Contextual Personalized Re-Ranking of Music Recommendations through Audio Features Based User Preference Models

Boning Gong

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

Contextual Personalized Re-Ranking of Music Recommendations through Audio Features Based User Preference Models

by

Boning Gong

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science

at the Delft University of Technology,
to be defended publicly on Thursday 27th of August, 2020 at 10:30 AM.

Student number:	4367308	
Thesis committee:	Prof. dr. ir. G.J.P.M. Houben,	TU Delft, Chair
	Dr. N. Tintarev,	TU Delft, Supervisor
	Dr. C.C.S. Liem,	TU Delft, Committee

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

ABSTRACT

With advancements in Internet and technology, it has become increasingly easy for people to enjoy music. Users are able to access millions of songs through music streaming services like Spotify, Pandora, and Deezer. Access to such large catalogs created a need for relevant song recommendations. Music recommender systems assist users in finding the most relevant songs by consistently matching them with the user's preference. Accurately representing these preferences is essential to creating accurate and effective song recommendations. User preferences are highly subjective in nature and change according to context (e.g., music that is suitable for running is not suitable for relaxing). Preferences for songs can be based on characteristics of high level audio features, such as tempo and valence.

This thesis proposes a new contextual re-ranking algorithm, which belongs to the group of contextual post-filtering techniques, to leverage users' contextual information. The algorithm uses two models, a global and personalized model, to model user preferences. These models use audio features to represent user preference in specific contextual conditions. The algorithm is able to re-rank any given music recommendation list. First, we analyze the correlation between audio features and contextual conditions. This analysis shows that the correlations are significant, thus audio features are suitable for representing user preference in contextual conditions. Thereafter, we implement and evaluate the re-ranking algorithm using accuracy metrics on the #NowPlaying-RS and InCarMusic datasets, using various initial recommender algorithms. Results show there is merit in applying such a re-ranking algorithm to increase recommendation accuracy. The personalized model, given enough historical data, consistently outperforms the global model.

Keywords: Music Recommender Systems; Contextual Post-Filtering; Audio Features; Personalization

PREFACE

Having no idea during high school what subject I would enjoy studying, I started exploring a wide variety of studies and universities. After comparing studies ranging from economics to medicine and from philosophy to computer science, the latter became my fascination. Fascinated by the possibilities within the world of computer science together with the international atmosphere and reputation of TU Delft, I decided to pursue my higher education here. TU Delft did not let me down. Now, after six wonderful years it is time to close this chapter and start a fresh new chapter in life.

It felt like yesterday when I first approached Prof. Geert-Jan Houben, the chair of my thesis committee, to discuss possibilities of doing my master thesis within the Web Information Systems research group. Prof. Houben recommended me to speak with Dr. Nava Tintarev, my dear supervisor. During our first meeting, I had some potential thesis topics and ideas, but they were very vague. Luckily, Nava was very attentive. She listened to my ideas, helped me structure them and steered me in the right direction for my research topic. She can be strict at times, like when I had to revise my proposal for the umpteenth time, but this was for my own good. Thank you, Nava, for all the structure and valuable feedback you have given.

Nava introduced me to my daily supervisor, Mesut Kaya. My biggest gratitude goes to him. Without him, I would have been stuck and lost many times during my thesis. Our bi-weekly meetings always gave me a new boost of motivation and energy to continue my research. Thank you, Mesut, for providing direction, insightful knowledge and valuable suggestions and feedback. Next to this, I would like to thank Prof. Geert-Jan Houben and Dr. Cynthia Liem for both being part of my thesis committee.

In addition, I would like to thank all my friends. Thank you for creating all the great memories together during my time in Delft and for the support whenever I was down. Last, but not least, thank you mum and dad. Both for always supporting me, regardless of your opinions regarding my choices, and for providing me with all that you have so I can reach my full potential.

Thank you all!

Boning Gong
Delft, July 2020

Contents

List of Figures

List of Tables

1

INTRODUCTION

Music is a universal language, is sometimes said. Even though people speaking one language do not understand the lyrics of songs from other languages, they still can enjoy it. For many people, music is one of the major ways to entertain themselves. In the present day, with advancements in Internet and technology, music has become increasingly easy to access. Using a smart phone with internet connection, one can simply utilize streaming services, like Spotify, Pandora and Apple Music, to access millions of artists and songs from all around the world. With this many options, it has become increasingly difficult for users to discover new enjoyable songs. Many potentially interesting songs are yet to be discovered. With this trend, it has become increasingly more valuable and relevant for researchers to research music retrieval and recommender systems.

Recommender systems are systems that help users to deal with an overflow of information and options through filtering only the most relevant services, content and/or items. Music recommender systems aim to assist users in finding and discovering new music specifically. These systems require the ability to retrieve specific songs, that consistently match with the user's preference, from large music databases in an efficient manner.

1.1. RESEARCH MOTIVATION

Initially, music recommender systems mainly relied on similarity comparisons between songs and users to create recommendations [5]. These systems model the long-term preferences and provide solutions to long-term needs of users [6]. Accurately representing users'preferences is essential to the performance of music recommender systems [7]. Many additional methods have been developed more recently to further improve representation of user preferences.

One such way is to incorporate contextual information of the user [6]. Understanding user preferences requires intelligent preference learning techniques that are able to relate user context to concise and context specific music preference [8]. According to sociological and psychological research of music, people's short-term needs are often influenced by their current context, e.g. external factors, psychological state and/or the activities a user is engaged in [9, 10]. People often seek music for specific contextual situations, like events or emotional states, rather than based on artists or song content [11]. Music preferences of users change through influences from the physical environment, such as activities or geo-location [12]. People prefer different music when working out in a gym compared to reading a book on the couch, for example. This led to an emerging interest in contextual music selection and recommender systems [13].

In this work, we use three definitions to describe context. The first are contextual dimensions, which refers to specific categories of contexts, e.g. *time of day*, *activity* etc. The second are contextual conditions. A contextual dimension exists of multiple contextual conditions, e.g. *morning* and *afternoon* within *time of day*. Lastly, there are contextual situations, which refer to a combination of different contextual conditions that make up a situation of a user, e.g. *sunny* weather, while *jogging* in a *happy* mood.

Recommendation systems that use contextual information can be divided into 3 categories, namely contextual pre-filtering, contextual post-filtering and contextual modeling [14]. An in depth description is given for each in Chapter 2. Both filtering methods have the benefit over contextual modeling that no additional changes are required to the existing recommender system. For pre-filtering, only the input is adjusted, while

for post-filtering the recommender's output is altered [15]. Pre-filtering methods have been well developed, but post-filtering research efforts have been limited [16].

Contextual modeling is the most powerful method of the three [16]. However, it comes with its own challenges and limitations. For these contextual modeling methods, machine learning (ML) techniques have gained more focus among researchers recently. Increasingly these techniques have been applied to capture the relation between contexts and user preference. Examples include a context-aware collaborative music recommender that uses pre-trained neural networks where semantic tags are used as input by Liang et al. [17], a listening sequence-based context-aware music recommender that learns low dimensional representations of music through neural networks by Wang et al. [18] and a short-term music recommender built using a knowledge-based attentive recurrent neural network by Lin et al. [19].

There are two challenges when machine learning approaches are used to create recommendations. The first is the explainability of these recommendations to users. It is important to be able to explain why a certain song was recommended when users ask about it. The second one is regarding the explainability to the designers of the recommenders themselves. It is often complex, if not impossible, to fully understand how the model uses factors and weights, since many of them are created using black box algorithms [20]. These recommenders do not provide any insights in the relation between contextual information and user music preferences.

In this work, we propose a post-filtering approach: a contextual re-ranking algorithm, which ranks higher songs that are more suitable to a user's current contextual condition. It can be applied to any existing music recommender's output, is easily understandable and explainable, and works with any contextual dimension. Users like or dislike songs based on the characteristics of audio features, like tempo, vocal, instrument etc [21]. Significant correlations exist between music preferences expressed in audio features and personality traits [22]. Since user preferences are closely related to audio features, it should be a good predictor for user preferences in order to create accurate recommendations. Therefore, we use audio features to model user preferences for specific contextual conditions. To the best of our knowledge, our approach is novel, since it uses audio features to models users' context specific preferences. It is also important to realize that music preference is highly subjective; while one person may experience two songs as dissimilar, a second one may feel a high resemblance [8]. Moreover, the music one user may prefer in a given context may be different from what another user prefers in the same context (i.e., what is considered good morning music differs across users). That is why we compare a global preference model with personalized user preference models for the re-ranking algorithm.

1.2. RESEARCH QUESTIONS

We focus on re-ranking music recommendations by representing user preferences in contextual conditions through the usage of audio features. The goal of the re-ranking is to give songs which are more suitable to a given contextual condition relatively higher positions. In order to do this, we propose the following main research question:

- **Main Research Question:** How can the relation between audio features and contextual factors be used to improve music recommendation accuracy through re-ranking?

In order to answer the main research question, we propose the following three sub-research questions:

- **sub-RQ1:** How are contextual conditions of different contextual dimensions related to audio features?

The first sub-research question aims to give us more insight into how and which contextual conditions are related to specific audio features. This helps us to get a general idea of which audio features to include and how significant they will be when representing user preferences. Our hypothesis here is that there exists a significant correlation between certain audio features and contextual conditions. Since there are a wide spectrum of audio features and contextual conditions, it is difficult to hypothesize how exactly they are correlated.

- **sub-RQ2:** How does re-ranking, based on audio feature representations of user preferences in different contextual conditions, affect music recommendation accuracy?

The second sub-research question looks at the performance of the re-ranking and accuracy of the re-ranked recommendations compared to initial recommendations. The accuracy of recommendations are measured

using various ranking based metrics. Our hypothesis here is that if there is a strong enough correlation between audio features and contextual conditions, such a re-ranking algorithm is able to further increase the accuracy of recommendations made by existing recommender algorithms.

- **sub-RQ3:** How do global audio feature representations of user preferences in different contextual conditions affect the re-ranking results compared to personalized audio feature representations of user preferences in the same contextual conditions?

For the third sub-research question, we use two separate models as input for the re-ranking algorithm. The personalized model gives more weight to user specific preferences of audio features in certain contexts, while the global model combines all users'preferences in one model. Here, we expect that the personalized model will outperform the global model, since different users have different preferences in specific contexts. One user might enjoy sad music to complement rainy weather, while another user might prefer happy tunes to compensate it. The personalized model is, in such a case, a more accurate representation of user preference than the global model.

1.3. CONTRIBUTIONS

To the best of our knowledge, we are the first to create a straightforward re-ranking algorithm based on mappings of user context to audio features representing user song preferences. Through this research we contribute the following:

- An increased understanding of the underlying relationship between user contexts and audio features, especially how they relate to user preference.
- A comparison between global and personalized user preference models for different contextual conditions using audio features.
- A creation of a contextual re-ranking algorithm, belonging to the group of contextual post-filtering methods, that is easily explainable and is able to re-rank any music recommendation list.

1.4. THESIS OUTLINE

The rest of this thesis report is organized as follows:

Chapter 2 provides an overview of existing research and general background information related to our research. It gives an introduction of recommender systems, audio features and examples of how other researches used contextual information in their works. Furthermore, it compares existing research to our research and illustrates differences.

Chapter 3 presents our analysis and evaluation results of the correlation between audio features and contextual conditions. This chapter aims to address the first sub-research question.

Chapter 4 gives an overview of the methodology that we use to carry out our research. This includes gathering, evaluating and selecting datasets, experiment design and recommendation accuracy evaluation methods among others.

Chapter 5 presents our proposed re-ranking algorithm and elaborates on the global and personalized models that will be used in the experiment. The re-ranking algorithm addresses the second sub-research question, while the two different models are designed to address the third research question.

Chapter 6 elaborates on the whole pipeline of our experiment together with all choices that were made during this process. It also provides a detailed description of the re-ranking system and the results obtained.

Chapter 7 concludes our work, discusses limitations and provides direction for potential future work.

2

Literature Review

We describe in this thesis how user context can be leveraged through a re-ranking algorithm to increase personalization and improve recommendation quality. Before doing this, we will first describe the background and related work of context-aware (music) recommender systems. An introduction into recommender systems in general will be given first in Section 2.1 together with an overview of traditional recommender algorithms. Section 2.2 elaborates on the definition of context and how it will be used in our research. Thereafter, Section 2.3 explains what context aware recommender systems are, how they can be grouped and which context-aware recommender algorithms there are. A variety of context-aware music recommender systems are given in Section 2.4. Section 2.5 will describe how audio features have been researched and used within both general and contextual music recommender systems. This chapter concludes with an explanation together with a general algorithm for re-ranking in Section 2.6.

2.1. Recommender Systems

With the exponential increase of online services, tools and their accompanying data, solutions to find the right or interesting data has never been more relevant. Recommender systems are such search and decision tools which assist users in finding and discovering information and/or items. They require the ability to retrieve specific data, that consistently match with users' preferences, from large music databases in an efficient manner. This way, recommender systems help users overcome information overload [23]. Traditionally, recommender systems were classified into three categories, based on how the recommendations are made. Adomavicius et al. describe them as follows [6]:

- *Content-based recommendations:* Recommendations are based on the similarity of items compared to the items the user liked previously
- *Collaborative recommendations:* Recommendations are based on the preferences of users that are most comparable to the current user
- *Hybrid approaches:* Any mix of the above mentioned methods

2.1.1. Traditional Recommender Algorithms

These traditional systems rely on recommender algorithms based on underlying user and or item similarity/preferences. There are various ways to calculate these similarities and preferences and make recommendations based on the output. A variety of these traditional recommender algorithms is listed below. Some of them are used to produce the initial recommendation list in our experiments later on.

Item/User Average

This algorithm is the straightforward, computationally cheap and the most simple one. It basically takes the average rating within the training set based on a given user or item. Item Average takes the average rating that all other users have given to a song that needs to be predicted. This average rating will be used as prediction, independent of the user that it is recommended to. In the same way, User Average uses the average rating

of all items that a user have rated previously, independent of the songs that are recommended. A formal expression would be:

$$\hat{r}_{ui} = \frac{1}{n} \sum_{k=1}^{n} r_k \tag{2.1}$$

Where $\hat{r}_{ui}$ represents the predicted rating for song i to user u. n represents the amount of available ratings in the training set for song i in the case of Item Average and the amount of available ratings in the training set for user u in the case of User Average. r_k then represents each rating given to song i for Item Average and rating given from user u for User Average.

These algorithms are included here because of their simplicity. If the re-ranked list does not have an increased accuracy compared to these outputs then the chance is small it will work for outputs on more sophisticated recommendation algorithms.

USER/ITEM KNN

One of the earliest and more traditional recommendation algorithms are the User and Item k-nearest neighbours algorithms. The UserKNN was first mentioned by Resnick et al. [24]. For a given user, item and context, this algorithm will look for all users who already rated the test item before. For each of these users, the similarity to the given user is calculated. Often a similarity measure, like the Pearson correlation coefficient, is used based on all the ratings of the users. The algorithm then sorts these users by their similarity. Using a weighted average rating of the given item for the k, which is a tweakable parameter, most similar users, a prediction is calculated.

