ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group

Semester Thesis

History-Based Collaborative Filtering for Music Recommendation

Fabian Reichlin

April 2008

Dept. of Information Technology and Electrical Engineering
Swiss Federal Institute of Technology Zurich

Prof. Dr. Roger Wattenhofer
Distributed Computing Group

Advisors: Michael Kuhn, Olga Goussevskaia

Abstract

In this thesis we present history-based collaborative filtering, a novel approach to recommend unfamiliar music to users which them is nevertheless going to suit, in order to broaden theirs horizon of musical familiarity. We refer to people with similar music taste which experienced musical transitions when generating a new recommendation, and show that the results are accurate in terms of musical direction.

Our works builds upon a previously developed map of music, a space of songs with approximately 430'000 vertices. We derive the musical transitions from extracted listening data by analysing way points in the map of music, and by explicitly looking for transitions in users' listening behaviour. The profiles with most significant musical taste transitions are used as references for producing new recommendations. We evaluate our data set with two evaluation measures.

Contents

1 Introduction **1**
- 1.1 Related Work . . . 2
- 1.2 Contributions . . . 3
- 1.3 Organization . . . 4

2 History-Based Collaborative Filtering **5**
- 2.1 History-Based Collaborative Filtering . . . 5
  - 2.1.1 Analysing User Profiles and Data Crawling . . . 6
  - 2.1.2 Distance Measures in User Profiles . . . 7
  - 2.1.3 Clustering of Users' History . . . 8
  - 2.1.4 Data Refinement . . . 11
  - 2.1.5 Aggregation of Transition Vectors . . . 12
- 2.2 Expert Recommendation . . . 15
- 2.3 Combined Recommendation Algorithm . . . 15

3 Evaluation **17**
- 3.1 General Data Analysis . . . 17
  - 3.1.1 Clustering Process . . . 17
  - 3.1.2 Refinement Process . . . 18
  - 3.1.3 Song Point and Transition Vector Properties . . . 19
- 3.2 Evaluation Measures . . . 21
  - 3.2.1 Transition Vector Angles . . . 22
  - 3.2.2 Song Count Approach . . . 23
- 3.3 Results . . . 27
  - 3.3.1 Transition Vector Angles . . . 27
  - 3.3.2 Song Count Approach . . . 28
- 3.4 Discussion . . . 31

4 Conclusion **33**
- 4.1 Future Work . . . 33

Chapter 1

Introduction

Digitalisation of music libraries has rapidly increased in the last five years. With improved performance of home computers, easier-to-use media players and – most importantly – cheaper and faster internet access, people not only tend to listen to their music at their home computers, but also to purchase songs and records through webstores, such as Amazon.com[1] or iTunes[2]. This allows sellers to precisely capture which songs were bought, how often certain artists were played and which tracks fell out of favor. The statistics of a user become more powerful the more electronic purchases are done, and the more music is played through media players. Researchers and traders seize the opportunity to exploit the huge amount of available user data and to use the findings for their respective interests. The most prominent application of such information retrieval research are recommender systems. In many e-commerce applications, the user is greeted with a section of recommendations tailored for their profile when opening respective start pages. Thereby, products which could best suit the users based on the purchases done in the past, and as an inspiration for making new acquisitions, are presented. Not to neglect, a significant amout of research comes from non-profit organisations, where movies, music and even jokes are recommended [1].

Despite the glut of available music recommendation systems, there is still evidence that people still rather prefer the opinion of a long time music expert to the output of an obscure computer algorithm. One example is the successful Album Club[3], a popular subscription service for music. The company selects a fixed number of albums each month, and sends them home to the customers. Their staff members handpick the latest records on the market, and the customer is guaranteed to receive "the best and most interesting music of today" and to get records "different from, and most often ahead of, the mainstream". Wouldn't it be great to mime such an expert by an algorithm of our own, and to cash on the money that had gone to the subscription

[1] See `http://www.amazon.com/`.
[2] See `http://www.apple.com/itunes/`.
[3] See `http://www.thealbumclub.com/`.

service instead? We tried to find a solution to this challange, such that we could produce a real profitable algorithm.

Another cutback of current recommendation systems – despite all economical considerations – is their limit in *newness*. They might produce good recommendations of products the user is going to like, but most of the cases, recommendations are still close to circles the user is familiar with. A user might be tired of the music he is listening to, but nevertheless does not know what else he should listen to, because recommendation results would still be similar to the songs he is tired of. Also, wouldn't it be interesting to get really *new* recommendations, of products the user has never heard of before, of music that is different from genres he knows about, but nevertheless is going to like? We think this topic has no been sufficiently addressed by current recommender systems, and believe there must be ways to bypass this shortage, as people generally change their taste in music over a longer time span. Finding and evaluating such ways was the major part of this work.

We present a novel recommendation algorithm which analyses other users' history and explicitely looks for transitions in their taste of music. We store those transitions in a database and refer to them when a new recommendation is generated by taking their region of interest *after* the transition as a reference region to which new recommendations should point to.

1.1 Related Work

Recommender systems are a relatively young research topic, where first publications were not made until the mid-1990s [2]. In this time, the increasing popularity of the internet and its e-commerce applications such as prime example Amazon.com gave research in recommender systems a sharp boost. Generally speaking, it is the goal to make recommendations as precise as possible, in the sense that the user is really going to like the recommended product, and – once he is introduced to the new product – to pinch more money from the customer by selling new items.

Basically, current recommender systems can be divided into three main categories: content-based, collaborative, and hybrid recommendation approaches [2]. Content-based recommendations refer to the user himself by analysing features and extracting metadata of his items. Those features are used to produce recommendations of items similar to the ones he preferred in the past. Collaborative recommender systems (or collaborative filtering systems) on the other hand relate to people with profiles similar to the user, where items which those people liked are recommended. Hybrid approaches finally combine elements from content-based and collaborative recommender systems.

A subset of collaborative filtering systems are item-to-item collaborative systems. These algorithms focus on finding items that customers tend to purchase together rather than extracting item features. The purchase lists are used to compute item-to-item similarities and hence to obtain recommendations, as opposed to user-based algorithms, where similarities between users are analysed. A famous real-world exam-

ple of item-to-item collaborative filtering is the recommendation algorithm used by Amazon.com [3]. The algorithm's scalability and fast reaction to changes in user data were crucial for applying item-to-item collaborative filtering in a web store of that extent. Sarwar et al. [4] showed that item-to-item collaborative algorithms provide dramatically better performance than user-based algorithms, while at the same time providing better quality than the best available user-based algorithms.

Applied to the context of music recommendation, several approaches to identify music similarity have been proposed. Ellis et al. [5] investigated inter alia user collection data from a music sharing service to derive artist similarity. These distance judgments were then converted into a set of points in a 3D Euclidean space to permit visualisation of the dataset's geometric configuration. [6] on the other hand analysed Mel-frequency cepstral coefficients of audio signals to build a statistical model for song similarity.

Lorenzi [7] exctracted user data from Audioscrobbler[4] to anaylse song-similarities. Audioscrobbler is the data back-end of the popular social music network Last.fm[5], where listening statistics on song, artist and user level can be retrieved. Using the community data, he basically counted the number of co-occurences of any pair of songs in all of the user's 50 most played tracks list and derived song similarity (or song distance) out of them. If two songs co-occured in more track lists, those two songs were determined to be more basically similar. To comprise all song dependencies, the songs were embedded in a 10-dimensional Euclidean space, using the song distances determined in the track list anaylsis. Thus, each song was assigned a 10-dimensional coordinate, whereby each song can be identified using the embedding metrics. Clearly, a great basis for item-to-item collaborative filtering was established.

In total, the resulting space features a graph of 430'000 nodes. It was shown that songs are grouped by genre, where similar songs are close to each other, and more diverse songs appear apart. The interested reader can explore the map of songs on the MusicExplorer[6] webpage.

Coming back to recommender systems, it was this work's goal to produce novel recommendations. This issue has so far only been vaguely covered by current recommender systems. Examples of such approaches include topic diversification [8], a method to diversify personalised recommendation lists in order to reflect the user's complete spectrum of interest and thus improve user satisfaction. Another approach is case-based sequential ordering of songs [9], where new and meaningful playlists formed by a large collection of previously compiled playlists are recommended.

1.2 Contributions

We address the afore-mentioned finiteness of current recommender system's recommended products by focusing on novelty in music recommendation. The contributions

[4]See `http://www.audioscrobbler.net/data/webservices/`.
[5]See `http://www.last.fm/`.
[6]See `http://www.musicexplorer.org/`.

we make in this paper are the following:

History-based collaborative filtering. We propose an approach which analyses musical listening histories and identifies their waypoints in the map of music. Based on these waypoints, it was possible to filter out significant musical taste transitions, and to derive a global pattern of musical listening behaviour.

Expert recommendation. The pattern of listening behaviour was used to recommend music of new regions to users. Because the users are likely to be unfamiliar with new regions of interest, we propose an algorithm which mimes experts of regions to recommend popular music of these regions.

