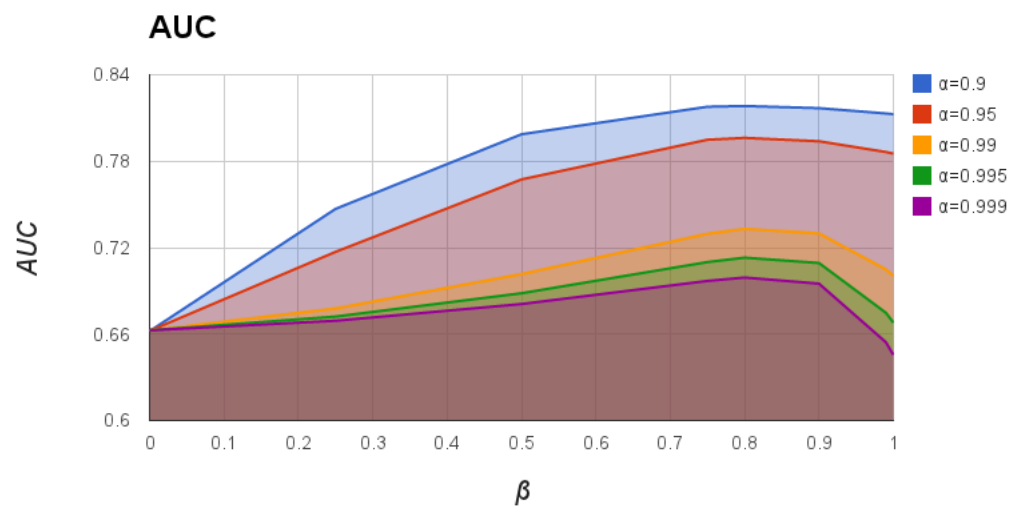
$\beta = 0$  betekent dat we enkel de inhoudsgebaseerde techniek in beschouwing nemen terwijl  $\beta = 1$  enkel rekening houdt met de sociaalgebaseerde techniek. We kunnen nu opnieuw voor elke waarde van  $\alpha$  deze vergelijking optimaliseren voor verschillende waarden van  $\beta$ .

Opnieuw werd dit experiment meerdere malen uitgevoerd op een verschillende training- en testset. de resultaten zijn gevisualiseerd in figuren 4.15 en 4.16.



FIGUUR 4.15: mAP voor verschillende waarden van  $\beta$

We kunnen vaststellen dat voor verschillende waarden van  $\alpha$  het optimum voor  $\beta$  rond 0.8-0.9 ligt. Dit zorgt ervoor dat bij het aanbevelingssysteem de sociaalgebaseerde techniek op basis van

FIGUUR 4.16: AUC voor verschillende waarden van  $\beta$ 

matrixfactorisatie zwaar doorweegt, maar dat men door in mindere mate de inhoudsgebaseerde scores mee te nemen in de uiteindelijke predictie tot een beter aanbevelingssysteem kan komen.



## Hoofdstuk 5

### Besluit

In deze masterproef werd onderzocht of het mogelijk is om het *koude startprobleem* deels te verhelpen door een collaboratieve filtertechniek te gebruiken die toelaat om data deels te voorspellen op basis van akoestische informatie in de plaats van luistergeschiedenis. Met behulp van verborgen factormodellen kan een score voor een gebruiker en een muziknummer geschreven worden als het product van een muziekfactor en een gebruikerfactor. Voor nieuwe nummers zijn de muziekfactoren nog onbekend en kunnen deze aan de hand van geclusterde MFCC-vectoren met behulp van regressie-analyse geschat worden. De resultaten toonden aan dat deze aanpak beter werkt dan willekeurige muziekfactoren, maar deze benaderingen scoorden beduidend slechter dan wanneer de correcte muziekfactoren gebruikt worden. Wanneer deze inhoudsgebaseerde techniek echter correct gecombineerd wordt met de sociaalgebaseerde techniek kan het aanbevelingssysteem in vele gevallen reeds betere aanbevelingen genereren.