ItemKNN was first mentioned by SarWar et al. [25]. It works in a comparable fashion as the UserKNN algorithm. ItemKNN looks at the set of all items that the test user has rated previously. In this list, the similarity of previously rated items to the test item is measured. Also here any similarity measure can be used. The weighted average rating of the k most similar items will then be used as prediction for a given test user and item.

MATRIX FACTORIZATION

Matrix factorization is an unsupervised learning method that received more exposure recently recently [26, 27]. It uses a user-item interaction matrix where rows are representing users and columns representing items. User-item pairs can have a value, which represents the rating that has been given by that user to a specific item. This matrix will then be factored to a joint latent factor space with dimensionality f. This space is often represented by two sub-matrices , one representing items and the other representing users. Individual items and users are represented by vectors, which contain the estimated latent factors, in these sub-matrices. The inner products of user-item vectors from the two matrices model the interactions between them. Say we have a vector $q_i \in \mathbb{R}^f$ representing item i and a vector $p_u \in \mathbb{R}^f$ representing user u, the dot product $q_i^T p_u$ tries to capture the interaction between user u and item i. The approximated rating in this case is represented by $\hat{r}_{ui}$ and gives us the following formula:

$$\hat{r}_{ui} = q_i^T p_u \tag{2.2}$$

As we can see from above, obtaining the rating approximation is fairly easy when the two matrix representations of users and items are known. The biggest challenge is to accurately create them from the user-item interaction matrix. Singular value decomposition (SVD), a well-established technique, is closely related to such modelling [28]. However, since the initial user-item interaction matrix is often sparse, applying conventional SVD is rather difficult. It is undefined when input from the matrix is incomplete. Also, the danger of overfitting arrives when only applied to a few entries. In order to estimate all user vectors p_u and item vectors q_i, the regularized squared error for the known entries need to be minimized:

$$\min_{q,p} \sum_{(u,i) \in K} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{2.3}$$

The set of user-item pairs is represented by K, r_{ui} is the known rating of user u to item i and λ controls the extend of regularization, which is normally determined by cross-validation.

These are the very basic components of matrix factorization based models. We will proceed to describe three specific matrix factorization models next.

STOCHASTIC GRADIENT DESCENT BASED MATRIX FACTORIZATION

The first is the so called stochastic gradient descent (SGD), which uses a specific approach to minimize the error of Equation 2.3. This approach iterates over all given ratings in the training set, predicts r_{ui} and computes the prediction error for it by subtracting the prediction from the actual rating. The next step is to adjust the parameters in the opposite direction of the gradient. The update of the user and item vectors are thus as follow:

$$q_i \leftarrow q_i + \gamma(e_{ui} * p_u - \lambda * q_i) \tag{2.4}$$

$$p_u \leftarrow p_u + \gamma(e_{ui} * q_i - \lambda * p_u) \tag{2.5}$$

SGD generally is a popular approach [29–31] that is relatively quick. Still, in some situation, it might be better to use the alternating least squares optimization method, which we will describe next.

ALTERNATING LEAST SQUARES BASED MATRIX FACTORIZATION

Alternating Least Squares (ALS), like SGD, is a method to minimize the error in Equation 2.3. Both q_i and p_u are unknowns that need to be optimized. If one of them would be fixed, the problem can be solved optimally. This is what ALS does, it rotates between optimizing the item vectors and optimizing the user vectors. If item vector q_i is fixed for example, this method will optimize p_u by solving the least-squares problem and the other way around [32]. So alternately, the whole of Equation 2.3 will be optimized until some convergence criteria is met.

Generally, SGD is faster and easier to use compared to ALS. However, there are two specific scenarios where ALS is probably a better candidate. The first case is when parallelization is an option and preferred. Since only one unknown variable is being solved at a time, it gives potential for parallelization of the algorithm [33]. The second case is when implicit feedback, which will be described in more detail later in Section 4.1.2, data is used. These datasets are not as sparse explicit feedback datasets. So looping over each given entry, like SGD does, would not be optimal [34].

BIASED MATRIX FACTORIZATION

A benefit of matrix factorization is that it is flexible when dealing with different data aspects. It is based on the underlying framework to model users, items and their interactions. However, often the interaction between users and items alone is not sufficient enough to estimate an accurate rating. There will be variations in ratings because of specific effects from certain users and/or items, which are known as biases. These biases are independent of interactions, but dependent on specific users/items instead. Some users tend to rate items higher generally than others, for example. The same applies to some popular/high quality items compared to unpopular/low quality items.

So only using the interaction of $q_i^T p_u$ is not sufficient enough. The accuracy can be increased by identifying and adding these biases to the prediction model. Biased matrix factorization (BiasedMF) is such a matrix factorization method [28].This method incorporates a first-order approximation of the bias. For a given rating r_{ui}, where u represents a user and i an item, the bias is defined as follows:

$$b_{ui} = \mu + b_i + b_u \tag{2.6}$$

where μ is the general average rating, b_i is the observed deviation of item i and b_u the observed deviation of user u. Here the deviations are measured from the general average. For example the average rating of all songs of all users is a 3.2. A very popular song has an average rating that is 1 higher than the general average, but a very critical user tends to rate songs 1.5 lower than the general average. The estimate for this song for this user would be 3.2 + 1 - 1.5, which results in a 2.7. So in this method Equation 2.2 becomes:

$$\hat{r}_{ui} = q_i^T p_u + \mu + b_i + b_u + \tag{2.7}$$

Where all the symbols represent the same elements as described previously above. In a similar matter, Equation 2.3 becomes:

$$\min_{q,p,b} \sum_{(u,i)\in K} (r_{ui} - p_u^T q_i - \mu - b_u - b_i)^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2) \tag{2.8}$$

BAYESIAN PERSONALIZED RANKING

Unlike the methods described above, Bayesian Personalized Ranking (BPR) is a technique that is suitable for both explicit feedback and implicit feedback datasets. This algorithm is different from the above mentioned ones since it is optimized for item ranking and not for item prediction [35]. When an implicit feedback dataset is given, there is no negative feedback in the data, only interactions of users with certain items. This model assumes that if a user has interacted with an item, he/she prefers it over other not interacted items. When multiple items have been interacted with by the same user, preference is assumed to be the same. This is also true for all items that the user has not interacted with.

Furthermore, it uses item pairs as training data and the optimization is based on giving these items pairs a ranking as correctly as possible. This in contrast to above methods which try to predict ratings for single items. The models optimization criterion is based on Bayesian analysis using a likelihood function. For a more detailed description and how it can be applied to both matrix factorization models and kNN, please see [35].

2.1.2. ADDITIONAL RECOMMENDER SYSTEMS

So traditional recommender systems use algorithms that compare similarities between items, users or a combination of it to create recommendations [5]. These systems model the long-term preferences and provide solutions to long-term needs of users [6]. However, in order to further increase the relevance of the recommendations, more data regarding the user and/or items would need to be incorporated [6]. Reviewing subsequent and more recent research, multiple new types of recommender systems have been developed. Lu et al. classifies recommender systems into eight types for example. The additional five they identified are [36]:

- *Knowledge-based recommendations:* Recommendations are based on knowledge about items, users and/or how they are related to each other
- *Computational intelligence-based recommendations:* Recommendation models are based on specific computational models, e.g. Bayesian models, artificial neural networks, genetic algorithms etc.
- *Social network-based recommendations:* Recommendations are based on the user's interaction with other users within the platform
- ***Context awareness-based recommendations:*** Recommendations are based on contextual features of the users, e.g. location, emotions, activity, weather etc.
- *Group recommendation techniques:* Recommendations are based on a group of users in a situation where the overall preference is not clear

Our research is closely related to the group of context awareness-based recommender systems, but before elaborating further on this specific recommender system, we first need to have a clear definition of context.

2.2. DEFINITION OF CONTEXT

In order to be clear about what context exactly is, we need a consistent and overarching definition of "Context". Different researchers have used different definitions. In the earlier days, context was mainly defined by example. Schilit and Theimer used a combination of location, identity of people nearby, objects and changes to these objects as context [37]. Ryan et al. used location, environment, identity and time as a user's context [38]. But these are often hard to apply and not general enough.

Other researchers tried to define context as the environment or situation. Some only consider the environment of the application, some only of the users and others use a combination of them. Brown defined it to be all the features of a user's environment that the application knows about [39]. While Hull et al. defined context to be everything of the current situation, including the whole environment [40]. These definitions in turn are too general to be used. They provide little guidance on how to use the separate elements of context.

A definition that is useful should not be too general nor too specific. The definition that we will be using in this research, and is widely cited, is given by Dey et al. [41]:

Context: *"Any information that can be used to characterize the situation of entities (i.e., whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves."*

For our research, we will be focusing on the situational information of the user. This includes, but is not limited to, *time of day, day of week, season, weather, mood, social company, location, activity* etc.

2.3. CONTEXT-AWARE RECOMMENDER SYSTEMS

Now we have a definition of context, it is time to take a closer look at what context-aware recommender systems are exactly. These systems are different from normal recommenders, since they also incorporate all kinds of contextual information.

Specifically, traditional recommender systems utilize a prediction function F to estimate ratings:

$$F : User \times Item \rightarrow Rating \tag{2.9}$$

where *User* and *Item* are input variables for the prediction function F that will output the estimated rating. This is also known as a 2D approach to recommendations [42].

Since context-aware recommender systems have context as additional input, the function F will become:

$$F : User \times Item \times Context \rightarrow Rating \tag{2.10}$$

where *Context* can represent one more multiple contextual dimensions. For example, the context can just be *(reading)* or *(Sunday, afternoon, sunny, happy)*.

We can generalize the above as:

$$F : User \times Item \times C_1 \times ... \times C_n \rightarrow Rating \tag{2.11}$$

where n represent the amount of different contextual dimensions used as input and each C_i stands for the specific value of that contextual dimension. This would bring us a $n+2$ dimensional problem.

These new dimensions can be embedded in different ways using different methods. Based on how exactly context is used, three forms can be distinguished according to Adomavicius and Tuzhilin, namely *contextual pre-filtering, contextual post-filtering* and *contextual modeling* [42].

In the three subsections below we describe each of the different types of contextual recommender systems together with examples, benefits and drawbacks. The fourth subsection focuses on context-aware recommender algorithms that are used in these systems.

2.3.1. CONTEXTUAL MODELING

Contextual information is directly being incorporated into the model. This means that the interactions between the contextual dimensions, users and items are being captured within the recommendation model. An additional drawback with these models is that it is often hard to explain how or why exactly a recommendation is made. Multiple recommendation approaches have been proposed using either heuristics or predictive modeling techniques, which we will describe below.

Matrix factorization is an example of an upcoming approach that has gained attention because of the Netflix Prize competition [28]. It uses dimensionality reduction techniques to predict the missing ratings within a given matrix. This is done through transformation of given information to a latent factor space [43]. Karatzoglou et al. used matrix factorization and embedded different contextual factors [44]. They did this by modelling the $Users \times Items \times Contexts$ as a n-dimensional tensor. This would result in a compact model that can be used to predict contextual ratings. The benefit here is that any amount of contextual dimensions can be incorporated. This, however, also comes at a cost, for a big amount of contextual dimensions, the parameters will grow exponentially. It requires the values to be discrete instead of continuous.

There are also other recommendation techniques that have been proposed specifically for context-aware recommender systems. Oku et al. proposed to embed the contextual features directly into the recommendation space using machine-learning methods [45]. They specifically use support vector machines (SVMs), which basically splits the items of users into a group of preferred and not preferred items in various contexts. This n dimensional space is then used to create a separating hyperplane, which is optimized to maximize the distance between the two sets of items. This plan is then a classified for all future item input.

Twardowski proposed a session-aware recommender system, one where no straightforward user information is needed [46]. The recommendations are made based on the current user session, which is represented as a sequence of events. Interactions between different variables within a session and items are explicitly modelled using matrix factorization. After that, both Recurrent Neural Network (RNN) and Feed Forward Neural Network (FFNN) are used to make predictions for a given session. The RNN captures dependencies

between different events in sessions, while the FFNN estimates ranking scores. It uses the output of the RNN, which is a representation of a session context, together with the items as input. The final recommendation list is created through minimizing the loss in the network.

Wu et al. proposed a graph based model that incorporates contextual information in the creation of recommendations [47]. In this model, the context-aware recommendations are represented by a searching problem, in which relevant items need to be found for a given user in a given context graph. Within the graph, the vertex set is divided into sets of users, items, attributes and contexts. Edges represent connections between different sets. So for example the edges between users and items represent the interactions that users have had with those items. An adjacent matrix is used to represent the context graph. In the end, recommendations are made based on a combination of items and users that are closest to the given user. This approach resembles a combination of item-based and user-based collaborative filtering.

2.3.2. CONTEXTUAL PRE-FILTERING

Contextual information is leveraged by selecting only the relevant set of data for the given context. Only this relevant data is used as input for the recommender system, other data is discarded. Predictions are still begin made by the traditional 2D recommender system.

Pre-filtering has the major advantage of being easily integratable with numerous traditional recommendation techniques [42]. This gives the advantage of having a cheap, easy and quick improvement for existing commercial recommender systems. When applying pre-filters, the selected context basically serves as a query/filter for selection of the most relevant training data. For example, only the previous ratings for a user when watching a movie on Friday evening is used when recommending what movie to watch on a Friday evening for a movie recommender system. This particular example uses an exact pre-filter, since an exact context has been used as filter.

This also gives some drawbacks. Using an exact context that is too specific can be too narrow. For example, a movie on Friday evening, with friends, in the cinema. This would be *C=(Friday, Friends, Cinema)*. It is very probable that the previous rating data of this specific context is very sparse or even non existent. Or perhaps some contextual factors add more noise than value. In order to tackle this problem, Adomavicius et al. [6] proposed a generalized pre-filter approach. In this approach, specific contextual situations are being put together in more general ones. So for example, instead of using *Monday* to *Sunday*, one can use *Weekday* and *Weekend* as contextual values for weekday.

As seen in the previous examples, contextual pre-filtering will build more local models, rather than global models to make predictions based on all available data. Focusing only on data that is relevant to the context of the current user, a prediction model is created to make more local predictions. Researchers who apply this, should pay attention whether such a local model outperforms a more general global model.

Baltrunas and Ricci [48] proposed a different way of pre-filtering. Instead of removing data, they used their so called *item-splitting* approach, where items in the dataset are split into two or multiple fictitious sub-items based on the different contexts in which they can be found. This means that the traditional 2D recommender systems have a more fine grained dataset as input. Similar to this, Baltrunas and Amatriain [49] proposed *microprofiling*. The approach is the same as *item-splitting*, but instead of splitting up items, users will be split up into several sub-user profiles. Each profile representing the user with his/her preferences in a specific context. Subsequently, predictions are based on one or multiple of these sub-user profiles and the given context, instead of one global user model.

2.3.3. CONTEXTUAL POST-FILTERING

Contextual information is leveraged by either removing or re-ranking the resulting recommendations based on their relevance for the given context. So initially, the whole dataset and the traditional 2D recommender systems are still being used. Here we have the same advantages as with the pre-filter approaches. No change is needed to the traditional recommender systems. Post-filtering just makes adjustments to the obtained recommendation list. This can be done through either adjusting the order of the recommendations or through removal of irrelevant recommendations [42]. An example would be that whenever a user is eating dinner he or she only listens to jazz music, the post-filter can then proceed to remove all non jazz related songs within a music recommender system. Contextual post-filtering can be classified into model-based and heuristic techniques. For model-based, probabilistic models are created in order to calculate the probability of a user liking an item within a specific context. This probability is then used to adjust the initial recommendation list. On the other side, we have heuristic approaches, which looks for similar item characteristics or attributes for a user in a specific context. These attributes are then used to adjust the ranking [50].