1.3 Organization

This thesis is organised as follows. We present the idea of history-based collaborative filtering as well as expert recommendation in Chapter 2. In Chapter 3, we evaluate the generated data set of musical listening behaviour. We end this paper with a conclusion and an outlook in Chapter 4.

Chapter 2

History-Based Collaborative Filtering

A major issue of current recommender systems is their lack in innovative recommendations. Recommended products might be accurate and suitable for the requester, but it is also in recommender systems' nature not to exceed the user's horizon of familiarity. Hence, their big strength of recommending high-quality items that are *similiar* to previous items becomes also their major drawback, as very *different* items will not be recommended. Often, recommendations refer to other people with similar taste, where items that were bought by other people – but not yet in possession by the requester – are recommended. As a result in music recommendation, most of recommended records for a given song are from the same genre, or even the same artist. In addition, a wider space of music is in most cases not explored. It is unknown how far a user could go in terms of dissimilarity, where genres would be far away, but the artists nevertheless be liked by the user.

To explore this topic, we investigated explicitly varieties in users' profiles, which were extracted from musical listening data. We tried to find transitions in tastes of users and to use these findings for a novel music recommendation algorithm. We call this new method *History-Based Collaborative Filtering*, where distant songs are favoured to be recommended. In the next section, we discuss this algorithm in detail.

2.1 History-Based Collaborative Filtering

The music taste of a person can be considered as constantly developing. It is rare that people still listen to the same music they liked ten years ago. Hence, we thought it would be interesing to develop an algorithm which would hint in which direction a person could go to in the future, based on music taste transitions that other people with similar profile experienced.

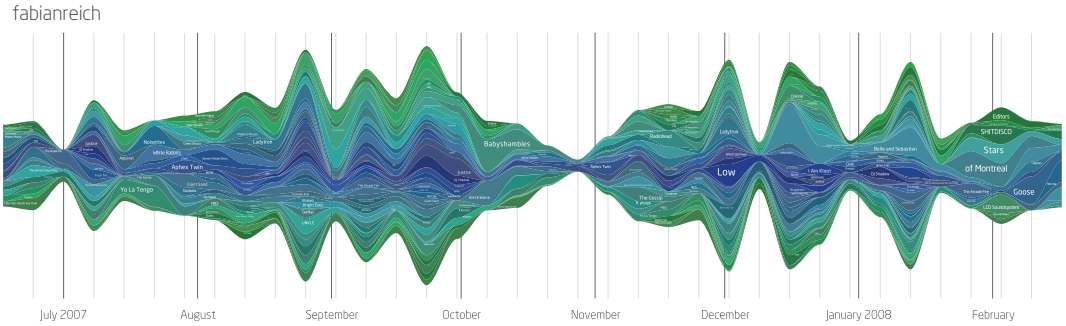


Figure 2.1: LastGraph of Last.fm user *fabianreich*.

The approach we present was inter alia inspired by LastGraph[1], a musical listening history visualiser for Last.fm user accounts. This web service analyses user histories and draws wavegraphs spanning a specified timeframe, depending on the individual frequency of artists' occurence at each time instance. The more an artist was listened to during a specific time tick, the thicker the artist's bar was pronounced compared to bars of other artists. The result is a wavegraph of continually growing and dimishing bars of different artists over a longer timespan. The drawings reveal some interesting properties. First, no artist keeps its frequency of occurence over the whole user history. More precisely, if one artist had a higher playcount than other artists (and thus a thicker bar), this characteristic was not preserved throughout the whole graph, meaning this artist's playcount sunk after a while, and other artists arose instead. Second, reoccuring artists did not have the same importance (i.e. thickness of bars) compared to before. This somehow tightens the assumption that users listen to artists in an intermittent fashion. Figure 2.1 illustrates what a LastGraph for a specific user profile does look like.

With these visulalisations at the back of ours mind, we hoped to find similar patterns in some users' history, where transitions in music taste would be pronounced, and to use them for our recommendation algorithm.

2.1.1 Analysing User Profiles and Data Crawling

As already mentioned in Section 1.1, the basis of this work was given by a 10-dimensional Euclidean space of song nodes developed by Lorenzi [7]. As each song is clearly identified by a 10-dimensional coordinate, Euclidean distances between two songs can easily be calculated, and be put on a par with similarity between songs. Recall that similar songs are located close to each other, as expressed by short Euclidean distances. Large distances, on the other hand, indicate variety.

Since Euclidean distances well reflect the similarity between songs, we can easily determine when users performed a change in their taste of music by having way points far away from their initial region of interest. To have user data to build our

[1]See `http://lastgraph.aeracode.org/`.

analysis upon, we extracted musical listening data from Audioscrobbler – the same repository that was previously used to determine song similarities. Audioscrobbler's Web Services API not only allows to retrieve a user's 50 most played tracks list, but also to access a weekly track chart containing each week's most played songs. This way, the musical listening history of a user can be retrieved and reconstructed on a week-by-week basis. Each song of the weekly chart then can be identified and assigned a coordinate in the Euclidean space, using the similarity measures of the underlying work. Thus, an eventual movement of song points over a longer time span can be captured by a shift of regions in space.

To collect user data from Last.fm users, a crawler was set up. The crawler reads out XML files of the weekly chart lists retrieved from Audioscrobbler, and stores song IDs together with user ID, playcount and timestamp (to capture what week a song was listened to) in a database. At the time of writing, the weekly track lists of 13'367 users were crawled.

2.1.2 Distance Measures in User Profiles

Using the embedding, songs of the weekly chart list were identified in space. As we wanted to capture the listening behaviour of users over a longer time span, we divided the characteristics of their musical listening history into periods of one week. Having the weekly chart lists, songs of a single chart list were equivalent to characteristics of one period. As individual songs to characterise the musical listening behaviour of one week seemed an indicator of too much diversity, we decided to calculate a weighted centre of mass (or centroid) for each week (where the weight was given by the playcount of the song), and to represent the listening behaviour of every week by a single centre of mass. Instead of having up to hundreds of songs per week, one single point stands for all songs the user listened to in that week.

To calculate the centre of mass, following equation was used. In the N-dimensionial Euclidean space, the weighted centre of mass is

$$c_i = \frac{1}{\sum_{j=1}^{M} p_j} \sum_{j=1}^{M} p_j x_{ij} \qquad \text{for } i = 1 \dots N \tag{2.1}$$

where x_{ij} denotes the i-coordinate and p_j the playcount of song j at time instance t. M is the number of songs the user listened to in week t.

To have an indicator of diversity for songs that form the centre of mass, the weighted standard deviation

$$s_i = \sqrt{\frac{1}{\sum_{j=1}^{M} p_j^2} \sum_{j=1}^{M} p_j^2 (x_{ij} - \bar{x}_i)^2} \qquad \text{for } i = 1 \dots N \tag{2.2}$$

was used, where the mean value $\bar{x}_i$ corresponds to the centre of mass c_i of equation (2.1).

As we now are given a single point that represents the musical listening behaviour for one week, the aggregation of weekly points eventually forms a pattern of movement.

To corroborate our belief, the Euclidean distance $d_{t_m t_n}$ between any two points $C_{t_m} = (c_1^{t_m}, c_2^{t_m}, \ldots, c_N^{t_m})$, and $C_{t_n} = (c_1^{t_n}, c_2^{t_n}, \ldots, c_N^{t_n})$ of user u was calculated, where $t_n > t_m$. Using this measurement, the remoteness of point C_{t_n} to point C_{t_m} can be captured. To compensate for the deviation of the individual points that form the centre of mass, distance $d_{t_m t_n}$ is divided by the mean deviation at time t_m,

$$\bar{s} = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} s_i \tag{2.3}$$

which results in

$$\tilde{r} = \frac{d_{t_m t_n}}{\bar{s}_{t_m}} \qquad \text{for } t_n > t_m \tag{2.4}$$

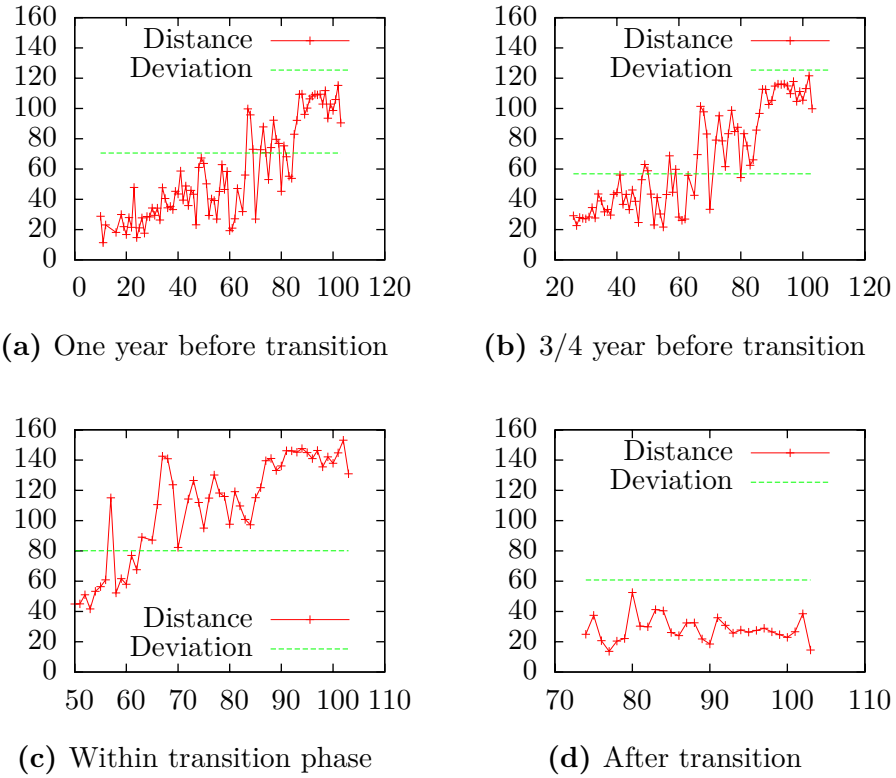
To illustrate precedent calculations, let us go through an example where different distances (or phases) are clearly visible. Figure 2.2 shows explicit differences in beginning and end of a user profile. The x-axis denotes the point in time (or week number), where the y-axis describes the respective values. The distance line represents the distance from a fixed centre of mass at time t_m to the centroids of the x-axis. In case of this Figure, t_m is located at the start point of the individual subfigures. The same holds for the deviation line, which shows the mean deviation at the starting point of each subfigure.