Een mogelijke oplossing zou zijn om naast MFCC-vectoren die enkel akoestische informatie opnemen, heel wat externe informatie over het nummer ook in beschouwing te nemen. Meta-informatie zoals de tijdsgeest van het nummer of de taal waarin gezongen wordt draagt wellicht ook in belangrijke mate bij tot vorming van de verschillende muziekfactoren. Bij nieuwe muziknummers is deze informatie echter zelden beschikbaar. Ook een verdere akoestische analyse waarbij zaken zoals ritme en tempo ook in beschouwing worden genomen kan de voorspelling nauwkeuriger maken.

Verder kan ook gekeken worden of het mogelijk is om de muziekfactoren beter te kunnen voorspellen met andere technieken dan lineaire regressie. De kostfunctie zou ook aangepast kunnen

worden zodat de nadruk meer ligt op het correct voorspellen van extreme waarden voor muziekfactoren.

We kunnen besluiten dat het voorspellen van muziekfactoren een complex probleem is en dat deze voorspelling nuttig kan zijn wanneer absoluut geen luisterinformatie voorhanden is, maar dat collaboratieve filtertechnieken op basis van matrixfactorisatie al snel superieur zijn indien er een beperkte luistergeschiedenis beschikbaar is.

# Bijlage A

## Dataset

### A.1 Million Song dataset

In deze masterproef werd vertrokken van een subset van een dataset die toegankelijk werd gesteld in het kader van de Million Song Dataset Challenge [6]. Deze subset bestaat uit 10000 muzieknummers en 20000 gebruikers die al dan niet geluisterd hadden naar een muzieknummer. Een record uit de dataset bestaat uit een gebruikersid, een muzieknummerid en een getal dat correspondeert met het aantal afspeelbeurten. Elk muzieknummerid kan gelinkt kan worden aan offline metadata die beschikbaar werd gesteld door The Echo Nest. Deze metadata bestond o.a. uit titel, nationaliteit van de artiest, akoestische informatie, en een verwijzing naar een kort audiofragment van het nummer. Deze audiofragmenten werden gebruikt als basis om de MFCC-vectoren te berekenen.

### A.2 MusiXmatch dataset

De musixmatch dataset is een verzameling van songteksten in bag-of-words formaat. Eerst volgt een rij van topwoorden in dalende volgorde van populariteit. Elke volgende regel uit de dataset bestaat uit het id van de MSD, een id van musixmatch gevolgd door een lijst koppels van de vorm `idx:cnt`. Dit correspondeert met het aantal voorkomens van het woord met index `idx` in de lijst van topwoorden.