Panniello et al. [51] carried out an experiment in order to evaluate the performance of exact pre-filtering against *Weight* and *Filter* based post-filtering. Whereas *Weight* weighs the relevance for each prediction based on a specific context and adjusts the order, *Filter* simply removes recommendations based on lowest relevance. For this, they used data from the e-commerce domain. Their results showed that the Weight post-filter outperformed the exact pre-filter, which in turn outperformed the Filter post-filter. So the conclusion is that neither post- nor pre-filter is always the dominant approach. The best choice is highly dependent on the data, application and context.

Taneja and Arora proposed their own clustering based contextual post-filtering approach (Clu-PoF) [52]. T-mean tests have been carried out first to find the most relevant contextual features. Those were then used to create clusters using a hierarchical clustering approach. Clusters that are contextually similar are combined, since the ratings in those contexts will also be similar. The prediction is then done through weighted average of ratings of similar users in the same or most comparable contexts. Movies that are irrelevant or less relevant in a cluster will be filtered out as last step.

2.3.4. CONTEXT-AWARE RECOMMENDER ALGORITHMS

Above we divided context-aware recommender systems into three approaches with a few examples from other works for each. These systems have their own context-aware recommender algorithms, which differ from traditional recommender algorithms since they rely on contextual input for their output. An overview of these algorithms is given below.

ITEM-CONTEXT/USER-CONTEXT AVERAGE

These algorithms are basically the contextual versions of the Item and User Average algorithms in the initial recommender algorithms list. Item-context Average will return the average rating per item per context, while User-context Average does the same but per user and context instead. These algorithms are still fairly simple, computationally cheap and at the same time use contextual information. Even though all these algorithms are simple, they still provide the chance to evaluate whether the re-ranking algorithm adds some kind of performance increase.

CAMF-C/CAMF-CI/CAMF-CC/CAMF-CU

CAMF stands for Context-Aware Matrix Factorization and is an extension of the classical Matrix Factorization approach [27] in which context is included. Baltrunas et al. [53] came up with different flavors of CAMF models based on different levels of the interaction of context with ratings.

The first, and most general one, is CAMF-C, where the C stands for context. Here, the assumption is made that each contextual condition has in some way an influence on the ratings. So it does not matter what item is being predicted, the same contextual condition will have the same influence on all items. This is done by introducing one single parameter for each contextual condition and models the deviation for each.

The second, with a more middle complexity compared to the other three, is CAMF-CC, where the CC stands for context-category. Here parameters are introduced for each contextual condition and item category. This means that the assumption here is that specific contextual conditions, e.g. sunny, have a different influence per item category, e.g. pop songs or piano songs. Thus the parameter here will also be used to model a deviation like in CAMF-C.

The last two versions are CAMF-CI and CAMF-CU, where CI stands for context-item and CU for context-user respectively. These two are most granular and thus the most complex models. For both, a parameter is introduced for each contextual condition and item pair. So sunny would for example have a certain influence on one song, but can have another on another song. These models both introduce many new parameters and are therefore computationally more heavy, but it will also give increased performance when the context directly influences items.

CAMF_LCS/CAMF_ICS/CAMF_MCS

Three other types of context-aware matrix factorization methods are described by Zheng et al. [1]. These three alternatives use similarity-based contextual modeling to compare the similarity between contextual situations. They argue that the more similar two conditions are, the more similar the recommended lists of items should be. In [1] they describe how a similarity-based approach can be applied to matrix factorizations.

The similarity between contextual situations, all of which have at least 1 valid condition, are estimated based on correlation. The representation or model of these similarities have an impact on the performance. So it was necessary for the researchers to try and evaluate multiple ways to represent situation similarity.

Independent Context Similarity (ICS) only measures the similarity between conditions when they are taken from the same dimension. So for example, the similarity between morning and evening is measured, but not the similarity between morning and sunny. The former are both taken from the contextual dimension of time of day, while sunny in the latter comes from the weather dimension. So here an independent relation between conditions is assumed. Zheng et al. chose to take the product of all similarities per condition to represent the final similarity between two distinct situations.

Latent Context Similarity (LCS) deals with the sparsity problem of contextual datasets. Contextual situations often have only a few ratings, especially when they are made up of multiple dimensions. So a situation may occur where the test set requires a similarity calculation of a new pair of conditions, but where the training set has not seen this combination. So what Zheng et al. [1] did was to use vectors of weights to represent each condition, where the weights represent latent factors. In their experiments they chose to use 5 of such weights for each representation. The dot product is then used between two vectors to represent the similarity of a pair of conditions. So now, even when a new pair shows up, the weights of the vectors that represent each of the two unique conditions are learned and updated over existing pairs. So the similarity of this new pair can easily be calculated by taking the dot product between the two vector representations. This model introduces extra parameters that needs to be optimized, so it is more computationally expensive than ICS, but it also provides more flexibility.

Last, but not least, there is Multidimensional Context Similarity (MCS). Here a multidimensional coordinate system is used to represent dimensions. Figure 2.1 shows such an example. Values for each condition are parameters that needs to be optimized in the learning process. In Figure 2.1, for example, the conditions of family and kids are assigned a new value. As one can observe, their change of position will also influence the distance between other conditions. A limiting range of [0,1] is used to make sure that conditional representations are comparable. So one can see this representation of situations as a cube, where the length of each side is 1. The similarity is then calculated through the inverse of distance between two points. Zheng et al. proceeded to use the Euclidean distance, but other distance measurements can be used as well [1].

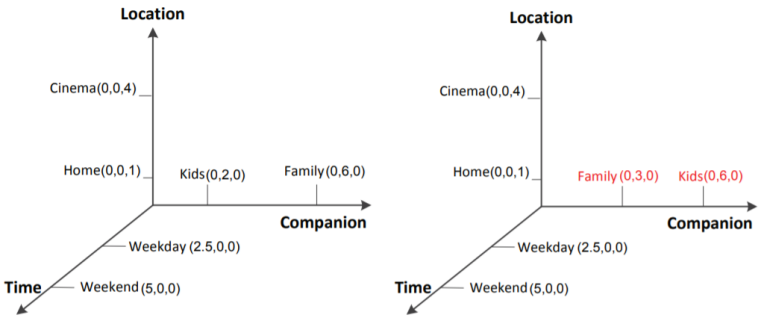


Figure 2.1: Representing contextual dimensions through the multidimensional coordinate system [1].

USERSPLITTING/ITEMSPLITTING/USER-ITEMSPLITTING

UserSplitting (US-x) is a type of contextual pre-filtering, first introduced by Said et al. [54], where the context is used to direct data selection. In this case, context-based user micro-profiles are created for each user and contextual condition. This technique is based on the idea that different people have different tastes when influenced by different conditions. That is why separate user profiles are created for each contextual condition. Since pre-filtering only affects the input data, any traditional recommender algorithm can and needs to be applied to it afterwards to get a recommendation list as output. The performance generally is shown to be better because of the more granular input compared to traditional collaborative filtering methods.

ItemSplitting (IS-x) uses exactly the same ideas of UserSplitting, but in this case items are split into item micro-profiles for each context. Also this is then proceeded to be fed into a regular recommendation algorithm.

Last, but not least, there is User-ItemSplitting (UI-x), in which both users as well as items are split into micro-profiles. Both splits are based on contextual conditions.

2.4. CONTEXT-AWARE MUSIC RECOMMENDER SYSTEMS

Multiple context-aware recommender systems have been developed for the task of music recommendation. This section describes the contextual factors that were taken into account, the exact method and their performance for the ones that have been evaluated. They have been categorized by the contextual factors that the authors utilized.

TIME BASED

Baltrunas et al. built a time-aware recommender system that is able to recommend accurately a song or artist for users [49]. The system partitions user profiles into micro profiles, that represent the user in specific time spans. In their experiment, they used different time spans, ranging from hourly (even and odd hours) to yearly (cold and hot seasons). The data was gathered from the Last.fm service[1], where each track also contained the time stamp of when the user played this specific track. Recommendations are made based on multiple micro-profiles for the time span that the user is in, instead of using a general user profile. Their system outperformed the baseline (context-free) prediction algorithm when evaluated, especially the daily and hourly time spans gave the best improvements.

Dror et al. modelled music ratings using temporal dynamics and item taxonomy [3]. For their work, they utilized the Yahoo! Music service to collect user data over a whole decade. This data contained dates and one-minute resolution timestamps, which allows for temporal analysis. In order to capture the temporal effects, they incorporated session factor vectors in the user representation. These were learned by fixing all other factors and making SGD iterations over only the ratings in the current session. Next to this, they also modelled the user session bias. During the evaluation, embedding the user session bias showed a significant improvement in lowering the root-mean-square error.

MOOD BASED

Rho et al. used support vector regressions (SVR) in order to classify mood and embed it in a recommender system [55]. First they created a user mood map based on music that the user listens to. Seven musical features are extracted and analyzed for each song. A SVR model is then trained and used to determine the mood of songs. Based on the predicted current mood, an ontology-based recommender, which uses a context knowledge repository, is used to embed this. In their experiment, they reached 87.8% accuracy when using their trained mood mapper. Unfortunately no user study was done to evaluate the model.

Andjelkovic et al. created MoodPlay, a hybrid recommender system that uses an interactive interface which integrates user mood and music content [56]. Their model works as follows, first, an offline computation is done to calculate artist similarity, based on both audio content as well as mood. For the mood of artists, the Rovi mood metadata was used[2] and for the distance metric the Euclidean distance was used. During a user session, a general mood is calculated for each user based on recent artists that users listened to. Then, artists that are closest to the general mood, ranked by distance, are recommended. Last, but not least, users have freedom to move through their affective space within MoodPlay. The system keeps track of this and applies a decay function to the preference trail when new recommendations are made. Also here, no evaluation regarding recommendation quality or user satisfaction was done.

Shan et al. built an emotion-based recommender system based on an emotion model extracted from film music [57]. First, film segments are analyzed for both audio features as well as emotion. Emotions are detected using a mix of caption, speech, sound effect and visual features. Then, specific audio features are extracted and used to represent the emotion. This was done through affinity discovery. For this, Mixed Media Graph (MMG) was adopted and modified, an affinity graph algorithm. The original MMG was proposed by Pan et al. [58] and was used to find correlations between different media in a collection of multimedia objects. For their experiment, they took 20 animation films and built an emotion model based on it. After that, a set of top 10 of songs were created for each test song, based on the affinity graph that was constructed from the training set. Performance was measured based on the similarity of the emotions of the returned songs and the emotion of the test song. According to their proposed metric, the model performs well. However, also here no evaluation regarding user satisfaction was done.

LOCATION BASED

Kaminskas et al. built a location-aware recommender system to recommend suitable songs for places of interest (POIs) [59]. They manually gathered 3 songs per each of the 123 artists for 25 POIs of 17 major cities

[1] https://www.last.fm/
[2] Rovi API: http://developer.rovicorp.com/docs

through their knowledge-based technique. In their experiment, they compared five different approaches to recommend music for specific POIs. These approaches are the regular genre based recommender, a manual and automatic tag-based recommender, a knowledge-based recommender and a hybrid recommender that is a combination of the knowledge-based and auto-tag-based recommenders. Both tag-based and knowledge-based methods outperformed the straightforward genre-based method, however the clear winner was the hybrid method.

Cheng and Shen built Just-for-me, an adaptive location-aware social music recommender [7]. They extended the three-way model of Yoshii et al. [60] to incorporate location context and music popularity as well. The model uses a set of latent topics to model music content, under specific user's preferences, to location context. For their experiment, they used a mix of datasets that were extracted from different sources (Twitter, Youtube, Grooveshark and Last.fm). Whenever the location was missing, so called pseudo-observations were used. Machine learning methods inferred context for each user-item observation, by using a classifier that has been trained on existing location data. They compared their system against two other systems, one which recommends tracks randomly and the other is a simple contextual post-filter approach, described by Adomavicius et al. [42]. Evaluations showed that the Just-for-me system greatly outperformed both systems.

Schedl et al. created six different recommender systems, of which four used geospatial data of users [61]. For training and evaluation purposes, they used a data set of listening activities inferred from microblogs, the so called MusicMicro dataset. This dataset contains also longitude, latitude, country-id and city-id, next to the regular songs and users. Their GEO model solely uses the geospatial distance of users to calculate proximity. Afterwards, artists of the user's closest neighbors are recommended. Next to this, they also have the GEO-CF, CF-GEO-LIN and CF-GEO-GAUSS models. GEO-CF simply takes the union of the GEO and a standard collaborative filtering (CF) model. The other two models are based on CF and weigh users using either a linear or exponential geospatial distance measure over their geospatial distances. During their evaluation, the GEO-CF model appeared superior to all other five models.

WEATHER BASED

Sen and Larson uses Fuzzy Logic models together with raw contextual sensor data in order to recommend music [62]. Their idea is to model the user's context through data gathered from sensors and to create recommendations based on it without using any previous listening behavior. The sensors measure location, indoor/outdoor, activity, date, time and weather among others. The weather information in this case is mainly used to determine the mood of a user. The objective is to get a general idea depending on the impact of the weather on a user's mood. Subsequently, two categories are created, atmospheric-based and situation-based metrics. Queries are generated for each and expanded using the Last.fm API. These tags are being used as queries to retrieve songs from SoundCloud, which are then uploaded to EchoNest to do audio analysis and pick the right songs to recommend. In their future work they state to develop the system and evaluate it, however to the best of our knowledge, it has not been evaluated yet.

Baltrunas et al. built a context-aware music recommender that is specifically adapted to users traveling by car [63]. For this purpose, they have built a mobile application (Android) that recommends items based on initial input of the users through a Web based application. For each of their contextual factors, including weather, they did an evaluation for impact to different music genres using a probability distribution based on the user inputs. After this, they selected the most relevant contextual factors and these were embedded into a matrix factorization model. For their test data, their personal MF proved to give an improvement of 7% over the baseline model and 3% over the personalized model.

ACTIVITY BASED

Dias et al. built Improvise, a model that recommends personalized songs based on user input and feedback regarding their activities [64]. First, they associated specific genres and songs with activities through a user study. After that, the audio features are extracted with help of EchoNest. The generic model then proceeds to use the top 100 songs for each activity and their median 20% of standard deviation for the audio features in order to create hyper-rectangles. This means recommendations can created for users in certain situations, even without having any historical behavior. The personalized model works the same way as the global, except that the top 100 songs are continuously updated based on the new songs played by the current user. In their evaluation they showed that users tend to select more songs from the personalized model compared to the generic model. Also, the user satisfaction generally is higher for the personalized model.

Wang et al. created a music recommender system using a probabilistic model based on collected sensor data of mobile phones [65]. The first part is inferring the context of a user based on the sensed data. For this

they used a probabilistic distribution over all possible contexts. The second part models how suitable a song is for a given context. This adaptive model is based on the relationship between audio features and context and evolves accordingly with the user's behavior, e.g. listening the full song means positive feedback.