First, as can be seen in Figure 2.2 (a), the distance line in a first phase is between approximately 15 and 65, spanning around 65 weeks. Thus, Euclidean distances of centroids within a period of more than a year to centroid of week 1 are smaller than 65, and also below mean deviation of week 1, which is roughly above 70. To point it up, (b) shows similar properties, at a slightly later moment of time. The distances and deviation now refer to centre of mass of week 20. Clearly, distances up to week 65 are still within range of mean deviation, which are followed by a rapid increase. After week 90, distances swing into a persistent higher level. (c) is located at the beginning of a transition phase. Distances of centroids rapidly exceed the mean deviation of week 50, until they again turn into a higher level after week 90. Subfigure (d) shows distances from the moment where the transition phase is almost completed. Euclidean distances to week 72 are distinctly lower than its mean deviation, meaning that the movement of following centroids is much smaller than the range they are in.

We use these characteristics to perform a heuristic clustering over all available users.

2.1.3 Clustering of Users' History

As seen in Figure 2.2, there are obviously users which perform a transition in their musical listening profile. The most interesting users for our work were the ones that have a clear differentation from a first to a second stage, with a transition phase of low importance in-between. If many users that made a similar transition from region A to region B existed, we could consider this as evidence that a change in musical taste from one region to another follows a global trend. We therefore based the main assumption of this work on crawled user profiles which have a clear seperation from a first to a second phase.



(a) One year before transition

(b) 3/4 year before transition

(c) Within transition phase

(d) After transition

Figure 2.2: Different phases of transition of Last.fm user *mattymee*.

To comprise transitions in musical listening profiles, we developed a heuristic to explicitely detect such changes. The main idea is as follows. The musical listening history is grouped in three phases: start-phase, transition-phase and end-phase. Each of the phases is required to have a certain length (in terms of weekly centroids) in minimum, whereas the transition-phase's length is fixed. With respect to the minimal length, boundaries between start- and transition-, and transition- and end-phase can be drawn. If – for a certain allocation – the average between all weekly centroids in the start phase and those in the end-phase produce a big Euclidean distance, and at the same time the deviation of weekly centroids that form a cluster is small, we have a good indicator that a user's profile contains an expressive change in listening behaviour.

Algorithm 1 shows details of this heuristic. The criterion for drawing boundaries in a user's profile is the ratio between total distance and maximum mean deviation of both clusters. Figuring the weekly centroids strung together in a time axis, the boundary for start- plus end-phase are iteratively set throughout the time axis, and relative start- and end-clusters are formed. If a maximum in this ratio is detected, the boundaries are set at the respective position. Note that this algorithm does not detect noticeable transitions in the musical listening history of a user *per se*, instead it just marks where a maximum for the *individual* user was detected. We will later see in Chapter 3 that just a small portion of users experience a transition that is of sufficient significance to serve as a reference.

Algorithm 1: Clustering of a user's history

Input: C — the consecutive centroids of songs for each week.
Output: c_1, c_2 — two clusters of weekly centroids.

$r_{max} \leftarrow 0$;
$startIndex \leftarrow \text{begin}(C)$;
$endIndex \leftarrow \text{end}(C)$;
$minZone \leftarrow \text{length}(C)/numIntervals$;
for $t \leftarrow startIndex$ **to** $endIndex - 2 \cdot minZone$ **do**
    $vec_1 \leftarrow \text{Cluster}(C, startIndex, t)$;
    $vec_2 \leftarrow \text{Cluster}(C, t + minZone, endIndex)$;
    $dev_1 \leftarrow \text{Deviation}(C, startIndex, t, vec_1)$;
    $dev_2 \leftarrow \text{Deviation}(C, t + minZone, endIndex, vec_2)$;
    $d \leftarrow \text{Distance}(vec_1, vec_2)$;
    $r \leftarrow d/\text{Max}(dev_1, dev_2)$;
    if $r > r_{max}$ **then**
        $r_{max} \leftarrow r$;
        $c_1 \leftarrow vec_1$;
        $c_2 \leftarrow vec_2$;
    end
end

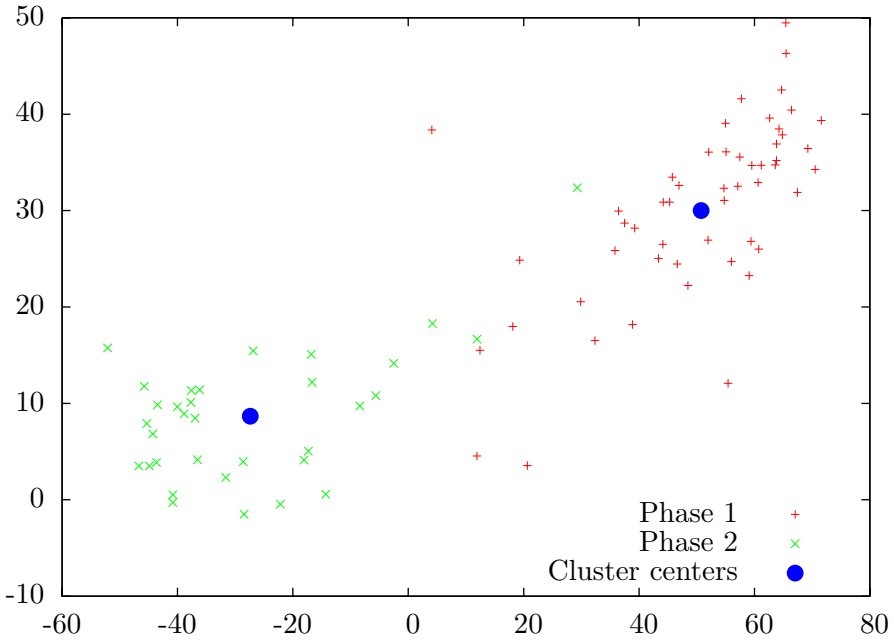


Figure 2.3: Weekly centroids of Last.fm user *mattymee* before and after transition (first 2 dimensions).

To illustrate, Figure 2.3 shows the result of clustering of a certain user. Although only the first two dimensions of the 10-dimensional space are drawn, it is evident that the musical listening history is partitioned. Centroids within the transition-phase are not drawn. The algorithm clusters weekly centroids from the start-phase (phase 1) and end-phase (phase 2) into two super-centroids (cluster centers). Of importance, only the two centroids (or cluster centers) c_1 and c_2 serve as the characteristic – in a very abstract manner – of musical transition behaviour for our subsequent calculations. As a consequence, a so-called transition vector $\vec{T} = \overrightarrow{c_1 c_2}$ represents a musical taste transition.

2.1.4 Data Refinement

The clustering of weekly centroids described in the previous section and the resulting transition vectors are not fine enough to be used for our recommendation algorithm without any modifications. First, the clustering algorithm did not exclude any user profiles by itself. The only outcome of the algorithm is a measurement of likeliness that user profiles represent musical taste transitions. Second, if there was any outliers among the users that experienced a transition, they were not excluded. Therefore, a transition vector refinement was performed, with the objective of having a small but effective and homogenous set of vectors that best represents taste transitions.

Algorithm 2 details the process of cluster refinement, which is done in an iterative

manner. First, most significant transition vectors were retrieved. Significance in this context was equivalent to distance between cluster start and end point. As we were interested in strong musical taste transitions, user profiles with biggest distance between cluster centers were determined to be relevant. Then, for each round, all remaining vectors were compared to their closest neighbouring vectors. As a measure, the average angle between each vector and its neighbouring vectors was calculated. If the angle was big, this vector was determined to be an outlier, because it would not be well aligned in direction of the other vectors. In this manner, the vectors with biggest angle were iteratively removed, until the maximum angle was lower than a certain threshold. As a result, a set of fewer but more homogenous vectors was produced. Figure 2.4 illustrates a vector that would be removed by the refinement process, since the average angle to its closest neighbouring vectors exceeds a threshold. As the choice of parameters has a great impact on the outcome of this refinement, the choice of parameters and its results are going to be discussed in Chapter 3.