10cbcd627472477dfbec90fb75017f8df6ce84ec	SOGPLBE12A58A80442	1
10cbcd627472477dfbec90fb75017f8df6ce84ec	S0BWSGV12AB018B5E0	1
10cbcd627472477dfbec90fb75017f8df6ce84ec	SOWINI12AB018CC51	1
10cbcd627472477dfbec90fb75017f8df6ce84ec	S0XHVRT12A81C2320D	1
e9dc6b4c2b22aa6dc8260e1963021567728055b2	S0UIGCD12AB0186713	1
e9dc6b4c2b22aa6dc8260e1963021567728055b2	S0YOMRA12A6D4F9975	11
e9dc6b4c2b22aa6dc8260e1963021567728055b2	S0LMIUU12A58A79C99	3
e9dc6b4c2b22aa6dc8260e1963021567728055b2	S0M0BBZ12A8C144831	8
e9dc6b4c2b22aa6dc8260e1963021567728055b2	S0PXYDY12A6D4FA876	6
30e4a688e6fc9c8bfe55998af3996a909ae34449	S0TQBKC12A8C13E961	1
30e4a688e6fc9c8bfe55998af3996a909ae34449	S0AIBYI12AB0185C5B	1
30e4a688e6fc9c8bfe55998af3996a909ae34449	S0SMONK12A8C139059	1
30e4a688e6fc9c8bfe55998af3996a909ae34449	S0MYDVM12A6D4F7935	1
30e4a688e6fc9c8bfe55998af3996a909ae34449	S0HXBKV12A6D4F820D	2
30e4a688e6fc9c8bfe55998af3996a909ae34449	S0DZ0ER12A8C1360FB	1
30e4a688e6fc9c8bfe55998af3996a909ae34449	S0QGYDJ12A6310F359	1
18ce1da0e1017e31baaa5f80afa64ee3c7fab379	S0KWSEA12A8C141C9D	1
18ce1da0e1017e31baaa5f80afa64ee3c7fab379	S0PGCXT12A8C138AD1	5
18ce1da0e1017e31baaa5f80afa64ee3c7fab379	S0DDDDWP12AB0188D17	1
18ce1da0e1017e31baaa5f80afa64ee3c7fab379	S0TVJCB12A8C136E46	1
18ce1da0e1017e31baaa5f80afa64ee3c7fab379	S0UFTBI12A8C183F65	6
18ce1da0e1017e31baaa5f80afa64ee3c7fab379	S0SXLTC12AF72A7F54	9
18ce1da0e1017e31baaa5f80afa64ee3c7fab379	S0SNJIT12A8159E8DB	2
18ce1da0e1017e31baaa5f80afa64ee3c7fab379	S0KENKR12AB01828F7	7
0b254c684efb08fd04933add2d1e4191d2a87bac	S0OPUTL12A8C143381	1
0b254c684efb08fd04933add2d1e4191d2a87bac	S0P0ANU12A8AE48C9B	2
0b254c684efb08fd04933add2d1e4191d2a87bac	S0UDEUC12A6D4F95A7	1
0b254c684efb08fd04933add2d1e4191d2a87bac	S0EB0WM12AB017F279	1

FIGUUR A.1: Een voorbeeldfragment uit de dataset. Rechts bevindt zich het gebruikers id, in het midden het songid en rechts het aantal afspeelbeurten.

```

TRAAEDY12903CBFA7,4193161,1:5,23:11,24:4,42:6,72:1,90:7,156:1,190:1,229:1,243:4,251:1,474:7,482:12,530:1,544:2,553:1,603:1,632:
15,650:15,695:1,710:8,826:2,978:12,1022:1,1246:2,1276:1,1328:2,1349:6,1380:1,1452:30,1641:10,1673:1,2105:1,2414:19,2533:1,2700:2
,2747:12,2806:2,2825:1,2843:4,3055:1,3178:1,3271:1,3359:2,3665:1,3989:5,4189:2,4199:1,4213:1,4502:2,4633:2,4684:5,4815:2
TRAAEEQ128F42180B2,3561951,47:1,408:1,605:1,1057:1
TRAAEHW128F9344FD3,5967283,1:5,2:7,5:3,6:2,8:3,10:2,11:5,12:8,13:1,18:1,19:2,22:1,23:2,24:6,28:1,30:2,32:1,39:2,40:1,43:6,51:1,5
5:3,67:1,75:6,82:1,87:1,95:6,125:1,133:1,146:2,163:1,180:1,195:1,283:1,291:1,337:1,338:1,403:1,455:1,491:1,497:1,503:2,535:1,548
:1,569:6,584:1,596:2,676:1,753:1,784:1,877:1,1035:1,1097:1,1114:1,1223:1,1525:1,1888:1,2069:1,2614:1,2774:1,3175:6,4089:1,4525:1
TRAAEJH128E0785506,1018402,1:9,2:12,3:2,4:3,5:1,7:1,9:2,11:8,16:1,17:4,20:4,21:1,25:2,26:1,29:1,30:1,31:2,32:10,34:1,35:1,37:1,4
4:1,51:1,53:1,55:2,57:1,59:1,61:1,62:1,67:1,72:2,73:3,74:2,89:1,108:1,113:1,120:1,123:1,153:2,154:3,155:1,164:1,166:10,173:1,199
:1,229:1,240:1,245:1,254:1,275:1,303:1,310:1,324:1,337:1,347:2,352:11,444:2,497:1,536:1,602:3,635:1,668:1,914:1,3314:8
TRAAEJV128F423CF04,1861819,1:11,3:19,6:3,7:1,10:1,12:7,13:1,17:1,22:1,23:1,25:2,26:1,30:3,33:9,36:3,43:3,56:7,61:1,68:1,82:3,84:
2,85:1,88:1,89:1,103:1,133:1,186:1,211:1,247:1,264:3,360:1,419:1,499:8,617:3,680:1,2766:1

```

FIGUUR A.2: Een voorbeeldfragment uit de mxm dataset.