2.5. AUDIO FEATURES IN RECOMMENDER SYSTEMS

Since our research will be focused on how audio features are related to contextual factors, it is important to understand which and how audio features are used in existing research. This section will describe what and how specific audio features are used in both existing general music recommender systems and context-aware music recommender systems.

2.5.1. AUDIO FEATURES IN TRADITIONAL MUSIC RECOMMENDER SYSTEMS

Li et al. [66] used a combination of lower-level audio features in order to capture a more accurate similarity comparison between items, which further improves the recommendation accuracy. As lower-level audio features they considered mel-frequency cepstral coefficients (MFCC), Centroid, Roloff, sum of scale factor (SSF), Flux, Pitch and Rhythmic. In the end, a combination RollOFf, Rhythm and MFCC showed the best performance. Li et al [67] proposed a new collaborative music recommender system (CMRS) based on clustering of audio features. As audio features, two feature sets are used, rhythmic content and timbral texture. Timbral texture features contain the spectral centroid, flux, rolloff point, SSF and the first five MFCC coefficients, while rhythmic content features contain six features that are extracted from the beat histogram using Gorge's method [68]. Bogdanov et al. [69] use semantic music similarity models to create improved recommendations. In order to do this, they calculated low-level audio features for each track using their in-house audio analysis tool called Essentia. Over 60 features were extracted, including inharmonicity, tristimuli, spread, skewness, kurtosis, crest, MFCCs, key strength, pitch, BPM etc. These are used to infer semantic descriptors, which are used again for the classifiers. For a full list, please see the paper. Lu and Tintarev [70] used a total of six attributes, of which the two audio features key and tempo, in their approach of re-ranking recommendations to adjust for more diversity and improve accuracy using personality traits. They used Spotify to gather all attributes. A brief explanation is given about the mixed composition of the audio features energy, which makes it unsuitable for their research since it is unclear how different personalities impacts each sub feature within energy.

2.5.2. AUDIO FEATURES IN CONTEXT-AWARE MUSIC RECOMMENDER SYSTEMS

Cheng and Shen [7] created a novel recommender system called Just-for-me that uses a unified recommendation model that integrates audio features together with contextual factors. For the analysis of audio features, tracks are represented by bag of audio words. These so called audio words are generated using a combination of the first 13 MFCCs, the zero-crossing rate (ZCR) and pitch. Kaminskas et al. built a location-aware recommender system to recommend suitable songs for places of interest (POIs) [71]. One of the used models was an automated tag-based recommender system. The auto-tagger in this model uses a set of audio features that are defined by the block-level framework (BLF) [72], which uses overlapping blocks of the Cent spectrum audio representation. Chen et al. [73] analyzed the relation between emotions, through user-generated text, and music through factorization machines. They embedded 53 audio features, including loudness, mode, tempo and danceability that were extracted using the EchoNest API. Schedl et al. [61] combined music context and music content in a hybrid model. For their audio features, they used various rhythmic patterns that are extracted from the audio signal. These include onset patterns and coefficients, timbral features like MFCCs and two custom descriptors for attackness and harmonicness. These are used to estimate similarities between songs.

Novelty. Our work belongs to the group of contextual post-filtering approaches, as explained in Subsection 2.3.3. Similarly to previous approaches, our approach uses a similar weighing function as *Weight PoF* [51]. However, instead of using a rating probability of relevance based on similar users, we use context specific audio feature representations to measure similarity. Where Lamche et al. [74] create context models around items, we do the opposite by creating audio feature models around contexts. Furthermore, unlike [7, 73], our representation does not use any matrix factorization techniques or create any other latent spaces. Instead, we use a simple vector representation which allows for straightforward distance measurements when comparing songs to contexts. Our novel approach gives an interesting performance comparison to the more complex

latent models. Also, unlike Cheng et al. [7], we use readily available and standard audio features of Spotify [3] instead of creating our own based on low-level audio analysis. This makes it easier for other researchers to use the exact same audio features for other research by just calling the Spotify API.

2.6. RE-RANKING

As mentioned in Section 2.3, traditional recommender systems output a list R of top-N recommended items for a given user u based on predicted ratings. Often N is a relatively small number (<100), since real world users are typically only interested in several most relevant items. These items are selected and ranked according to some measurement. In a more formal notation, item i_x is ranked higher than item i_y ($i_x \succ i_y$) if $rank(i_x) < rank(i_y)$, where $rank : I \rightarrow \mathbf{R}$ is a function that represents the measurement as ranking criterion.

Given such a list and ranking, re-ranking approaches will change the order of the top-N recommended items by traditional recommender systems by calculating new measurements and determining a new rank. For such a re-ranking algorithm, R will be used as input. For each item $i \in R$, a function f will be applied to calculate the new rank. The last step is to iteratively go through R and insert them, based on their new ranks, in the final recommendation list RL. See Algorithm 1. This also means that an existing conventional recommender algorithm (baseline algorithm) is needed. Chapter 4 will further describe which ones we will be using and explain why.

Algorithm 1 General re-ranking algorithm

Input R, a set of recommended items for user u
Output RL, the same set of items as R, but in a different order

1: $O \leftarrow []$
2: **while** $|R| > 0$ **do**
3: $\quad i \leftarrow \arg\max_{i \in R} f(i, RL)$
4: $\quad$ delete i from R
5: $\quad$ append i to RL
6: **return** O

Function $f(i, RL)$ here represents a general scoring function for item i from R that will be put into O. The re-ranking algorithm will iteratively go over all items and maximize or minimize function f. Adomavicius and Kwon [75] for example compared five different re-ranking methods. In their Item Popularity re-ranking approach, rank $f(i, RL)$ would be based on the amount of ratings for item i relative to the amount of ratings for other items in list RL. Or for their Item Absolute Likeability re-ranking approach, rank $f(i, RL)$ is based on how many users liked item i relative to how many users liked the other items in RL. Chidlovskii et al. [76] re-ranked search results based on user feedback. The importance of keywords in the query would be re-calculated based on user and community profiling.

In this chapter we have elaborated on existing and related research on (contextual) recommender systems, audio features usage and re-ranking. This gives us enough knowledge to take into account and use during this research. In the next chapter, we analyze the relation between audio features and contextual conditions. This relation is crucial for the re-ranking, since these audio features are used to represent user preference for contextual conditions and based on this a new suitability score for each song in the initial recommendation is calculated.

[3] https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

3

Context-Audio Feature Correlation Analysis

In this chapter, we analyze the distinctiveness of audio features in different contextual dimensions and conditions. The goal is to answer the first sub-research question, which has been defined in Chapter 1 as:

sub-RQ1: How are contextual conditions of different contextual dimensions related to audio features?

The answer to this question indicates whether it is useful to carry-on with the current research question. If there is no correlation, either positive or negative, between audio features and context, re-ranking based on contextual audio features would be useless. In this analysis, audio features related to specific contexts are gathered through the Spotify API [1]. The higher the degree of correlation, the greater the confidence in potential effectiveness of audio features as re-ranking input later on.

Section 3.1 elaborates on the used audio features and where they are extracted from. After that, an elaboration of the chosen contextual dimensions is given in Section 3.2, followed by the visualizations of the results in Section 3.3. Section 3.4 concludes this chapter by summarizing the findings and answering the first sub-research question.

3.1. Audio Features from Spotify

Through the Spotify API, developers are able to retrieve different audio features for any given tracks that is available on Spotify. Spotify is a music streaming service created in 2006. Today the day, Spotify supply users with a low-latency library of over 50 million unique songs. Spotify has over 200 million monthly active listeners and more than 3 billion playlists in their database [77].

The audio features endpoint from the Spotify API provides low-level audio analysis. Some features, like 'danceability' are proprietary. In March 2014, Spotify acquired 'The Echo Nest' with intents to further extend their music discovery expertise. The technical underlying for the discovery engine is sustained by The Echo Nest [78].

All audio features are precise for the input track. Most audio features are represented by a value between the 0 and 1, often representing the confidence of that feature. Only *loudness* and *tempo* have different ranges of values. These two are normalized so their values are also in between 0 and 1. This way, they can be consistently visualized together with the other audio features. A brief description for each of the audio features that are used in this analysis is given below. The audio features *key* and *mode* are left out, since they are not continuous and are therefore not descriptive enough to distinguish different contexts.

Acousticness

Measures whether a track is acoustic or not. An acoustic song is a song where solely or mainly instruments are used that through acoustic means. This is the opposite of songs where electric or electronic means are used to produce them. A value of 0.0 indicates that a song is is probably not acoustic, while 1.0 gives high confidence of a song being acoustic. The overall distribution, measured over all songs in the Spotify database, is skewed to 0.0.

[1] https://developer.spotify.com/documentation/web-api/

DANCEABILITY
Describes how suitable a track is to dance to. It is a proprietary combination created by Spotify from other musical elements, e.g. overall regularity, beat strength, *tempo*, rhythm stability etc. A value of 0.0 indicates that a song is not suitable to dance to, while 1.0 indicates a very danceable song.

ENERGY
Represents the measurement of activity and intensity of a song. Energetic tracks tend to feel faster, louder and more noisy. For example, death metal would be classified as a genre with many high *energy* song, while classical piano music tend to be of lower *energy*. Also this feature is calculated by Spotify itself. It uses a combination of dynamic range, timbre, general entropy, onset rate and perceived loudness. The closer to 1.0, the more energetic a song is.

INSTRUMENTALNESS
Predicts whether a song contains vocals or not. Only spoken words and rap are in this case counted as vocals. So sounds made by mouth, like "ooh" or "aah", are not treated as vocals. The closer a value is to 1.0, the higher the confidence for a track containing no vocals. Generally, all songs with a value above 0.5 should be instrumental. The overall distribution, measured over all songs in the Spotify database, is highly skewed to 0.0.

LIVENESS
Detects whether a song has been recorded in a setting where there is an audience present. The closer the value is to 1.0, the higher the confidence of a song being recorded with audience. Everything higher than 0.8 indicates a strong likelihood of a track being live. Also here the overall distribution has a skew to 0.0.

LOUDNESS
Measures the overall *loudness* of a track in decibels (dB). These values are average over the full duration of a track, so it is the relative *loudness*. It indicates the physical strength of a song (amplitude). Values are between -60 and 0 dB, with a strong overall skew to 0.0. Within our own analyses we normalized these values in such a way that they all fall between 0.0 and 1.0. Where 0.0 stands for -60 dB and 1.0 stands for 0 dB.

SPEECHINESS
Detects the presence of spoken words. The closer the value is to 1.0, the more speech there is exclusively, e.g. audio books, podcasts or talk shows. Between 0.66 and 1.0 indicates tracks probably made up entirely of spoken words. Tracks between 0.33 and 0.66 have both music and speech, either layered or in different parts of the track, e.g. rap music. Values below 0.33 indicate music and other non-speech tracks. Also here the distribution, measured over all songs in the Spotify database, is skewed towards 0.0.

VALENCE
Measures the positivity conveyed by a song. Values are between 0.0 and 1.0 and the higher the *valence* value, the more positive a song is, e.g. happy, euphoric, cheerful etc. Songs closer to the 0.0 are more negative, e.g. sad, angry, depressing. *Valence* shows an equal distribution, measured over all songs in the Spotify database, generally.

TEMPO
Represents the overall beats per minute (BPM) for a song. It is the speed or pace of a song and is directly calculated from the average beat duration. It has a normal distribution, measured over all songs in the Spotify database, with an average around 125-130 BPM.

3.2. CONTEXTUAL DIMENSIONS AND CONDITIONS
Knowing which audio features are going to be used, the next step is to select specific contextual dimensions together with their conditions. For this initial analysis three contextual dimensions are used, namely activity, time of day and mood.

The reasons for choosing these three contextual dimensions are threefold. Firstly, as seen in Chapter 2, these dimensions have been studied and used in previous research. Researchers have shown that these dimensions affect the listening behavior and preferences of users [55–57]. When incorporating them into recommendation algorithms, often the performance would increase [64, 65] [3, 49]. Secondly, these contextual

dimensions exists of conditions that are straightforward and people tend to agree on songs that represent them. For example, the mood 'sad' or the activity 'sleeping' have public Spotify playlists with many followers. Higher follower count indicates that more people enjoy the playlist and agree with the selection of songs representing the specific playlist. This also imposes a limitation for certain contextual dimensions. In the datasets that are used later on, there is, for example, a contextual dimension road type. This dimension indicates the type of road the user is currently driving on, which Spotify do not have public playlists for. Last, but not least, these are contextual dimensions that are also available in some the datasets, that are described later on, which are used for training, validation and offline evaluation purposes. Knowing which of these contextual dimensions and conditions are most descriptive will help selecting the right ones for the re-ranking algorithm.

Each contextual dimension has multiple conditions that can be applicable to a user at a specific moment. For time of day, each user can only have one specific condition at a time. A user is either listening to music in the morning or afternoon and not in both. For the dimensions activity and mood this is not necessarily the case. A user can for example be walking and sightseeing at the same time, or sad and angry. When two contextual conditions are closely related, it is not clear which condition and how it impacts user preferences. That is why for this analysis, only distinctive and often opposite conditions are used, e.g. sad and happy for mood or running and sleeping for activity.

Two contextual dimensions exist out of 4 conditions, while the last one has only 2 conditions. For the activity dimension, running, walking, sleeping and focus are used as conditions. Focus here represents all activities where a deep focus is needed, e.g. studying and reading. Time of day has been divided into four conditions, namely morning, afternoon, evening and night. Morning represents roughly the moment between 6:00 and 11:59, afternoon 12:00-17:59, evening 18:00-23:59 and night 00:00-05:59. For the mood dimension only happy and sad are used, since they are clear moods, opposites and there are many playlists available for both.

3.3. ANALYSIS & VISUALIZATIONS

After selecting the provider, audio features and contextual dimensions, it is time to do the analysis and visualize the results. For each available contextual condition, the same approach is used in order to analyze its audio features. The first step is to look up and select publicly available playlists that are either created by Spotify or one of its users. These playlists all are related to the given condition, e.g. "Songs for sleeping" for sleeping or "Sad Beats" for sad. Since everyone can just create public playlists, only playlists with at least 1000 followers are selected. This increases the confidence that the songs in these playlists are actually representative for the specific contextual condition, since so many other users agree.

For each contextual dimension a total of at least 500 unique songs are gathered, extracted from at least 4 different playlists. Table 3.1 shows the exact number of songs for each contextual condition. After collecting these songs, all audio features are extracted, as described above, for each of the songs. These audio features are then used to perform an independent t-test, which are described in more detail in the sub-section below. The average value for each audio feature, based on these 500+ songs, are used in the visualizations.

The implementation of pulling data from Spotify and carrying out the analysis have been done using Python 3 on a Windows machine. The code is publicly available on GitHub.[2]

3.3.1. INDEPENDENT T-TESTS

T-tests are specific hypothesis tests that are designed to compare means of two groups, which can be related for given features. These tests tell us whether those differences might have happened because of chance or whether they are statistically significant.