Algorithm 2: Transition vectors refinement

```
Output: R — a set of refined transitions.
T ← MostDiverseTransitions();
foreach transition t in T do
    c_1 ← StartCoordinates(t);
    add c_1 to set of start coordinates;
end
ang_max ← MaxAngle(T);
while ang_max > ang_thr do
    ang_max ← 0;
    foreach transition t in T do
        ang ← AngleToNeighbours(t);
        if ang > ang_max then
            ang_max ← ang;
            t_max ← t;
        end
    end
    c_1 ← StartCoordinates(t_max);
    remove c_1 from set of start coordinates;
    remove t_max from set of transitions T;
end
R ← T
```

2.1.5 Aggregation of Transition Vectors

So far, mostly quantitative aspects of the underlying data and algorithms have been discussed. To see what transition vectors look like and how they perform in a more real-world environment, we aggregated transition vectors and ploted them in a 3D-graph. As our data space is 10-dimensional, we put a grid on a hyperplane that cuts

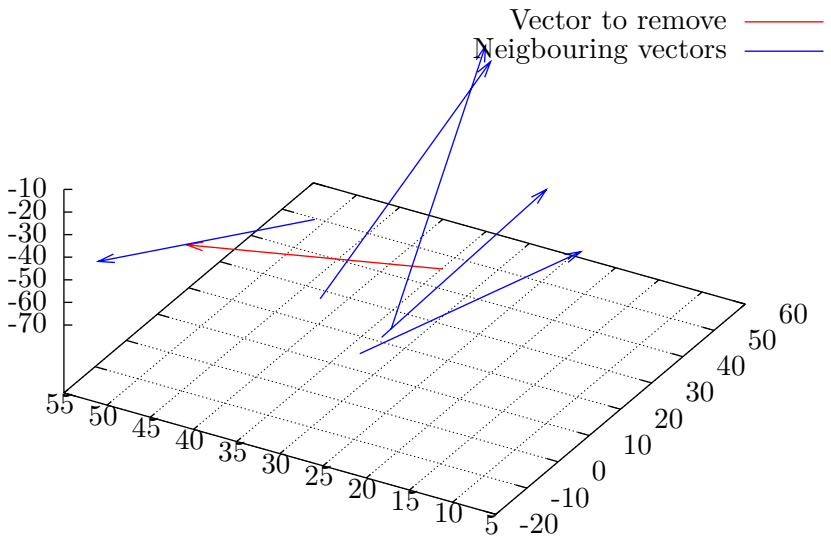


Figure 2.4: A vector marked to be removed by the refinement process and its closest neighbours. In a next step, also the other orthogonal vector to the left is likely to be removed (first 3 dimensions).

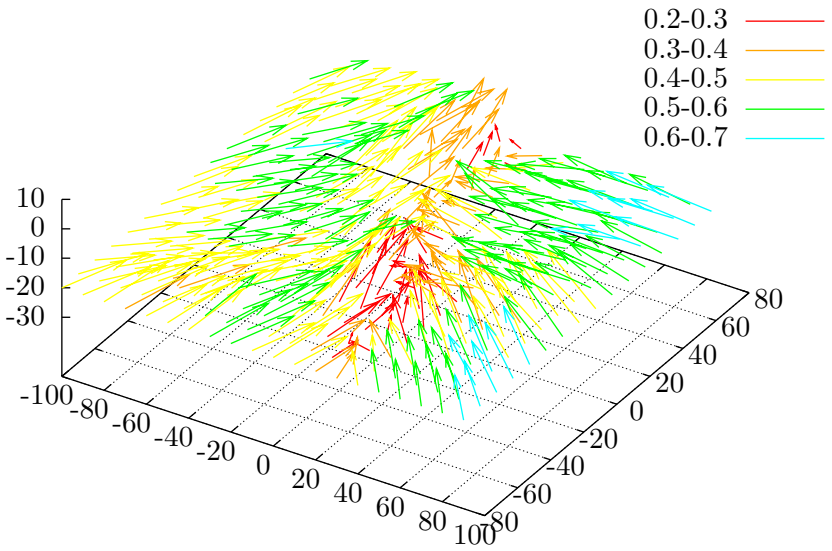


Figure 2.5: Average vectors of closest neighbours to gridpoints of hyperplane (first 3 dimensions).

the space, where dimension 1 and 2 are variable, and dimension 3 to 10 are fixed. Within the variable dimensions, a grid was formed to draw the general direction of transition vectors in different regions. For each of the grid points the 20 closest neighbours were evaluated, and their average vector was plotted, by moving average vectors to their respective grid points. For visibility reasons, the average vectors were shrunk to a length of 30. To comprise uniformity of different regions, the vectors were colored according to the ratio of average vector length to average length of vectors,

$$\tilde{r} = \frac{\| \sum_{i=1}^{M} \vec{v_i} \|}{\sum_{i=1}^{M} \| \vec{v_i} \|} \tag{2.5}$$

where M is the number neighbouring vectors. If close-by vectors are well aligned, the ratio is closer to 1. On the other hand, if vectors are irregular, the length of the averaged vector shrinks compared to the average length of individual vectors, and a ratio closer to 0 results. The resulting plot can be seen in Figure 2.5.

What we can see from the plot is that transition vectors around this hyperplane are nicely aligned, and experience smooth transitions from one region to another. According to the coloring of vectors, the shrinking of averaged vectors calculated in (2.5) is for most grid points not below 0.5. This would mean that transition vectors within a certain region form an almost uniform vector field, which indicates that transitions from one genre to another are not random. One could argue the plot in

Figure 2.5 may be composed of a few dominant transition vectors, which would shape this plot; as the $20 \times 16 = 320$ grid points are composed of approximately 240 different transition vectors, one transition vector dominates an average of just 1.3 grid points. Considering the beauty of this plot and the plain alignement of grid vectors, we had plausible evidence that the different transition vectors eventually form a global pattern of movement. Thus, we investigated further in recommendations based on transition vectors.

2.2 Expert Recommendation

One initial goal of this work was to mime expert recommendation to make our algortihm more sophisticated (and also a bit more economical). Thus, we wondered how to best incorporate such an expert recommendation algorithm. One possibility that arose from the results of previous section was to use transition vectors to point users into new and unknown regions, and then use experts to recommend catchy and popular music of those new regions, such that it would be easier for users to get familiar with new genres. The algorithm we developed is based on a simple idea. If a person out of many listens to a certain song the most (in terms of absolute playcount), we could say this person best knows this specific song, or even a set of similar songs. Other songs the same person listens to that have an even higher playcount can then be considered as the "best" music this person has to offer, and would be the ones this person would recommend. Therefore, we defined an expert of a set of songs as the persons with highest playcount, because they are in general most experienced. Algorithm 3 shows how song recommendation based on expert opinion could work.

2.3 Combined Recommendation Algorithm

Combining expert recommendation with transition vectors, we imagined to produce a successful music recommendation algorithm. The procedure of recommending music to a user could work as follows. First, the user would be asked to provide their Last.fm username, or a set of songs they like. Out of the tracks that are provided (in case of Last.fm username, the most played tracks could be retrieved), song coordinates would be identified in the Euclidean space to calculate their centre of mass. The centre of mass would represent the starting point from where closest musical taste transitions of other users would be searched. Having a set of transition vectors close to the starting point, either their average vector would be calculated, or their end points would be averaged. In both cases, a vector would point from the starting point into a new region, based on musical taste transitions that other users experienced. As easy-to-listen songs from the new region should first be recommended to the user (exotic music is probably not what the user is going to like), we could search for closest songs of the vector end-point and apply expert recommendation as described in Algorithm 3. This way, a fully functional music recommendation algorithm could be realised. For testing purposes, we wrote a program called MusicMate, which works

Algorithm 3: Expert finding and song recommendation

Input: S — a set of songs.
Output: R — a set of recommended songs.

```
foreach song s in S do
    U ← TopUsers(s);
    foreach user u in U do
        T ← TopTracks(u);
        D_u ← Distance(S, T);
    end
    E′ ← ClosestUsers(D);
    add E′ to set of experts E;
end
foreach expert e in E do
    T ← TopTracks(e);
    foreach track t of T do
        A ← Union(Artists(S), Artists(R));
        if Artist(t) not in A then
            add t to set of recommended songs R;
        end
    end
end
```

in a similar fashion as described here.

In the next chapter, the evaluation of transition vectors is going to be discussed.

Chapter 3

Evaluation

We talked about various aspects of recommender systems in Chapter 1, and described ideas and preliminary results of our recommendation algorithm in Chapter 2. Now we will discuss the data we generated in more detail, and draw conclusions from the results.

The analysis of the generated transition vector data set revealed some interesting properties. First, we could show that individual transition vectors are related to their neighbours in terms of direction. Second, we found out that averaging of neighbouring transition vectors better represents the alignment of a region than accounting for their individual directions. We concluded from these findings that single transition vectors are not strictly aligned, but that the aggregation of close-by vectors yields a general direction of musical taste transition. Hence, deterministic changes in musical listening behaviour exist in principle.