### A.3 Last.fm dataset

De last.fm dataset is een dataset die tag-informatie van muzieknummers en similariteit tussen verschillende muzieknummers bevat. Deze informatie kan ook online opgevraagd worden met behulp van de Last.fm API wanneer de methoden 'getTopTags' en 'getSimilar' gebruikt worden. De dataset bestaat uit een verzameling van json-gecodeerde tekstbestanden. Elk muzieknummer correspondeert met 1 bestand. Deze dataset is gelinkt aan de MSD via het attribuut 'track\_id'.

```

{"artist": "The Bar-Kays",
"timestamp": "2011-08-03 15:26:43.246805",
"similars": [],
"tags": [{"soul", "100"}, {"funk", "40"}, {"60s", "20"}, {"Rhytm and Blues", "20"}, {"Saturday Night",
"20"}, {"REAL soul music", "20"}, {"rock top", "20"}, {"rock top funky soul", "20"}, {"title is a full
sentence", "20"}, {"groove", "0"}, {"you cant sit down lp version", "0"}],
"track_id": "TR7OKHS128F147ED44", "title": "You Can't Sit Down (LP Version)"}

```

FIGUUR A.3: Een voorbeeldbestand uit de last.fm dataset.

# Bibliografie

- [1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.99. URL <http://dx.doi.org/10.1109/TKDE.2005.99>.
- [2] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009. ISSN 1687-7470. doi: 10.1155/2009/421425. URL <http://dx.doi.org/10.1155/2009/421425>.
- [3] Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006. ISBN 1401302378.
- [4] Fabio Aioli. A preliminary study on a recommender system for the million songs dataset challenge. 2012.
- [5] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, ICDM '08*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3502-9. doi: 10.1109/ICDM.2008.22. URL <http://dx.doi.org/10.1109/ICDM.2008.22>.
- [6] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.
- [7] Mark Levy and Mark Sandler. A semantic space for music derived from social tags. In *ISMIR*, pages 411–416, 2007.

- 
- [8] Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press, 1966. URL <http://www.webuse.umd.edu:9090/>.
  - [9] F. Å. Nielsen. Afinn, mar 2011. URL <http://www2.imm.dtu.dk/pubdb/p.php?6010>.
  - [10] C. Laurier. *Automatic Classification of Musical Mood by Content-Based Analysis*. PhD thesis, Universitat Pompeu Fabra, 10/2011 2011. URL [files/publications/PhD\\_Cyril\\_Laurier\\_2011\\_Music\\_Mood\\_Classification.pdf](http://files/publications/PhD_Cyril_Laurier_2011_Music_Mood_Classification.pdf).
  - [11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
  - [12] Kerstin Bischoff, Claudiu S. Firan, Raluca Paiu, Wolfgang Nejdl, Cyril Laurier, and Mohamed Sordo. Music mood and theme classification - a hybrid approach. In *ISMIR*, pages 657–662, 2009.
  - [13] S. Davis and P. Mermelstein. Experiments in syllable-based recognition of continuous speech. *IEEE Trans. Acoust., Speech, Signal Processing*, 28:357 – 366, Aug. 1980.
  - [14] Uwe Ligges. *tuneR: Analysis of music*, 2011. URL <http://r-forge.r-project.org/projects/tuner/>.
  - [15] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research - Proceedings Track*, 15:215–223, 2011. URL <http://dblp.uni-trier.de/db/journals/jmlr/jmlrp15.html#CoatesNL11>.
  - [16] Adam Coates and Andrew Y. Ng. Learning feature representations with k-means.