When a t-test has been carried out in Python, it returns three results. It returns the t-score, p-value and the degrees of freedom parameter. The t-test assumes a null hypothesis where the two means are equal to each other. The test uses the sample size and variability of both samples in order to test it against the null hypothesis. If the t-score is 0, it means the null hypothesis is validated and that the two groups are identical. The larger the t-score, the more different the two groups are. It basically is the ratio of difference between the mean of the two groups and the difference that exists within those two groups. The p-value on the other hard represents the probability of the results of the sample data happening by chance. This value is always in between the 0 and 1. Where 0.05, for example, means that there was a 5% chance that the results happened by chance. The lower the p-value, the more certain one can be about the results. We apply Bonferroni correction,

[2] https://github.com/boninggong/ContextAudioFeaturesAnalysis

Contextual Factor	#Songs	#Playlists
Running	553	6
Walking	502	5
Sleeping	632	6
Focus	559	4
Morning	568	7
Afternoon	524	7
Evening	548	7
Night	545	4
Happy	511	5
Sad	516	5

Table 3.1: All contextual conditions together with how many songs and playlists were used for the analysis of their audio features.

since we apply many independent t-tests. [3] The last parameter is the degrees of freedom, which refers to the sample values and their freedom to vary. This basically is essential when assessing the validity of the result about the null hypothesis.

For each possible combination of contextual conditions within each dimension such a t-test has been carried out. The results can be found in the tables at the end of this chapter. Audio features that show a significant correlation (as measured using the Bonferroni corrected p-value threshold) have been made bold. All audio features, including *key*, have been tested. The first and most remarkable observation is that *key*, as predicted, is the worst audio feature in terms of descriptiveness. In most cases, the p-value is greater than 0.5, which means there is a significant chance that *key* is not significant enough to represent specific conditions. Below we point out the results for each contextual dimension separately.

ACTIVITY

Table 3.8 to Table 3.14 show the results for all combinations of contextual conditions within the activity dimension. An interesting observation is that relaxing and sleeping are relatively similar to each other, which also applied to running and walking. On the other side, when comparing sleeping with running or walking, the audio features are more descriptive. The same applies when comparing relaxing with walking or running. Most audio features of these conditions have a p-value that is lower than 0.0001. This means the chance is smaller than 0.01% that these significant differences happened by chance. When looking at the more comparable pairs, like relaxing-sleeping and running-walking, we see that different audio features are descriptive. For relaxing-sleeping for example, *liveness* is not a good audio feature to keep them apart. The same applies for audio features *valence* and *tempo* for the running-walking pair.

TIME OF DAY

Table 3.2 to Table 3.7 show the results for all combinations of contextual conditions within the time of day dimension. The first observation is that audio features for these conditions are relatively less descriptive than they are for the conditions within activity. When looking at the comparison between morning and night for example, once can see that for the audio features *danceability*, *liveness*, *speechiness* and *tempo* the p-value is higher than 0.05. For *loudness* it is close to 0.03. This means the difference of these audio features in context of morning as condition versus the night condition might be due to chance. Surprisingly the comparison between morning and afternoon shows that audio features are more descriptive there. The highest p-value there, after *key*, is 0.0187 for *liveness*. Another observation is that different comparisons between two contextual conditions leads to different significance levels for the audio features. An example would be the audio feature *danceability*, which is a very descriptive when comparing afternoon to evening, afternoon to night or morning to afternoon, but not at all when trying to distinguish morning from night. This gives more uncertainty when using these audio features, since there might be noise depending on which condition is being modelled.

[3] There are a total of 117 tests, so we divide the usual p-value significance threshold of 0.05 by 130, resulting in a p-value threshold of 0.000385.

MOOD

Table 3.14 shows the results of the t-test applied to the contextual conditions of happy and sad within the mood dimension. When distinguishing happy from sad, six audio features are useful, namely *acousticness*, *danceability*, *energy*, *liveness*, *loudness* and *speechiness*. It can be seen that *valence*, *tempo*, *instrumentalness* and *key* are not good audio features to use when trying to distinguish these two conditions.

3.3.2. VISUALIZATIONS

For each contextual dimension there are two separate plots, one radio plot and one line plot. Each of those two plots shows for each contextual condition the average audio features. So every condition has nine corresponding audio features in each plot. They have been put in the appendix, because of their size. If the plots would be made smaller, the distinctions between average audio features would not be clear enough anymore. We proceed by giving a description of each plot.

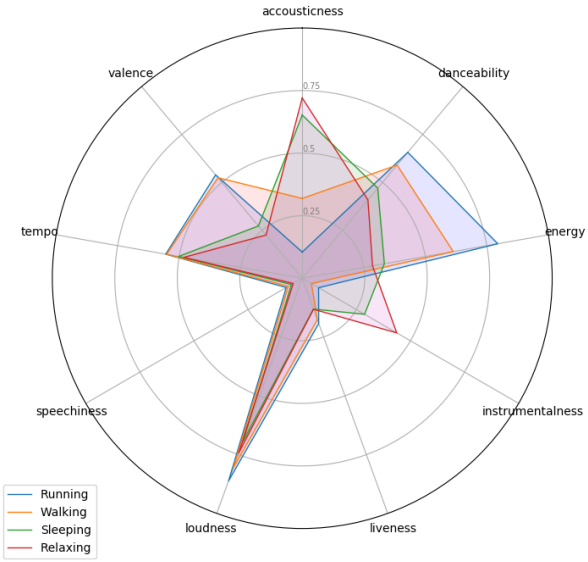


Figure 3.1: Radar plot of the average audio features for each of the four possible conditions within the **activity dimension**.

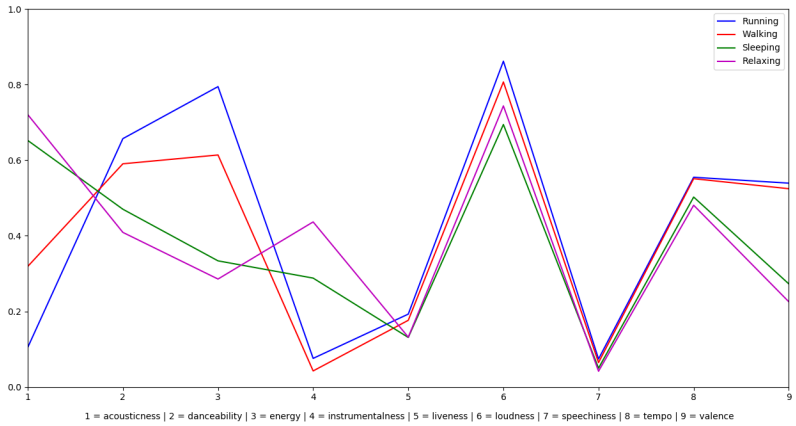


Figure 3.2: Line plot of the average audio features for each of the four possible conditions within the **activity dimension**.

ACTIVITY

The first two figures 3.1 and 3.2 are two different plots that visualize the average audio feature values for the activity dimension with running, walking, sleeping and focus as possible conditions. A few things stand out in these plots. Songs associated with running have higher *danceability* and *energy* levels than songs associated with walking. However, the audio feature *tempo* is about the same for both. Next to this, focus songs have significant higher *instrumentalness* values compared to the other three activities. A hypothesis here can be that high *instrumentalness* is less distracting for the user, so it is more suitable for activities that require focus. Focus songs also tend to have lower *energy* and *danceability* values. Songs suitable for focus and sleeping have higher *acousticness* than songs for walking or running. Furthermore, the audio feature *speechiness* is not a good indicator to differentiate between the conditions within activity.

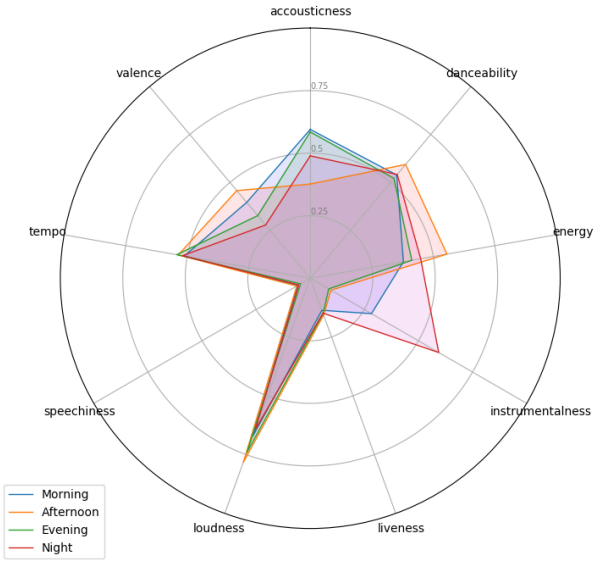


Figure 3.3: Radar plot of the average audio features for each of the four possible conditions within the **time of day dimension**.

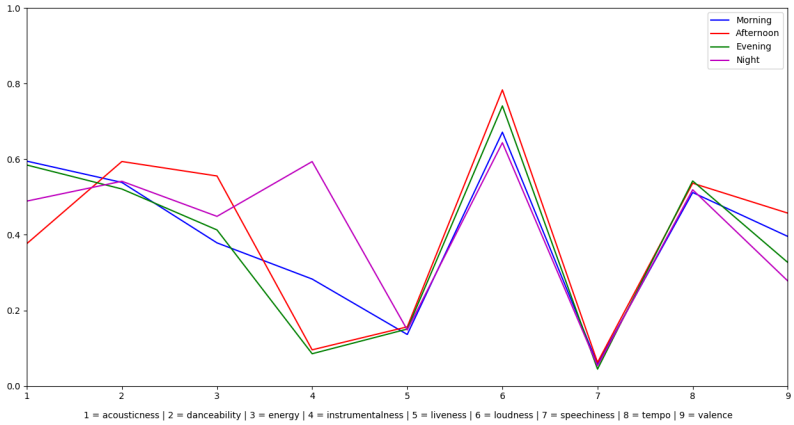


Figure 3.4: Line plot of the average audio features for each of the four possible conditions within the **time of day dimension**.

TIME OF DAY

The next plots in figures 3.3 and 3.4 are two different plots that visualize the average audio feature values for the time of day dimension with morning, afternoon, evening and night as possible conditions. The average audio feature value differences are smaller here compared to the results of the activity dimension, but are still

clearly present. It can be seen that songs related to night playlists tend to be more instrumental than during the other times. Afternoon playlists tend to be more energetic, danceable and having a positive vibe. It can also be seen that *speechiness*, *tempo* and *loudness* are not very descriptive for the time of day. So even though the difference between average audio features is smaller, time of day is a contextual dimension that is still able to provide useful information.

MOOD

Last, but not least, the two figures 3.5 and 3.6 are two different plots that visualize the average audio feature values for the happy and sad mood conditions. We can see that sad songs have higher *acousticness*, whereas happy playlists have higher values for *valence*, *danceability* and *energy*. Furthermore, *tempo*, *speechiness*, *liveness* and *instrumentalness* are not very descriptive for these two moods. The audio features of the happy and sad conditions show a strong difference in general, which means they are suitable contextual conditions to consider later on in the research.

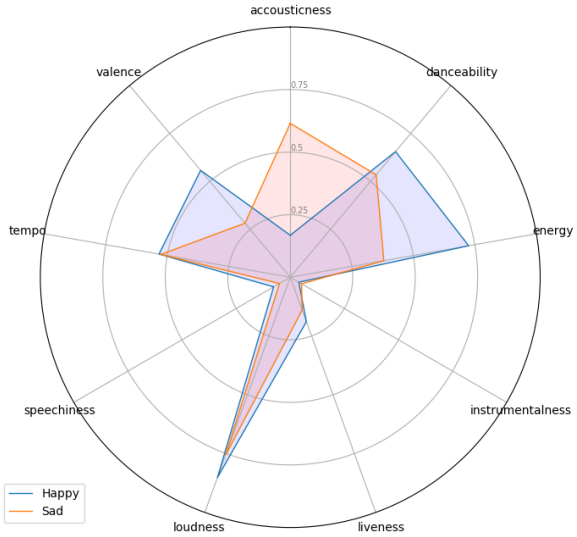


Figure 3.5: Radar plot of the average audio features for each of the two possible conditions within the **mood dimension**.

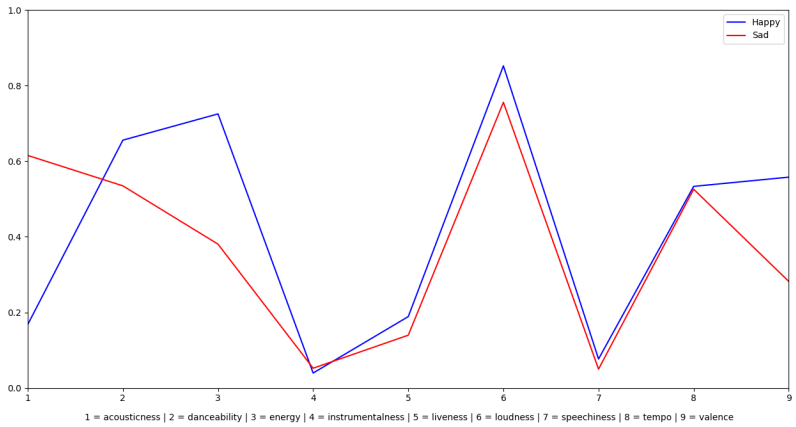


Figure 3.6: Line plot of the average audio features for each of the two possible conditions within the **mood dimension**.

3.4. CONCLUSION

The goal of this chapter was to answer the first sub-research question, which is defined as:

sub-RQ1: How are contextual conditions of different contextual dimensions related to audio features?

As can be seen from the analysis in this chapter, we can conclude that certain audio features are strong descriptive factors to distinguish between different contextual conditions. The degree of correlation and thus descriptiveness, is dimension and even condition dependent. The audio feature *valence*, as example of contextual dimension dependence, is significantly different enough to distinguish between all specific conditions within the time of day dimension, whereas it works less well within the sad and happy condition of the mood dimension. An example of condition dependence would be the audio feature *tempo* within the activity dimension. It is useful to distinguish between all possible combinations of conditions within the activity dimension, except when trying to distinguish the running and walking conditions.

If a t-value is positive, it means that the audio feature of the first contextual condition of the pair is greater than the same audio feature for the second mentioned condition. Also the opposite is true when the t-value is negative. A higher t-value means indicates that the audio feature is more suitable for describing a specific condition. So some audio features are better predictors than others. *Instrumentalness* works relatively well across all pairs of conditions within the time of day dimension, except when comparing afternoon with evening. For activity it can be seen that *energy* is the best audio features to describe each specific condition.

Since (some) audio features are significantly correlated to certain contextual conditions, it means that it makes sense to continue with our research by finding answer to the second and third sub-research questions in order to answer the main research question.