Before we discuss the evaluation measures and results, we first are going to analyse general properties of the embedding space and its transition vectors. Due to the limited time frame of this project, unfortunately no survey of human feedback could be conducted.

3.1 General Data Analysis

3.1.1 Clustering Process

We have mentioned that transition vectors are equivalent to clustered listening histories. As a data source, musical listening profiles from Last.fm users were retrieved, by setting up a crawler. Since these user profiles formed the basis of this work, let us first discuss crawling statistics. At the time of writing, 13'367 listening histories were crawled, as table 3.1 shows. For the clustering process (Algorithm 1, Section 2.1.3), not all of the user profiles were suited. First, due to transmission and crawling defects, some listening histories could only be partly retrieved. Second, parts of users did not contain enough history data (young or inactive user accounts), and were there-

	Number	Baseline
Total number of crawled users	13'367	100.00%
Corrupt user profiles (crawler failures, insufficent user data)	5'332	39.89%
Clustered user histories	8'035	60.11%
Transition vector refinement input	650	4.86%
Transition vector refinement output	575	4.30%

Table 3.1: User data statistics.

fore ignored by the clustering process. As a constraint, user profiles were required to have at least one year of music history. With this setting, we could ensure that user profiles contained a long enough listening history from which we would derive the weekly centroids. Additionally, hardly no outliers due to limited listening data would be included. At the end of the clustering process, 8'035 users remained.

3.1.2 Refinement Process

Let us look at transition vector lengths of clustered user profiles, as illustrated in Figure 3.1. This Figure shows a histogram of transition vector lengths in the x-axis, and the number of respective vectors in the y-axis. As can be seen, the majority of transition vectors have an Euclidean length between 10 and 50. One can agree that in general not all persons perform a transition in their musical taste, but only a small subset of them, which then would serve as our reference transitions. Thus, we decided to pick users with biggest Euclidean distance between transition start and end point, since the change these users experience is stronger, and not all users experience a transition within one year. Vectors with small lengths can be considered as insignificant because their length is too short compared to the total size of the embedding.

For the refinement process (Algorithm 2, Section 2.1.4), a number of parameters have to be set.

Number of vectors. The number of transition vectors with biggest Euclidean distance from the repository of clustered user profiles. A high number of vectors could involve many insignificant transition vectors, and blur the refined set.

Number of neighbours. For every round, the refinement process evaluates a fixed number of closest neighbours for each of the vectors. The average alignment of neighbours serves as an indicator if the individual vector was an outlier.

Distance threshold. If a vector's closest neighbours are too far away, their alignment would become meaningless. Thus, if the closest neighbour's distance exceeds a certain threshold, the neighbouring vector is ignored and not added to the individual set of neighbours.

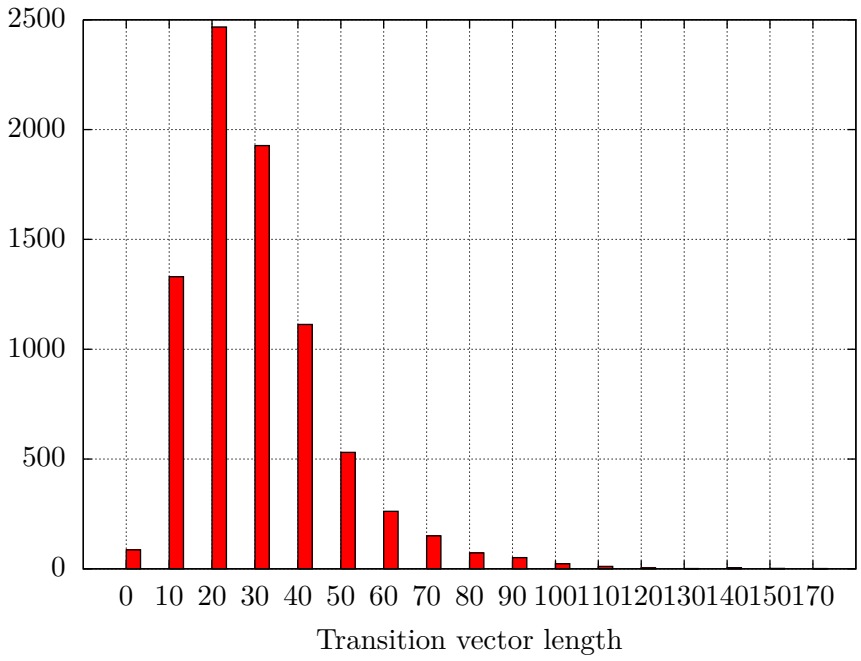


Figure 3.1: Histogram of transition vector length distribution. Note that each length division spans a length interval until the beginning of the next division.

Angle threshold. The maximal average angle between a single vector and its neighbouring vectors. If this angle is exceeded, vectors with biggest angle are iteratively removed.

As an input, we chose the 650 longest transition vectors, 5 neighbouring vectors, a distance threshold of 40 and a maximal angle of $\pi/2$ (90°) for each of the transition vectors. The result was a refined set of 575 transition vectors (last row in table 3.1), which had an average vector length of 75.92 (last row in table 3.2). As a reference value, Bossard [10] calculated that the diameter of the embedding space is around 500.

3.1.3 Song Point and Transition Vector Properties

To relate the refined set of transition vectors to the space of songs, we now will analyse properties of transition vectors in the context of the embedding. As previously mentioned, the space of songs is 10-dimensional. Needless to say, Euclidean distances between songs spanning different regions and musical genres become much bigger than in – let's say – two-dimensional space. Interestingly, analyses of the song space let us assume that songs themselves are clustered (or "clumped") into multiple smaller chunks. Table 3.2 shows evidence that this assumption is correct. Considering the average distance from a random transition vector *start* point to its next song (row

	Measurement	Value
1)	Average distance from a random song to its next song	9.39
2)	Average distance from a random song to its next start point	58.00
3)	Average distance from a random start point to its next song	25.45
4)	Average distance from a random end point to its next song	32.00
5)	Average transition vector length	75.92

Table 3.2: Song to transition vector statistics, based on the refined transition vector set. The term *point* refers to a transition vector start or end point.

	Measurement	Value
1)	Average distance from a start point to its next start point	28.22
2)	Average distance from an end point to its next end point	29.52

Table 3.3: Transition vector statistics, based on the refined transition vector set. The term *point* refers to a transition vector start or end point.

3), and the average distance from a random transition vector *end* point to its next song (row 4), we see that these two numbers subsistentally differ. Because the first distance is lower than the second, this implies that transition vectors start points are generally in more dense regions than their end points. Thus, there are regions where song points are closer to each other, and regions where they are further away. Hence the clustering of song points into chunks.

Table 3.2 reveals also other interesting properties. First, song points have an average distance of around 9 to their closest neighbouring song (row 1). If the 430'000 songs were uniformly distributed in space, this distance would be roughly 75 [10]. Hence another indicator for clustered song points. Second, the average distance from a song to its next transition vector start point (row 2) is not much smaller than the average transition vector length (row 5). In other words, to cover the distance from a song to its closest transition vectors is in similar dimensions than distances of transition vectors themselves. In an extreme case where a transition vector is in opposite direction to the song point we came from, the resulting transition shift would be marginal. Applied to song recommendation, substantial changes in music regions could not always be realised by our recommendation algorithm. To put things right, the density of significant transition vectors is likely to increase with additionnal crawling, such that the problem of sparsity could be resolved.

Considering distances among transition vectors themselves, Table 3.3 shows more statistics. The main point here is that transition vector density around their start points (row 1) is in similar ranges to the density around their end points (row 2). This suggests that transition vectors start and end points are uniformly distributed over the total embedding area.

Finally, Figure 3.2 illustrates the distribution of song and transition vector start and

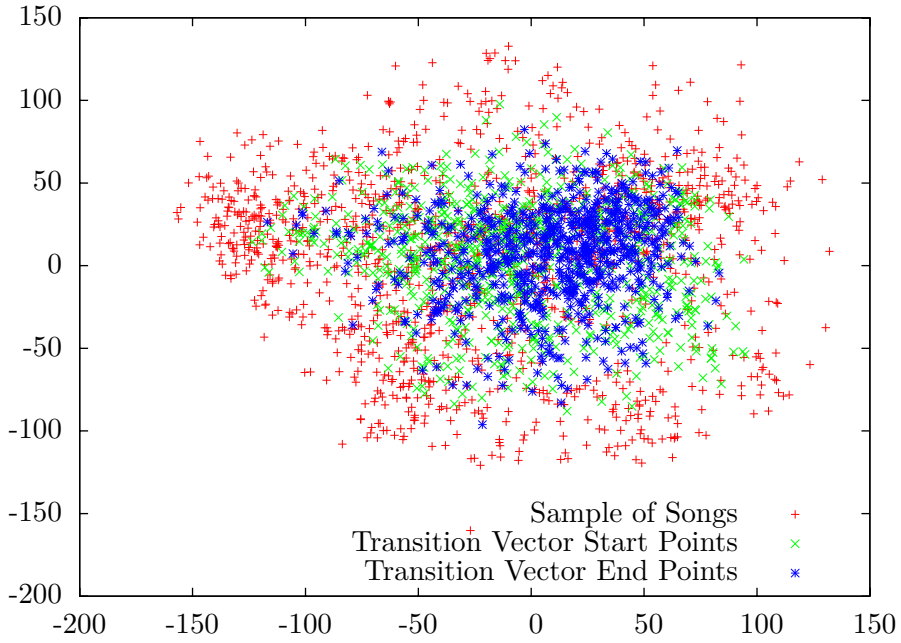


Figure 3.2: Start and end points of transition vectors (first 2 dimensions).

end points. As we can see, song points span a bigger region than transition vector points, which are aggregated around a smaller area. Especially transition vector end points are accumulated towards the centre. For song recommendation, music around the centre is therefore more likely to be recommended than songs on the border of the space.

3.2 Evaluation Measures

To measure the quality of our calculations, we used two different methods. First we evaluated angles between neighbouring transition vectors to make assumptions about their alignment. Afterwards we counted songs which were intersected by an area. As mentioned earlier, a proof-of-concept music recommendation program called MusicMate was written, but not evaluated in the context of a survey of human feedback.

To facilitate the notion of vectors in subsequent sections, we are going to introduce a couple of new terms.

Control vector. A control vector is a randomly chosen transition vector from the total repository of transitions vectors. This vector serves as the reference neighbouring vectors to control vectors are compared.

Reference vector. Reference vectors are closest neighbouring transition vectors to a certain control vector, and used to compose region vectors (see below). When

performing the evaluation, control vectors were omited from the set of reference vectors to not falsify the tests.

Region vector. Finally, a region vector is the combination of close-by reference vectors, by either taking their average, or by averaging their end points. Thus, the region vector is a combination of multiple transition vectors resulting in a single vector pointing from one region to another. Applied to music recommendation, we generate a region vector according to users' current region, and refer to the end point of the this vector when producing a new recommendation. In an extreme case where only one transition vector was considered, the region vector would be equal to this transition vector. Let us denote a region vector with $\vec{v} = \overrightarrow{C_1C_2}$, where C_1 is its start point, C_2 its end point, and r the length of $\vec{v}$.

3.2.1 Transition Vector Angles

For the first measurement, we wanted to evaluate the similarity between close-by transition vectors in terms of direction. If neighbouring vectors pointed to similar directions, it was an indicator that people from one region generally performed a musical transition to the region these vectors point to. Thus, the direction of transition vectors would indeed correspond to the general movement of music listeners from one region to another.

In the evaluation, we proceeded as follows. Out of the original repository of transition vectors, we chose a control vector[1]. For the control vector, its closest neighbouring vectors (or reference vectors) from the refined repository of transition vectors were identified. By calculating the angle of the control vector to an average or the sum of reference vectors, we got an indicator of uniformity of close-by vectors and thus a measurement of determinism to point from one region to another. An example of nicely aligned vectors can be seen in Figure 3.3, where a control vector and its closest neighbouring vectors were drawn. In this example, the resulting angle would be relatively small, as all reference vectors point to similar directions.

For the angle measurement, a number of parameters have to be set. Following parameters were most important when performing the evaluation.

Number of neighbours. The number of closest neighbours for each control vector.

Distance threshold. The maximal distance of the start point of a neighbouring vector to the start point of the control vector. If a reference vector is considered to be among the closest neighbours, but nevertheless exceeds the threshold, the reference vector can be ignored by the measurement.

Region vector. This defines how the region vector is composed. Either, the average of neighbouring vectors is calculated and then moved to the starting point of

[1]Note that we did not chose the control vector from the refined set of transition vectors, because then we would cheat, as the remaining vectors in the refined set are indeed aligned.

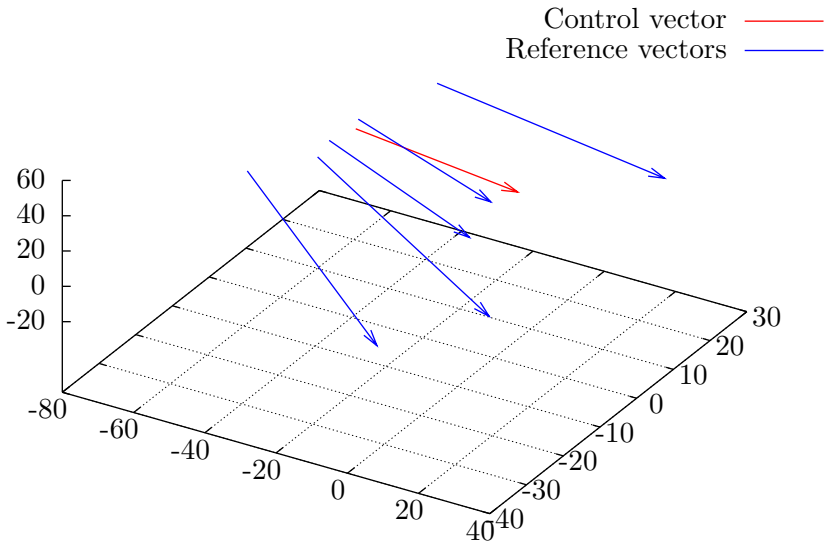


Figure 3.3: Transition vector of Last.fm user *Diagnosticfrog* and five closest neighbouring vectors (first 3 dimensions).

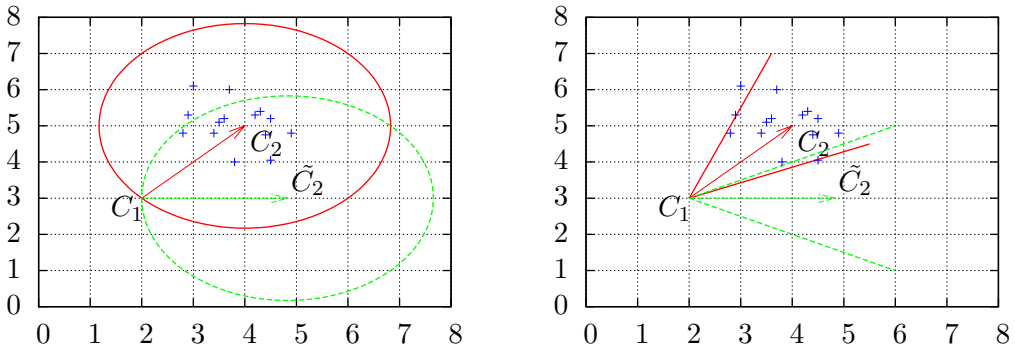
the control vector, or their end points are averaged to form a vector between the control vector starting point and the averaged end points.

Ratio threshold. As an exclusion criterion, some control vector measurements could be explicitly omitted. This represents cases where the outcome of the recommendation process can be *a priori* foreseen as imprecise, and thus better no recommendation than a bad one is generated. Here, the shrinking of average vector length compared to average length of neighbouring vectors according to equation (2.5) was used as a criterion. The resulting ratio represents the shrinking; if the ratio was below a threshold, no angle test was performed.

The results of the angle measurement are going to be discussed in Section 3.3.

3.2.2 Song Count Approach

A second measurement we applied operates on a song-basis. For a certain test user (or control vector), the closest neighbouring transition vectors were identified to generate a new region vector. Based on this region vector, an area was spanned. Since the test user itself also forms a transition vector, we know from the listening history which songs he actually listened to after transition. Thus, we counted the number of test user's songs after transition within this area. If the number of test user's songs intersected by this area was high, we could assume that the region vector was



(a) Ball approach **(b)** Cone approach

Figure 3.4: Original and random erea with songs of user after transition (first 2 dimensions).

accurate. Opposed to first measurement, not only the direction of transition vectors was involved, but also their length.

Preliminary Measures

In a first try, we neglected the total number of songs given by the embedding, and just considered songs a single user listened to after transition. The idea was to compare songs of the user in the intersection area to the total number of songs he actually listened to after transition. The more songs we matched, the better our region vector would be. Thus we defined a ratio

$$\tilde{r} = \frac{n_i}{u} \tag{3.1}$$

where n_i is the number of songs in the area, and u the total number of songs of this user after transition. We then defined two kinds of areas in which we were looking for a user's songs. As described, the areas were based on region vectors, which themselves are derived from neighbouring transition vectors.

In the beginning, we defined a 10-dimensional ball around centre C_2 of region vector $\vec{v}$, where the radius of the ball was given by the region's vector length r. We counted the number of songs in the ball and compared it to the total number of songs, according to (3.1). If we generated a random ball with similar radius r but random centre $\tilde{C}_2$, such that $\|\overrightarrow{C_1\tilde{C}_2}\| = r$, the outcome of (3.1) was in similar magnitude to the outcome of the ball formed with the original region vector. Hence, it was difficult to distinguish between a "correct" from a random vector. Figure 3.4 (a) illustrates this idea, where the red circle is the original segment, and the dashed green circle corresponds to the random segment. The blue dots are songs of the test user (or control vector) after transition.