Table 3.2: T-test result for all audio features when comparing afternoon-evening songs.

afternoon-evening	t	p
acousticness	**-9.78**	**0.0000**
danceability	**8.63**	**0.0000**
energy	**10.87**	**0.0000**
instrumentalness	-0.77	0.4411
key	0.18	0.8560
liveness	0.51	0.6111
loudness	**8.46**	**0.0000**
speechiness	**6.99**	**0.0000**
tempo	0.21	0.8356
valence	**10.06**	**0.0000**

Table 3.3: T-test result for all audio features when comparing morning-afternoon songs.

morning-afternoon	t	p
acousticness	**9.59**	**0.0000**
danceability	**-6.33**	**0.0000**
energy	**-11.37**	**0.0000**
instrumentalness	**9.31**	**0.0000**
key	0.24	0.8122
liveness	-2.36	0.0187
loudness	**-11.04**	**0.0000**
speechiness	-2.54	0.0112
tempo	-2.91	0.0036
valence	**-5.15**	**0.0000**

Table 3.4: T-test result for all audio features when comparing afternoon-night songs.

afternoon-night	t	p
acousticness	-3.47	0.0005
danceability	**6.05**	**0.0000**
energy	**5.24**	**0.0000**
instrumentalness	**-24.94**	**0.0000**
key	0.05	0.9624
liveness	0.53	0.5986
loudness	**13.04**	**0.0000**
speechiness	**4.00**	**0.0001**
tempo	1.96	0.0507
valence	**14.12**	**0.0000**

Table 3.5: T-test result for all audio features when comparing morning-evening songs.

morning-evening	t	p
acousticness	0.14	0.8910
danceability	1.74	0.0824
energy	-1.36	0.1726
instrumentalness	**9.30**	**0.0000**
key	0.44	0.6601
liveness	-1.90	0.0573
loudness	**-6.34**	**0.0000**
speechiness	**5.71**	**0.0000**
tempo	-2.81	0.0050
valence	**5.37**	**0.0000**

Table 3.6: T-test result for all audio features when comparing morning-night songs.

morning-night	t	p
acousticness	**4.98**	**0.0000**
danceability	-0.34	0.7336
energy	**-4.08**	**0.0000**
instrumentalness	**-13.01**	**0.0000**
key	0.30	0.7629
liveness	-1.74	0.0820
loudness	2.19	0.0286
speechiness	1.69	0.0915
tempo	-1.16	0.2462
valence	**9.47**	**0.0000**

Table 3.7: T-test result for all audio features when comparing night-evening songs.

night-evening	t	p
acousticness	**-4.95**	**0.0000**
danceability	2.11	0.0347
energy	3.10	0.0020
instrumentalness	**25.73**	**0.0000**
key	0.14	0.8862
liveness	-0.05	0.9626
loudness	**-8.71**	**0.0000**
speechiness	**4.80**	**0.0000**
tempo	-1.79	0.0732
valence	**-3.78**	**0.0002**

Table 3.8: T-test result for all audio features when comparing relaxing-sleeping songs.

relaxing-sleeping	t	p
acousticness	**3.99**	**0.0001**
danceability	**-5.64**	**0.0000**
energy	**-4.04**	**0.0001**
instrumentalness	**5.70**	**0.0000**
key	-0.54	0.5907
liveness	-0.19	0.8467
loudness	**-7.14**	**0.0000**
speechiness	**-3.97**	**0.0001**
tempo	-2.73	0.0065
valence	**-4.50**	**0.0000**

Table 3.9: T-test result for all audio features when comparing relaxing-walking songs.

relaxing-walking	t	p
acousticness	**21.41**	**0.0000**
danceability	**-17.32**	**0.0000**
energy	**-23.79**	**0.0000**
instrumentalness	**19.07**	**0.0000**
key	-2.67	0.0078
liveness	**-6.48**	**0.0000**
loudness	**-20.29**	**0.0000**
speechiness	**-8.00**	**0.0000**
tempo	**-8.46**	**0.0000**
valence	**-21.43**	**0.0000**

Table 3.10: T-test result for all audio features when comparing running-relaxing songs.

running-relaxing	t	p
acousticness	**-45.53**	**0.0000**
danceability	**24.24**	**0.0000**
energy	**49.48**	**0.0000**
instrumentalness	**-17.03**	**0.0000**
key	1.96	0.0501
liveness	**9.13**	**0.0000**
loudness	**28.69**	**0.0000**
speechiness	**12.03**	**0.0000**
tempo	**10.40**	**0.0000**
valence	**23.95**	**0.0000**

Table 3.11: T-test result for all audio features when comparing running-sleeping songs.

running-sleeping	t	p
acousticness	**-35.88**	**0.0000**
danceability	**20.20**	**0.0000**
energy	**41.98**	**0.0000**
instrumentalness	**-10.43**	**0.0000**
key	1.55	0.1218
liveness	**9.65**	**0.0000**
loudness	**20.21**	**0.0000**
speechiness	**8.46**	**0.0000**
tempo	**7.10**	**0.0000**
valence	**22.32**	**0.0000**

Table 3.12: T-test result for all audio features when comparing running-walking songs.

running-walking	t	p
acousticness	**-13.67**	**0.0000**
danceability	**7.84**	**0.0000**
energy	**15.57**	**0.0000**
instrumentalness	3.09	0.0021
key	-0.67	0.5018
liveness	2.63	0.0087
loudness	**12.52**	**0.0000**
speechiness	2.82	0.0050
tempo	0.45	0.6515
valence	1.14	0.2531

Table 3.13: T-test result for all audio features when comparing walking-sleeping songs.

walking-sleeping	t	p
acousticness	**-16.96**	**0.0000**
danceability	**12.73**	**0.0000**
energy	**19.89**	**0.0000**
instrumentalness	**-12.37**	**0.0000**
key	2.29	0.0224
liveness	6.79	0.0087
loudness	**12.34**	**0.0000**
speechiness	**4.81**	**0.0000**
tempo	**5.77**	**0.0000**
valence	**19.73**	**0.0000**

Table 3.14: T-test result for all audio features when comparing happy-sad songs.

happy-sad	t	p
acousticness	**-25.56**	**0.0000**
danceability	**14.76**	**0.0000**
energy	**26.92**	**0.0000**
instrumentalness	-1.28	0.2026
key	0.25	0.7999
liveness	**6.04**	**0.0000**
loudness	**18.52**	**0.0000**
speechiness	**7.14**	**0.0000**
tempo	0.71	0.4784
valence	22.37	0.2531

4

METHODOLOGY

Chapter 3 showed us that there exists a significant correlation between audio features and contextual conditions. This means we can carry on addressing sub-research questions 2 and 3. This chapter describes our research methodology and how it is designed in order to answer our questions. Section 4.1 starts by giving an overview of existing datasets together with a detailed description of the evaluation and the selection process. Next, Section 4.2 describes the selection process of the initial recommender algorithms to be used in our experiment. The idea of the experiment is briefly described in Section 4.3. Last, but not least, we explain which evaluation metrics there exists in Section 4.4.

4.1. DATASETS EVALUATION & SELECTION

In order to answer the research questions, a solid recommender system and re-ranking model need to be build. An important aspect to consider is the quality of the data that goes in the system. It does not matter how superb an algorithm or system might be, if the data does not meet some minimal quality requirements, the output is still meaningless.

4.1.1. CRITERIA FOR THE DATASETS

In our research music recommendations in specific contexts are the focus, thus useful datasets would be ones containing songs, users, contexts and, not least important, interactions between users and songs. Next to these factors, it would be preferable to have datasets that are both dense and contain a proper amount of songs and users. Both are needed in order to do meaningful analysis. Suppose a dataset contains 1000 users and 1000 songs, but every user only has rated 1 unique song. This dataset would be useless, since it is too sparse to extract any useful information. Now suppose the opposite situation, having a dataset with 10 users and 50 songs, where each user has rated 10 unique songs. This would be slightly better, since it is possible to build initial user and song models. Still, the dataset itself is so small that the obtained results could have easily be influenced by some random factors.

Regarding contexts, it would be nice to have multiple contextual dimensions with even more conditions. The re-ranking algorithm, which will be described in Chapter 5, is a general algorithm that work with any type of contextual input. So having different contexts, e.g. time, activity, weather etc., would be useful for three reasons. First, it would allow for comparison across dimensions, for relevance as well as impact. Second, it would show that the re-ranking algorithm is able to work with any context. Last, but not least, it would increase the robustness and certainty of the results, since it is based on different contexts from different datasets.

The interactions of users with specific songs can be divided into two types, explicit and implicit feedback. The next section describes each more in depth. Existing baseline recommender systems can deal with both types of data. However, explicit feedback data is preferred, since it provides more information and certainty regarding how much a user likes a song.

4.1.2. EXPLICIT VS IMPLICIT FEEDBACK DATA

The main distinction between explicit and implicit feedback is what the data represents. Explicit feedback exists of explicitly indicated preferences by users regarding their interaction with/interests in certain items.

An example would be a user who gives a movie a rating between 1 and 10 on IMDB. In our context of music, an example would be a user who rates a song between 1 and 5 after listening to it. Implicit feedback, on the other hand, only indirectly reflects the preference of a user through observations of the behavior of that user [34]. The domains from which these reflections can be gathered are diverse, e.g. viewing history, click history, duration of interaction etc. An example would be a user who keeps buying books from the same author on Amazon, without giving the books an explicit rating. An example in the context of music would be a user who tweets about a song he or she is listening to at this moment. Table 4.1 gives a compact overview of the characteristics of both.

	Explicit	**Implicit**
Accuracy	High	Low
Availability	Scarce	Abundant
Context-sensitive	Yes	Yes
User Expression	Positive, neutral and negative	Positive only
Measurement reference	Absolute	Relative
Numerical interpretation	User preference	User preference confidence
Evaluation metrics	Prediction accuracy, decision support and ranking based	Only decision support and ranking based

Table 4.1: Characteristics of both explicit as well as implicit feedback dataset.

The greater part of previous research are surrounded around processing explicit feedback. This is probably because explicit feedback is less noisy, thus more accurate, and more practical to use [34]. Implicit feedback is based on interpretations of tracked user behavior. These interpretations can, in most cases, only be guessed. If a user listens to a full song for example and tweets about it, we interpret this as positive feedback. The user, however, could have tweeted about it because he or she found the lyrics controversial or have some other negative remarks about the song. The same applies to the example where user purchases are tracked, what if the purchase was a present or if the user did not like the item after receiving it?

This also brings us to the different degrees of expressivity of the types of feedback. Whereas in explicit feedback systems, a rating of 1 out of 10 indicates a negative user experience, in an implicit feedback system we only identify positive user experiences. So explicit datasets have the benefit of containing clear positive and negative user preferences. On the other hand, implicit data can be measured relatively from little positive, where there is only 1 historic interaction, to positive extremity, where a user, for example, listened to a song 100 times before. Explicit feedback is in this case absolute, since an absolute scale is used, independent whether a user listened to a song 1 time compared to 100 times. It also means that the numerical values for both types of feedback have to be interpreted differently. Explicit feedback indicates to what extend a user likes or dislikes an item, whereas the number in implicit feedback indicates how confident we are that (and to some some extend also how much) a user actually liked an item.

Even though explicit feedback is more accurate, it is not more readily available compared to implicit feedback [79]. To collect explicit data, users need to actively interact with a system, indicating their explicit preference with each item that they have interacted with. For implicit data, the system needs to track user metrics, from which preference is deducted. Gathering implicit feedback, thus, forms less of a barrier for users.

The music datasets that we have to our availability contain a mix of explicit and implicit feedback. Despite all the above-mentioned differences, both are sensitive to the context of a user [79]. Since our research is focused on how contextual information, together with audio features, can be included in a re-ranking algorithm to impact recommendations, it means that both types of datasets are suitable for our experiment.

4.1.3. OVERVIEW AND ANALYSIS OF POTENTIAL DATASETS

Ideally, we would have multiple datasets meeting all of our requirements described previously. However, realistically, it is already very hard to find one such high quality dataset fulfilling all requirements. After thoroughly searching the internet and reaching out to relevant researchers in comparable fields, five (semi-)public datasets were found that all contain users, songs, interactions and contextual information. Table 4.2 gives an overview of these datasets together with the source and some properties. Below we give an in-depth anal-

ysis and evaluation of these datasets. We also elaborate on the final selection of datasets to be used in the re-ranking experiment.

Dataset	#Ratings	#Users	#Songs	#Contextual factors	Source
#NowPlaying-RS [80]	11,639,541	4,150,615	346,273	7	Twitter + Spotify API
MMTD [2]	1,086,808	215,375	133,968	11	Twitter
InCarMusic [63]	4012	66	139	8	Own experiment
Yahoo! Music [3]	262,810,175	1,000,900	624,961	2	Yahoo! Music Radio Service
MMG [81]	8370	70	~4000	10	Own experiment

Table 4.2: Overview and comparison of existing music recommendation datasets that contain contextual factors

#NOWPLAYING-RS

This dataset is created by Poddar et al. [80]. This dataset is an enriched version of a subset of the underlying #NowPlaying dataset, which was originally compiled by Zangerle et al. [82]. The listening events (LEs), which are a combination of user-song pairs, in this dataset were crawled from Twitter, each LE being extracted from individual unique tweets. Listening events were identified using the NowPlaying hashtag that users used to tweet about songs. The original #NowPlaying dataset only contained basic LE information, like timestamp, track, artist.

Poddar et al. used the subset of all LEs of the year 2014. They used the Twitter API[1] to add additional information of the user of the tweet, like tweet language, geolocation and time zone (when they are available). Also, the hashtags within each tweet got extracted, these could serve as contextual indicators for example. They then proceeded to extract affective contextual information using these hashtags. This was done through application of an unsupervised sentiment dictionaries. Multiple dictionaries were used, like AFINN [83], Opinion Lexicon [84] among others. Next to this, they also extracted the audio features for each unique track through the Spotify API. The track API was first used to match the right track and artist, from which they took the Spotify-ID to then extract the audio features. The audio features instrumentalness, liveness, speechiness, danceability, valence, loudness, tempo, acousticness, energy, mode and key were included.

Initial analysis of the hashtags within the dataset showed that most tags do not provide any additional information regarding the user's context. The top 10 most frequent occurring tags are depicted in Table 4.3. We can see that none of them are context related. When looking at contextual hashtags, the occurrences are very volatile, ranging from only 10 occurrences to 1852. They are depicted in Table 4.4.

Tag	Occurrence
nowplaying	11,089,013
listenlive	2,794,285
music	166,242
np	114,115
radio	111,469
chicagomusic	87,660
tunein	87,647
hitmusic	68,374
anghami	63,483
kiss92	56,546

Table 4.3: Top 10 most frequent occurring tags within the #NowPlaying-RS dataset

Context related tag	Occurrence
morning	1750
afternoon	5
evening	2
night	1852
happy	147
sad	27
workout	40
running	10
working	35
gaming	1657

Table 4.4: Random 10 contextual tags and their frequency within the #NowPlaying-RS dataset

Next to the scarcity of contextual information within this dataset, we're also missing any explicit user feedback. The dataset, gathered from tweets, only indicates that a user is listening to a specific track at the moment of tweeting about it. It does not say whether a user likes it or not, let alone to which extent. Metrics like listening duration are also not present, which limits the type of recommendation algorithms that can be used as baseline. One way to use this implicit feedback dataset is with the assumption that listening events indicate positive user feedback.

[1] https://developer.twitter.com/en/docs/tweets/search/api-reference

MILLION MUSICAL TWEETS DATASET (MMTD)

MMTD has been put together by Hauger et al. [2]. They crawled the Twitter Streaming API[2] from September 2011 to April 2013. This API radonmly gives 1% of all tweets. Within these tweets, only tweets containing music-related hashtags were kept. Hashtags like #nowplaying, #itunes, #thisismyjam etc. were used. This means that also this dataset, like the #NowPlaying-RS dataset, is based on implicit feedback. Tracks, timestamp and artists were extracted from the content of the tweet as well. They also extracted the genre by querying Last.fm using the artist and song information. Next to these, Mapquest[3] was used to map the geo coordinates of the tweets to cities and countries and GeoNames[4] to get the time zones based on these coordinates. Figure 4.1 shows the statistics of the countries created by Hauger et al.