As an alternative, we defined a 10-dimensional cone, starting at the start point C_1 of region vector $\vec{v}$. The axis of the cone was put between C_1 and C_2, which corresponds to $\vec{v}$. The surface spanned by the cone had a maximal angle of $\pi/3$ (60°) to its axis, meaning that a surface of maximal 120° was opened. The height of this cone was bound at $2r$. Similar to before, the ratio of songs according to (3.1) was calculated. To benchmark the cone measurement, a random cone was generated. With similar properties to the original cone, only its direction changed, such that its axis was now given by C_1 and $\tilde{C}_2$, where $\|\overrightarrow{C_1\tilde{C}_2}\| = r$. Figure 3.4 (b) shows what this approach would look like in two dimensions. In contrast to before, the outcome of equation (3.1) substantially differed for the original and the random cone. Hence we knew that we found a measurement for which we better could distinguish a "correct" region vector from a random one, and thus better evaluate the quality of individual transition vectors.

Main Measures

Although results of the cone approach looked promising, we decided that this measurement was not entirely fair, since only songs of the user, but not songs of total embedding were considered. As we saw in Figure 3.2 that song points and transition vectors are aggregated towards the centre of the embedding, random vectors are penalised, since they are more likely to point towards the border of the embedding than a region vector. Thus, the probability of capturing any songs in a border region is much lower than capturing songs towards the centre. Under this consideration, we decided to compare user songs after transition to the embedding's *total* number of songs within this cone,

$$\tilde{r} = \frac{n_i}{s_i} \tag{3.2}$$

where n_i is the number of a user's songs after transition, and s_i the total number of songs in this cone. With this comparison, vectors pointing into sparser areas experience a similar treatment to vectors pointing into denser regions when measuring the quality of transition vectors. As an example, Figure 3.5 illustrates when this measurement is applied. The blue dots are songs this user listened to after transition. The red dots – obviously in much higher occurrence – are songs that are spanned by the cone of this user's transition vector (note that we previously talked about region vectors; however, for this case, the region vector is equivalent to the transition vector). The green dots are songs in the embedding spanned by the opposite cone, which stands for the case when a cone is spanned by a random vector. Clearly, songs of this user are aggregated within the original cone, whereas songs within the opposite segment are rare. For this example, the outcome of equation (3.2) for the original cone is around 10 times higher than the opposite cone.

Again, a couple of parameters have to be set when applying this measurement.

Maximal angle. The angle between the axis and the surface of the cone. The bigger the angle the more user songs are intersected, but also the more songs of the embedding are contained.

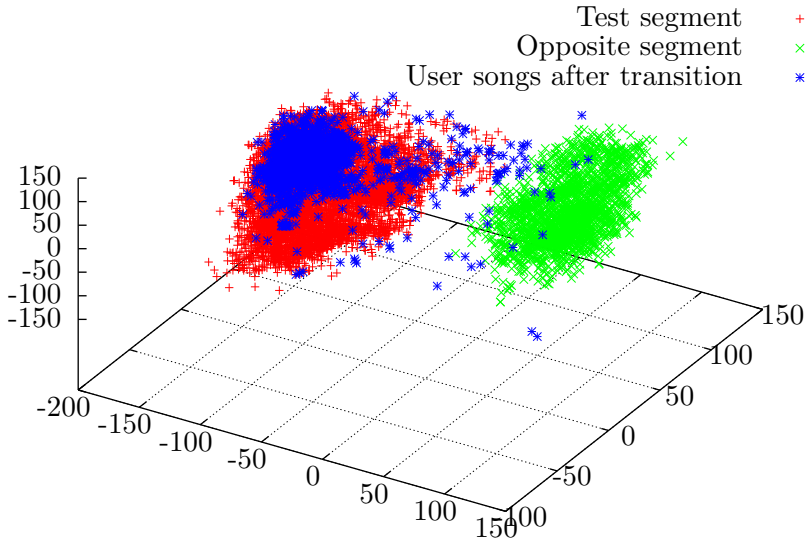


Figure 3.5: Total songs in cone spanned by transition vector and cone spanned by opposite vector versus songs of Last.fm user *mortyliendre* after transition (first 3 dimensions).

Maximal radius. The maximal radius between the start point of the cone and its furthest away song points; this corresponds to the height of the cone.

Target region. As before, this defines how the region vector – the vector that defines the cone – is composed of.

In the next section, results of both evaluation measurements are being presented.

3.3 Results

The outcome of the evaluation process was quite positive. First, we showed that close-by transition vectors – when explicitly omitting cases where vectors are too diverse – are within a certain angle to each other and thus aligned. Second, we showed that cones spanned by region vectors intersect with more user songs than cones that simply point towards the centre of the embedding, in relation to the embedding's total number of songs within the cones.

3.3.1 Transition Vector Angles

We described that region vectors can be composed of different numbers of neighbours, that the shrinking of the region vector can be an exclusion criterion and so on. For illustration purposes, we split the results into two parts, where the region vector was calculated differently.

Averaged Vectors

For this first test, the region vector was formed by averaging the neighbouring transition vectors. As described in Section 3.2.1, the angle of a randomly chosen transition vector (or control vector) was compared to the average of its closest neighbouring vectors from the refined set of transition vectors (Section 2.1.4). The result can be seen in Tables 3.4, 3.5 and 3.6. In Tables 3.5 and 3.6, we excluded cases where the shrinking of the resulting vector was too strong. If the shrinking was below this threshold, the angle between the randomly chosen vector and the resulting region vector was not incorporated into the average angle of all control vectors, but instead marked as "omitted". For each number of neighbouring vectors, 2'600 tests were performed. As can be seen (especially in Table 3.6), the higher the threshold of the shrinking ratio, the more cases are omitted. However, the resulting angle clearly improves. Interestingly, the more neighbours that compose the average vector, the better the angle between test and region vector, while the angle between test and individual vectors – the vectors that compose the region vector – remains on a similar level, as can be seen under "ind. angles".

Averaged End Points

In a second run, the region vector was generated by averaging the end points of closest transition vectors, while its start point was identical to the starting point of the control

Ngbs	Tested	Omitted	Tst/Omt	Ind. Angle	Avg. Angle
1	286	2314	0.12	1.91	1.91
3	2'600	0	∞	1.12	1.05
5	2'600	0	∞	1.14	1.00
9	2'600	0	∞	1.19	1.00
15	2'600	0	∞	1.21	1.00
19	2'600	0	∞	1.20	0.97

Table 3.4: Statistics of averaged vectors with minimal shrinking ratio 0.

Ngbs	Tested	Omitted	Tst/Omt	Ind. Angle	Avg. Angle
1	311	2'289	0.14	1.94	1.94
3	2'459	141	17.44	1.11	1.05
5	2'151	449	4.79	1.08	0.94
9	1'716	884	1.94	1.06	0.87
15	1'412	1'188	1.18	1.04	0.81
19	1'307	1'293	1.01	1.07	0.82

Table 3.5: Statistics of averaged vectors with minimal shrinking ratio 0.6.

vector. Similarly to before, the angles improved when setting a higher shrinking ratio threshold (Tables 3.7, 3.8 and 3.9 in ascending order), and also when the region vector was composed of more neighbouring vectors.

3.3.2 Song Count Approach

We also evaluated user songs after transition within a cone, as described in Section 3.2.2. Herby, songs of a randomly chosen user from the test set were counted within cones spanned by different vectors, and divided by the embedding's total number of songs intersected by these cones. We constrained the angle between surface and axis of the cones to $\pi/3$ (60°), yielding in an area that spans a total angle of 120°. The

Ngbs	Tested	Omitted	Tst/Omt	Ind. Angle	Avg. Angle
1	300	2'300	0.13	1.80	1.80
3	1'750	850	2.05	1.01	0.95
5	869	1'731	0.50	0.94	0.83
9	412	2'188	0.18	0.91	0.77
15	158	2'442	0.06	0.83	0.64
19	110	2'490	0.04	–	–

Table 3.6: Statistics of averaged vectors with minimal shrinking ratio 0.8.

Ngbs	Tested	Omitted	Tst/Omt	Ind. Angle	Avg. Angle
1	288	2'312	0.12	1.89	1.85
3	2'600	0	∞	1.14	1.07
5	2'600	0	∞	1.14	1.03
9	2'600	0	∞	1.18	1.02
15	2'600	0	∞	1.20	1.01
19	2'600	0	∞	1.21	1.00

Table 3.7: Statistics of averaged end points with minimal shrinking ratio 0.