Since this dataset has been gathered in a comparable way to #NowPlaying-RS, it is very similar. Both contain enough listening events, unique users and tracks, but MMTD contains less contextual information. It does not have the audio features and affective representation that the #NowPlaying-RS dataset contains. This is the reason we decided not to look further into and analyze this dataset.

rank	country code	number of users	rank	country code	number of tweets
1	US	70,204	1	US	227,432
2	ID	30,605	2	DE	153,163
3	BR	24,985	3	BR	145,049
4	MY	14,771	4	GB	130,951
5	FR	13,890	5	ID	94,245
6	GB	9,006	6	FR	65,525
7	RU	5,234	7	MY	50,648
8	NL	5,223	8	CA	27,370
9	MX	4,538	9	RU	23,542
10	TR	2,878	10	MX	18,717
11	ES	2,847	11	NL	18,320
12	SG	2,422	12	TR	14,479
13	PH	2,385	13	ES	11,811
14	CA	2,344	14	SG	7,637
15	IT	1,521	15	IT	6,412
16	JP	1,501	16	AR	6,319
17	ZA	1,297	17	PH	5,723
18	DE	1,133	18	JP	5,529
19	UA	1,062	19	UA	4,940
20	AR	874	20	ZA	3,733

Figure 4.1: Top 20 countries measured by user frequency on the left and amount of tweets on the right [2].

INCARMUSIC

Baltrunas et al. [63] created a context-aware mobile music player as well, but with a more specific focus for users that are in a car. They have created InCarMusic, an Android application offering song recommendations based on current contextual (driving) conditions. Since it is hard to collect user ratings while driving, they created a separate Web application that users had to use beforehand to collect ratings during certain driving situations. During this process, users had to imagine being in a certain contextual situation and give ratings based on this situation. The interesting part of this dataset is the type of contextual dimensions and conditions. Most of them are driving related, e.g. driving style, road, landscape, traffic etc. This dataset has both benefits as well as drawbacks. An advantage is that these conditions might be able to show new underlying relations between audio features and user preferences. One drawback is that the ratings are obtained in a simulation environment, which might be slightly off from real world scenarios. Another drawback is that since most contextual dimensions and conditions are very different from the contextual factors of the other datasets, the results can not be directly compared to each other. Also, this dataset is relatively sparse, so it might be hard to extract strong underlying relationships.

After obtaining this dataset, we proceeded to do some initial analysis. When only taking ratings and users that actually had at least one contextual factor, we were left with 42 users, 139 songs and 1742 ratings in

[2] https://dev.twitter.com/docs/streaming-apis
[3] http://www.mapquest.com
[4] http://geonames.org

total. There were a total of 8 context dimensions, namely *weather*, *trafficconditions*, *sleepiness*, *roadtype*, *naturalphenomena*, *mood*, *landscape* and *drivingstyle*. These 8 dimensions have a total of 26 conditions, with each dimension having between 2 and 4 unique conditions. Table 3.1 shows all dimensions with their respective conditions. The average rating is 2.6, with a standard deviation of 1.4. The rating distribution is as follows: 487 x 1.0, 372 x 2.0, 360 x 3.0, 283 x 4.0, 196 x 5.0.

Contextual Dimension	Contextual Conditions
Weather	sunny, snowing, rainy, cloudy
Traffic condition	traffic jam, lots of cars, free road
Sleepiness	sleepy, awake
Roadtype	city, serpentine, highway
Natural phenomena	night, morning, day time, afternoon
Mood	sad, lazy, active, happy
Landscape	urban, mountains, country side, coast line
Driving style	relaxed driving, sport driving

Table 4.5: All the contextual dimensions with their conditions of the InCarMusic dataset.

This dataset has a nice wide variety of contextual dimensions and conditions so we decided to further analyze how suitable it would be for testing the personalized model later in the experiment. As described in Section 4.3, in order to accurately answer the third sub-research question a dataset with enough individual user ratings is needed. This means that in a good dataset each user should have at least a few positive ratings for different conditions in multiple dimensions. For this evaluation we calculated the average amount of positive ratings for each contextual condition per user. The results are shown in Table 4.6. The average amount is of ratings per user per condition is in 10 out of 26 cases even under 1. Some dimensions, like drivingstyle and landscape, have conditions with a value higher than 1, but these are still under 2, which means that probably a few users have multiple ratings in these conditions but many have only 1 or none at all. So it would be hard to test the personalized model accurately with this dataset.

Contextual Condition	**Avg #ratings/user**	**Contextual Condition**	**Avg #ratings/user**
sunny	1.14	traffic jam	0.74
snowing	0.95	lots of cars	1.38
cloudy	1.14	free road	1.31
rainy	0.95	sleepy	0.64
night	1.24	awake	0.94
morning	1.10	sad	0.83
afternoon	1.12	lazy	0.86
day time	0.88	active	0.93
city	1.31	happy	1.07
urban	1.10	sport driving	1.62
mountains	1.33	relaxed driving	1.62
country side	0.90	serpentine	1.31
coast line	1.33	highway	1.70

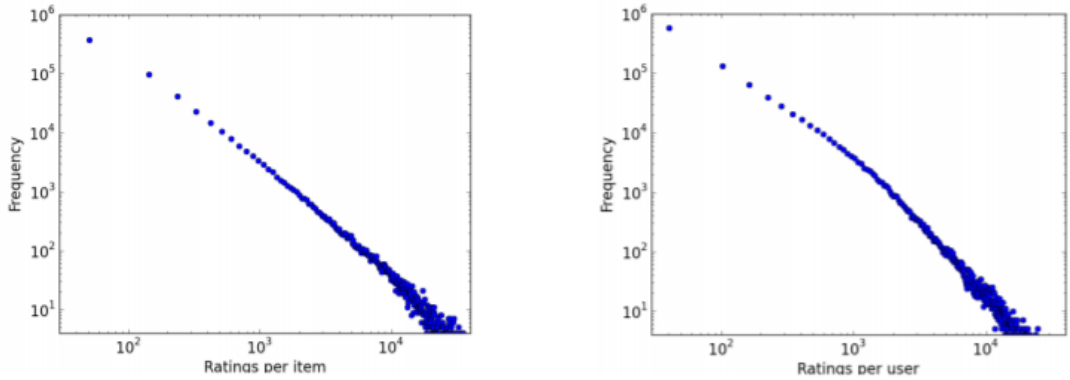
Table 4.6: The average amount of ratings per user for each contextual condition measured over the whole InCarMusic dataset

YAHOO! MUSIC

Dror et al. describes the KDD-Cup 2011 challenge as an open community challenge to predict user music preference using the Yahoo! Music dataset [3]. The competition is of less relevance to us. The dataset however, is more interesting. For the context, two datasets were created, namely Track1 and a smaller dataset Track2. They are similar except that Track2 omits times and dates. We proceed to describe Track1, other research also primarily describes.

The dataset was created from the Yahoo! Music online music radio stations by gathering user data between 1990 and 2010. It was one of the earlier music streaming services available online. The service was free with advertisements in between, users could remove them by upgrading to a premium membership. Users

were also able to give ratings to songs and artists based on a five star system. The system behind Yahoo! Music would then use this information to create more suitable recommendations. In the end, a total of more than 260 million ratings, 600 thousand songs and 1 million users were collected. They made sure that each user and song have at least 20 ratings in this set. Figure 4.2a and Figure 4.2b, obtained from the paper by Dror et al. [3], show the distribution of how many ratings users gave and how many times songs have been rated. As can be seen, they follow a power-law distribution where a few very popular songs have exponentially more ratings and a few users who use the service a lot and giving out many ratings.



(a) Song Distribution.

(b) User Distribution.

Figure 4.2: Both the distribution of users as well as songs show a power-law distribution [3]

Since users had to give rating ranging from 1 to 5, this dataset is made up of high quality explicit feedback. However, contextual factors in this dataset are more limited. As far as we know, only the timestamp is available as contextual factor. We were not able to get our hands on the full dataset. A request to the relevant Yahoo department has been made, together with a reminder, but without any success.

MOBILE MUSIC GENIUS (MMG)

Schedl et al. [81] created a context-aware mobile music player, namely Mobile Music Genius (MMG) for their research into context-aware music recommendations. It is an Android application which uses a SQLite database to store data. In the background the application unobtrusively gathers several contextual attributes of users during usage. Next to these automatically gathered factors, users also have the option to enter their activity and mood for each track that is currently being played.

Through their own experiment, they harvested contextual data regarding users context for two months. A total of 70 people signed up, foremost students from the Johannes Kepler University Linz. Out of these 70, 42 actually produced contextual data. 8000 listening events were recorded during this time. During this time, students were allowed to use their own music collection in order to minimize artificial adaptation of music listening behavior of users. Contextual factors that were gathered include date, time, weather, activity, mood and all kinds of factors related to the state of the phone, e.g. wifi, battery, bluetooth status etc. The top 10 of most frequent contextual factors, as they are classified by Schedl et al., are depicted in Table 4.7. Some contextual conditions like relaxing and traveling have a good amount of occurrences. However, they differ quite a lot, studying only has 128 for example, this would make a direct comparison unfair.

The contextual values would suite our needs. However, also this dataset does not contain any explicit feedback of users regarding the songs they listened to. They did keep track of and provide listening duration, but on closer inspection, only a fraction of the listening events actually have it. The top 10 most frequent listening durations are shown in Table 4.8. As can be seen, the occurrences decrease significantly when going down the list. Actually, close to 90% of all the data does not contain any listen duration at all. On even closer inspection, the listening events containing duration only comes from 2 specific users, users with ID 3 and 5.

4.1.4. SELECTION OF DATASETS

The previous sections presented five different datasets and elaborated on the creation of them and which properties they have.

For our research we strongly focus on building models that use contextual conditions to re-ranking recommendations. So in that sense, high quality and diverse contextual information is important when selecting

Context	Occurrence
unspecified	3842
relaxing	950
traveling	916
working	813
sports	608
shopping	209
eating	147
studying	128
partying	111
falling asleep	58

Table 4.7: Top 10 most frequent occurring contextual factors within the MMG dataset.

Listening duration	Occurrence
unspecified	7372
100	495
0	80
1	54
2	34
3	25
4	18
9	16
7	13
5	12

Table 4.8: Top 10 most frequent occurring listening durations within the MMG dataset.

datasets. Preferably having a big enough amount of listening events, users and songs that give more confidence in the underlying relations in the dataset. In the end, we selected to use the InCarMusic by Baltrunas et al. [63] and the #NowPlaying-RS datasets by Schedl et al. [81] for our research.

When selecting the most suitable datasets, we took all the requirements described in Section 4.1 into account. One of the most preferred features is that the dataset has explicit user feedback, e.g. a rating between 1 and 5. This would allow us to run a much bigger group of recommendation algorithms, which would in turn give us more insights in the performance of our re-ranking algorithm. From the five datasets described, only the Yahoo! Music [3] and InCarMusic datasets have explicit user feedback.

The Yahoo! Music dataset is the most comprehensive dataset, but is limited to only time and day of week as contextual dimensions. This is not too big of a problem, but we were also not able to get it from Yahoo or other sources, so we decided to leave this and move on to the other accessible datasets. The InCarMusic dataset has a wide variety of contextual dimensions which can be valuable to our research. The one main drawback of this dataset is that it is sparse. Having only 42 users with contextual information, it would not be too reliable to model underlying relationships. So for the second dataset we would need one with significantly more users, songs and listening events.

We're left with the MMTD, MMG and #NowPlaying-RS datasets. The MMG dataset includes a wide variety of contextual dimensions. However, the lack of consistency of occurrences for specific contextual conditions is concerning. This would not form a stable foundation for running a re-ranking algorithm on. The amount of users, songs and listening events here are slightly better than the InCarMusic dataset, but still not good enough for our needs. So this dataset is discarded as viable option. The MMTD dataset used the same extraction and creation method as the #NowPlaying-RS dataset. Both of them scraped tweets and used hashtags to determine whether the tweets represented a listening event. The #NowPlaying-RS dataset has been enriched with sentiment values and audio features and is in itself more comprehensive compared to MMTD. MMTD has been created from tweets between 2011 and 2013, while the underlying tweets in #NowPlaying-RS were gathered in 2014. That is why we decided to include the #NowPlaying-RS dataset. Even though it is an implicit feedback dataset, with limited contextual dimensions, it can still be valuable by providing underlying relations that can be modelled in our re-ranking algorithm. The size of the dataset also makes the relationship mapping and thus the re-ranking results more reliable.

4.2. INITIAL RECOMMENDER ALGORITHM SELECTION

In order to use and compare our re-ranking algorithm, initial recommender algorithms are needed. In the literature review (see Chapter 2) an extensive list of both traditional as well as contextual recommender algorithms was given. Traditional algorithms are used to create initial recommendation lists, which are input for our re-ranking algorithm. Contextual algorithms have two purposes. The first is the same as the traditional algorithms to create input for the re-ranking algorithm. The second is that they act as baseline comparison for our re-ranking results. Since they already include context, it is interesting how our audio features based contextual re-ranking compares against contextual algorithms. One might argue why we would contextually re-rank a contextual recommender algorithm. Here we can say that our re-ranking algorithm is different since our re-ranking uses underlying audio features of contextual information.

The selection of the recommender algorithms to be used is dependent on three factors. First and most

important, there needs to be an implementation of the algorithm that can be easily run with custom data. To implement all the algorithms ourselves is not feasible and lies outside of the scope of this research. Second, the algorithm needs to be compatible with the type of feedback in the dataset. Some recommender algorithms only work with explicit feedback, running an implicit feedback dataset using it would not make any sense. Last, the selection should be based on the general performance of the algorithm. Comparing the re-ranking output to only the worst performing algorithms would not provide much information. On the other hand, only comparing to best in class algorithms can marginalize the actual performance of the re-ranking algorithm.

So in the ideal situation, we would have multiple recommender algorithms, both traditional as well as contextual. It should be easily available for usage (there is an existing implementation), be suitable for the type of dataset (explicit or implicit feedback) and have a decent performance. Chapter 6 describes which algorithms are used for the experiment in the end.

4.3. EXPERIMENT

After having selected the most suitable datasets and initial recommender algorithms, the next step is to design and implement the re-ranking algorithm. The previous chapter has shown us that there is a correlation between audio features and contextual conditions and is something to take into account when designing the re-ranking algorithm. An example would be that the re-ranking algorithm uses the degree of correlation to give weights to specific conditions and/or audio features, since not all of them are significant. Chapter 5 gives a detailed description of the re-ranking algorithm that we created to answer our research questions.

The next step after designing the algorithm is to actually implement it and evaluate its performance. There are many programming languages available out there, but are using Python 3 to implement the whole experiment pipeline, including the re-ranking algorithms and any data preprocessing if necessary. There are two main reasons for this choice. The first is that it is our most proficient language. Second, because of its growing usage and popularity, there are many ready to use libraries. The pandas[5] library, for example, is very useful for processing and analyzing data. Next to this, it is also a good language to handle big amounts of data, which is useful if one of the bigger datasets are used.

The output of the initial recommendation is used as input for the re-ranking algorithm. Depending on the libraries/framework used, the output might have different structures. Here we make sure to use open-source libraries as much as possible. This gives the flexibility to make our own changes and tweaks to the underlying code, in order to have the output in the desired structure. Having a consistent initial recommendation list makes it much easier to implement the re-ranking system.