Ngbs	Tested	Omitted	Tst/Omt	Ind. Angle	Avg. Angle
1	279	2'321	0.12	1.86	1.84
3	2'455	145	16.93	1.11	1.06
5	2'167	433	5.00	1.10	1.00
9	1'734	866	2.00	1.07	0.91
15	1'429	1'171	1.22	1.07	0.88
19	1'291	1'309	0.99	1.06	0.86

Table 3.8: Statistics of averaged end points with minimal shrinking ratio 0.6.

height of the cone was bound to $2r$, equivalent to double of the region vector's length. The number of neighbouring transition vectors differed in the two settings. Tables 3.10 and 3.11 show some handpicked examples of this experiment, whereas an overall analyis follows later down. The first column indicates the user id in the internal database. The data from columns 2 to 4 ("ctrl vec") describe a cone spanned by this control vector. This column was included to have more precise reference values for cones spanned by other vectors. Columns 5 to 7 ("reg vec") are then the result of a cone spanned by averaging either 5 neighbouring transition vectors (Table 3.10), or 10 vectors (Table 3.11), referred as the region vector. Finally, we generated a cone simply pointing towards the centre of the embedding, having the same length as the

Ngbs	Tested	Omitted	Tst/Omt	Ind. Angle	Avg. Angle
1	306	2'294	0.13	1.87	1.82
3	1'728	872	1.98	1.01	0.98
5	849	1'751	0.48	0.95	0.88
9	443	2'157	0.20	0.91	0.79
15	191	2'409	0.07	0.86	0.67
19	106	2'494	0.04	–	–

Table 3.9: Statistics of averaged end points with minimal shrinking ratio 0.8.

region vector cone. This test was included to represent a greedy recommendation algorithm that would just point towards the centre of the embedding. The outcome can be seen in columns 8 to 10, denoted under “cntr vec”. The comparison between a region vector cone and a random vector cone was not included because the results were not statistical significant. For all of the different tests, “usr sngs” are songs of the user after transition within the cone, “all sngs” are all songs of the embedding in the cone, whereas “score” denotes the ratio of both the values, according to equation (3.2).

User ID	Ctrl vec Usr sngs	All sngs	Score	Reg vec Usr sngs	All sngs	Score	Cntr vec Usr sngs	All sngs	Score
2684	93	50’000	0.17	4	1’075	0.37	0	202	0.00
4239	28	3’677	0.76	81	16’713	0.48	54	11’377	0.47
4695	117	1’376	8.50	122	12’234	1.00	74	20’703	0.36
4843	1’828	19’088	9.58	0	0	0.00	0	0	0.00
4959	356	48’493	0.73	170	15’518	1.10	149	29’830	0.50
6861	204	4’205	4.85	35	1’325	2.64	48	4’128	1.16
6871	427	3’688	11.57	53	9’984	0.53	44	11’267	0.39
7644	23	9’439	0.24	164	53’177	0.31	181	69’346	0.26
11366	13	955	1.30	40	6’921	0.58	41	8’065	0.51
11971	0	291	0.00	0	3’108	0.00	1	11’199	0.01

Table 3.10: Examples of cone tests with region vector consisting of 5 neighbours.

User ID	Ctrl vec Usr sngs	All sngs	Score	Reg vec Usr sngs	All sngs	Score	Cntr vec Usr sngs	All sngs	Score
2791	51	27’344	0.19	21	2’921	0.72	5	5’348	0.09
2996	3	2’862	0.10	6	12’352	0.05	16	17’994	0.09
3453	900	35’255	2.55	74	2’276	3.25	32	1’258	2.54
4079	433	103’423	0.42	21	4’144	0.51	46	6’129	0.75
4400	257	37’672	0.68	27	11’079	0.24	9	14’088	0.06
4959	356	48’493	0.73	237	29’908	0.79	192	42’126	0.46
8068	2’278	22’842	9.97	0	13	0.00	0	61	0.00
8188	10	2’449	0.40	3	3’107	0.10	6	7’075	0.08
8204	3	1’051	0.29	34	3’244	1.05	49	7’416	0.66
9735	264	211’218	0.12	16	6’918	0.23	13	8’105	0.16

Table 3.11: Examples of cone tests with region vector consisting of 10 neighbours.

As the number of total embedding songs within a cone varies much more than the number of user songs intersected by a cone (see Tables 3.10 and 3.11), comparisons between vectors pointing to opposite directions are difficult to render. For the sake of fairness, we generated a cone pointing towards the centre of the embedding, where the density of total songs is in similar dimensions to the cone spanned by the region vector (see “all sngs" columns in tables). This made score values of region vector cones according to equation (3.2) much more comparable. Thus, we divided the score of the region vector cone by the score of the centre vector score, and averaged this ratio over multiple runs. The outcome of this experiment can be seen in Table 3.12.

Ngbs	Avg. Ctrl Vec Score	Avg. Reg Scr/Cntr Scr
5	1.55	3.38
10	1.98	1.72

Table 3.12: Average score of control vector and average ratio of region vector score to centre vector score, with differing number of neighbours for region vector. This test was performed with 100 different control vectors.

3.4 Discussion

Considering the angles between test and region vector, the evaluation of transition vectors delivered promising results. The outcome of the two variants is in similar ranges, whereas the averaging of neighbouring vectors yielded in slightly better results than the averaging of end points. Trading the number of successful cases against the cases where no recommendation would be generated, the average angle between a control and region vector was constrained down to 0.81 (46°), though still producing a recommendation every second time (Table 3.5, 15 neighbours: 1.18). For comparison only, a random vector would yield in an average angle of 90°, meaning that our method is almost double as precise. Accounting for the improving accuracy with increasing number of neighbours, we conclude that it is not just the closest neighbours that express the musical transition behaviour of users, but that instead the aggregation of neighbouring transition vectors expresses the general tendency of a region.

The comparison between region vector cones and centre vector cones was also positive. In these tests, the average score value of the region vector cones was up to three times higher than the score value of centre vector cones (Table 3.12, column 3). We are aware that these numbers can't be taken for granted due to statistical variance; however, they express the general tendency that more user songs are intersected by cones spanned by region vectors than by cones spanned by "random" vectors, in relation to the embedding's total number of songs. Therefore, areas we span by using transition vectors are more accurate in terms of song matching than areas spanned by a greedy music recommendation algorithm. Overall, the approach of history-based collaborative filtering experienced validation.

Chapter 4

Conclusion

We have presented a new technique for generating music recommendations with unique functionality. This work's algorithm does not strive for generating each user's personalised recommendation lists, instead it rather attempts to represent a general movement in musical listening behaviour, and to recommend music from regions a user could potentially be interested in. We think this approach has so far been unstudied and believe that this method holds great achievement potential. The results showed that extracted music listening histories combined with our map of music are accurate in terms of musical directions and song matching. Concerning the idea of expert recommendation, this proposal could not be evaluated due to the lack of a survey of human feedback. However, it seems that this recommendation algorithm favors to recommend more popular music within certain regions. We believe that the combination of transition vectors together with expert recommendation add to a mighty recommendation tool.

4.1 Future Work

For future development, the algorithms and underlying data set could be further refined. Considering the number of songs that form a weekly centroids, the deviation between individual songs and their weekly centroid is considerable. Clustering a whole musical listening history based on weekly centroids into a vector of two points makes the data set even more vulnerable to outliers, and it remains an open question how representative the generated data set of transition vectors really is. Nevertheless, this method showed to feature interesting properties as well as notable validation, and made it appealing to apply right because of this simplicity. As a suggestion for future work, the musical listening history of users could be analysed in a more fine-grained fashion, such that the history of users would be divided into more cluster points than just two of them. Also, the analysis of song points and transition vectors showed that the map of music contains holes, where songs are grouped around more dense areas. This property is not considered by our algorithm, and should be included by a refined

clustering process, so that transition vector start or end points would not be located within such holes.

Filtering user profiles by demographic aspects would be another interesting task. This way, users with similar gender, age or social environment would be grouped, such that music from people with more similar interests would be recommended and accuracy of the algorithm could be improved. However, the strength of this existing algorithm is its focus on musical taste, and not on the social environment of users. The back-end for generating recommendation of this algorithm is given by musical listening profiles of *any* users – where the common denominator of two users is right the same preferences in music.

Considering the database's limitations in the number of meaningful transition vectors, the number is likely to increase with additional crawling. Once enough musical listening histories of significant transitions are stored, the quality of the algorithm might be noticeably improved, and a bigger diversity achieved.

References

[1] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2): 133–151, July 2001.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005.

[3] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan/Feb 2003.

[4] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW 2001, 10th International Conference on World Wide Web*, May 2001.

[5] Daniel P. W. Ellis, Brian Whitman, Adam Berenzweig, and Steve Lawrence. The quest for ground truth in musical artist similarity. In *ISMIR 2002, 3rd International Conference on Music Information Retrieval*, October 2002.

[6] Beth Logan. Music recommendation from song sets. In *ISMIR 2004, 5th International Conference on Music Information Retrieval*, October 2004.

[7] Michael Lorenzi. Similarity measures in the world of music. Master's thesis, Swiss Federal Institute of Technology Zurich, Department of Computer Science, August 2007.

[8] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *WWW 2005, 14th International Conference on World Wide Web*, May 2005.

[9] Claudio Baccigalupo and Enric Plaza. Case-based sequential ordering of songs for playlist recommendation. In *ECCBR 2006, 8th European Conference on Case-Based Reasoning*, September 2006.

[10] Lukas Bossard. Pancho – the music explorer. Swiss Federal Institute of Technology Zurich, Dept. of Information Technology and Electrical Engineering, April 2008. Semester thesis.