All previously mentioned steps create and output the initial recommendation list together with its re-ranked version. The last step is to evaluate all the recommendation lists based on some kind of evaluation metrics. The next section gives a detailed description of which evaluation metrics are commonly used and how they work exactly. This evaluation we will also implement using Python 3, since the re-ranked recommendation list are given as output by the re-ranking system already.

4.4. EVALUATION

It is important to measure the quality of both the re-ranked recommendation list as well as the initial recommendation list to get an idea of how the re-ranking algorithm is performing. This can be done through several metrics that measure accuracy or relevance. Many metrics exists, each with their own benefits and drawbacks in specific situations. This section proceeds to describe some of the most common ones that are found in literature.

4.4.1. PREDICTION ACCURACY METRICS

The mean average error (MAE) is a straightforward way to measure prediction accuracy. It measures the average magnitude of errors in a given set of predictions. The formula is the average of all absolute differences between the actual value and the predicted value, where each difference has the same weight. The MAE is defined as follows:

$$MAE = \frac{1}{|R|} \sum_{r_{ui} \in R} |f(u,i) - r_{ui}| \tag{4.1}$$

[5] https://pandas.pydata.org/

R stands for the test set. Each item within the test set contains user u, item i and the actual rating r_{ui} of this user for the given item. Furthermore, $f(u,i)$ is the prediction function that outputs a predicted rating of user u for item i.

Another common metric is the root mean squared error (RMSE), it is a metric that is usually used to evaluate regression problems. It is very similar to the MAE, except that it does not weigh each difference equally. The RMSE gives relatively high weight to larger errors, which means that it is more useful when large errors are undesirable. The RMSE is defined as follows:

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{r_{ui} \in R} (f(u,i) - r_{ui})^2} \tag{4.2}$$

Here the symbols refer to the same data as with the MAE. MAE and RMSE are examples of statistical accuracy metrics. In those metrics, actual ratings are needed. However, those ratings are not in all cases available. If a ranking based recommendation algorithm is used, in the case of a dataset that does not contain any rating information for example, then the output will give arbitrary numbers and create a recommendation sorting these numbers. These numbers do not represent any rating prediction. Thus, there is no prediction to calculate the prediction accuracy.

4.4.2. DECISION SUPPORT ACCURACY METRICS

Luckily, there are decision-support accuracy metrics, which measure how well a recommender system can predict which items are relevant [6]. These metrics do not look at how accurate the prediction is, instead they only look at whether the items are relevant and how they are classified by the output. The following four classifications are used for prediction results:

	Recommended	Not recommended
Relevant	True Positive (TP)	False Negative (FN)
Not relevant	False Positive (FP)	True Negative (TN)

Table 4.9: Overview of possible results for made recommendations regarding an item for a specific user

In the ideal situation, one would have 100% TPs and TNs, without any FPs and FNs. In pratice, there are almost always FPs and FNs. The goal then is to try to minimalize FPs and FNs and maximalize TPs and TNs. Accuracy, precision and recall are such decision-support accuracy metrics that uses this classification to measure the prediction accuracy. The three of them are defined as follows:

$$Accuracy = \frac{(\#TP + \#TN)}{(\#TP + \#TN + \#FP + \#FN)} \tag{4.3}$$

Accuracy is the most general metric of the three, measuring how many right predictions are made relative to the amount of predictions.

$$Precision = \frac{\#TP}{(\#TP + \#FP)} \tag{4.4}$$

Precision measures how many right predictions are made out of all the positive predicted items.

$$Recall = \frac{\#TP}{(\#TP + \#FN)} \tag{4.5}$$

Recall measures how well the model does in terms of not leaving out items that should have been classified as relevant.

As can be seen from the given equations above, precision and recall influence each other. F-measure is a metric that summarizes the two into one equation:

$$F_\beta = (1 + \beta^2) * \frac{precision \times recall}{(\beta^2 \times precision) + recall} \tag{4.6}$$

Here, β is used to control the importance of precision compared to recall. A β of 0.5 means precision is more important, 1 means equally important and 2 means recall is more important.

Two other popular decision-support accuracy metrics are the hit-rate and hit-rank. Hit-rate evaluates how many items that are pre-defined as relevant to the user, were also found in the final recommendation

list [85]. However, the hit-rate is insensitive to ranking and permutations. Which means that a hit in the first position is treated equally well as a hit that occurs in the nth position. This is undesirable, since earlier hits of recommended items have more impact on users. Therefore, the hit-rank is created to deal with this limitation of the hit-rate [86]. It is defined as follows:

$$hit-rank = \frac{1}{m}\sum_{i=1}^{h}\frac{1}{p_i} \tag{4.7}$$

Here, m stands for the total number of users and h represents the number of hits within the final recommendation list, which occurred at positions $p_1, p_2, ..., p_n$

So the first two focus on the accuracy of the predictions made by the recommendation algorithm. The decision-support accuracy metrics following them look at relations between whether items are relevant and whether they are predicted as relevant. Those two fall short when dealing with recommendations where the rank is of importance.

4.4.3. RANKING BASED ACCURACY METRICS

This third category of measurements focus on how highly ranked relevant items relatively are. These are the so-called ranking based metrics. We describe the mean reciprocal rank (MRR), mean average precision (MAP) and normalized discounted cumulative gain (NDCG) below [87].

MEAN RECIPROCAL RANK

MMR is the simplest of the three. It measures the accuracy based on the position of the first relevant item. Given a recommendation list where the first relevant item is located at position k, then the MRR for that query is $\frac{1}{k}$. It only looks at the first found relevant item and ignores subsequent relevant items in the recommendation list. The MRR over several recommendation lists is given by taking the average of all individual MRR scores. It is formally defined as follows:

$$MRR = \frac{1}{|U|}\sum_{u\in U}\frac{1}{k_u} \tag{4.8}$$

Where k_u stands for the rank of the first relevant item in recommendation list u. U is the set that contains all recommendation lists.

MEAN AVERAGE PRECISION

Before the MAP can be defined, precision at position k (Prec@k) and average precision (AP) needs to be defined.

$$Prec@k(u) = \frac{\#\{relevant\ items\ in\ top\ k\ positions\}}{k} \tag{4.9}$$

Here u stands for a given recommendation list. So Prec@k looks at relatively how many relevant items there are in the top k items. After having this definition, the AP is defined as:

$$AP(u) = \frac{\sum_{k=1}^{m} Prec@k(u) * l_k}{\#\{relevant\ items\}} \tag{4.10}$$

Where m indicates the total amount of items, l_k is a binary value for whether the item is relevant (1) or not (0) and u represents the recommendation list. The MAP can then be defined as the mean value of all APs over all recommendation lists. Both the AP and MAP can also be calculated at position k. To do this, the AP only measures over the top k items instead of all m items and the denominator uses the amount of relevant items in the top k items instead of all relevant items.

MAP is different from MRR that it looks at the rank of all or multiple relevant items relatively to their positions instead of only the first relevant item.

NORMALIZED DISCOUNTED CUMULATIVE GAIN

Before defining NDCG, the discounted cumulative gain (DCG) needs to be defined. The online blog by Taifi [4] has a nice and elaborate illustration, which is depicted by Figure 4.3, of all the components that build up to the NDCG.

The goal of NDCG is similar to the MAP metric. Both of them give higher scores when relevant items are ranked higher in the recommendation list. The main differnce is that NDCG further tweaks this by looking

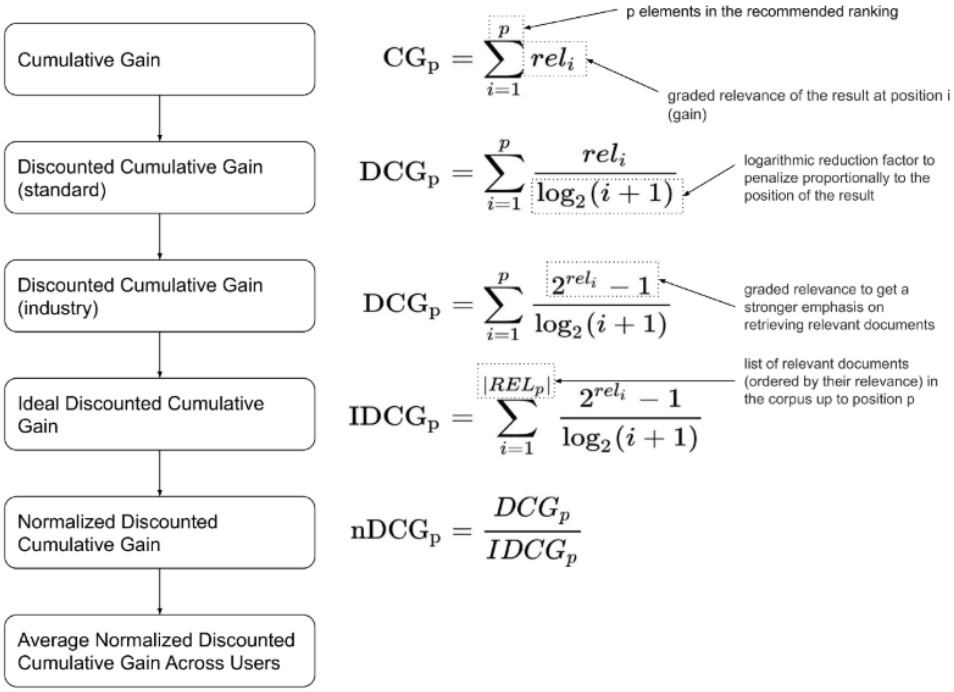


Figure 4.3: The different parts of which the NDCG is made up [4].

at how relevant items are. High relevance items should be higher ranked than low relevance, but still relevant, items, which should again be higher than not relevant items. So the NDCG is a particularly good fit for datasets that contain indications of how relevant items are.

5

PROPOSED RERANKING STRATEGY

In Chapter 1, the following two sub-research questions were defined:

sub-RQ2: How does re-ranking, based on audio feature representations of user preferences in different contextual conditions, affect music recommendation accuracy?
sub-RQ3: How do global audio feature representations of user preferences in different contextual conditions affect the re-ranking results compared to personalized audio feature representations of user preferences in the same contextual conditions?

We design a re-ranking algorithm together with global and personalized user preference models to adress the questions above. In this chapter, we start by describing the global model in Section 5.1 and the personalized model in Section 5.2. We describe the design of our re-ranking algorithm in Section 5.3, while Section 5.4 describes all adjustable parameters. Section 5.5 describes an opposite version of our re-ranking algorithm to strengthen our experiment and evaluation results. Finally, Section 5.6 occludes this chapter by stating potentially interesting ideas to test in the future.

5.1. GLOBAL MODEL

The first of the two models is called the global model. It's called global, since it utilizes all user ratings within the whole dataset to create user preference representations. These representations are created for each contextual condition using vectors of audio features.

For each unique contextual condition, the model goes through historical ratings in the training set for all users that previously have listened to a song while in this condition. In the case of explicit feedback data, a certain threshold is set to only include positively rated songs, e.g. if the ratings are on a scale from 1 to 5, only ratings higher than or equal to 3 are included. In the case of implicit datasets, all listening events are included. Here the assumption is made that whenever a user fully listened to a song, he/she likes that song. The decision to only including positively interacted songs is done to ensure that the model represents what users like in a specific condition. This also reduces noise, since the audio features of negatively interacted songs are excluded.

After extracting each positively interacted songs per condition, we gather all audio features of these songs. If the dataset does not already include the audio features or has missing values, we add them through usage of the Spotify API. In the, rather unlikely, case where the song can not be found through the Spotify API, we leave it out completely in our experiment. The audio features that are used to model the condition are the same ones as described in Section 3.1. For each audio feature, the average is calculated by taking the sum of all audio features of the selected songs and dividing it by the amount of selected songs. Thus, the global model can be represented as follows:

$$\vec{GM}_{c_k} = [a_1, a_2, \ldots, a_n] = \frac{1}{|S_{c_k}|} \cdot \sum_{s_j \in S_{c_k}} \vec{s}_j, \tag{5.1}$$

where $\vec{GM}_{c_k}$ is a vector representing the global model for contextual condition c_k, and computed by using the songs from set S_{c_k} that contains all $|S_{c_k}|$ positively interacted songs in condition c_k. $\vec{GM}_{c_k}$ is simply the

centroid of the vectors of the all songs $s \in S_{c_k}$. To make everything described above more concrete, we provide an example. Let's say the 3 audio features of *energy*, *tempo* and *acousticness* are used to model user preferences. We could have $\vec{GM}_{c_1} = morning = [0.32, 0.44, 0.82]$ and $\vec{GM}_{c_2} = evening = [0.78, 0.66, 0.21]$. What this example tells us is that, based on all users, there is a preference for more energetic songs with a higher tempo in the evening than morning, but higher accousticness songs are preferred during the morning. Since we are using the audio features described in Section 3.1, we will have user preference vectors consisting of 9 values.

5.2. PERSONALIZED MODEL

The idea of the personalized model is comparable to the global model. Also here, vectors of audio feature values are used to represent user preferences in unique contextual conditions. In the personalized model, however, instead of using positively interacted songs of all users to create the vectors, only the positively interacted songs of one unique user is used for each model. This means that an audio features based vector is created for each condition for each user.

The idea behind this model is that when specific user interactions are used, the audio features that represent a certain condition are more specific to what that individual prefers for any given condition. For example, person A might prefer to have calm piano music in the morning during breakfast, while person B prefers to have energetic dance music to get more awake in the morning. This personalized model is more granular, but is also more computationally expensive compared to the global model. It can be represented as follows:

$$\vec{PM}_{c_k,u} = [a_1, a_2, \ldots, a_n] = \frac{1}{|S_{c_k,u}|} \cdot \sum_{s_j \in S_{c_k,u}} \vec{s}_j, \tag{5.2}$$

where $\vec{PM}_{c_k,u}$ is a vector representing the personalized model for contextual condition c_k and user $u \in U$, and computed by using the songs from set $S_{c_k,u}$ that contains all $|S_{c_k,u}|$ positively interacted songs in condition c_k by user u. As an example we use the 3 audio features of *energy*, *tempo* and *acousticness* again for user 1 and user 2. User 1 prefers to listen to calm piano music during breakfast, while user 2 likes to use energetic dance music to get more awake in the morning. Possible personalized models would be $\vec{PM}_{c_1,u_1} = morning,\ user\ 1 = [0.1, 0.16, 0.98]$ and $\vec{PM}_{c_1,u_2} = morning,\ user\ 2 = [0.88, 0.76, 0.18]$.

5.3. RE-RANKING SCORING CALCULATION

Now that we have two different models that represent contextual conditions each in their own way, the next step is to use these two models to determine how songs in the initial recommendation list should be re-ranked. The idea is to calculate how similar a given song is compared to the condition in which it is recommended. This similarity is measured between the two audio feature vectors of the song and the user preference within the contextual condition, respectively. In order to compare the performance of the re-ranking algorithm to the initial recommendation algorithm, a balancing parameter, λ, is introduced in this formula as well. The resulting function takes both the newly calculated contextual similarity as well as the original recommendation score as input:

$$new\_score = \lambda * Sim(\vec{s}_j, \vec{GM}_{c_k}) + (1 - \lambda) * Rec(u, s_j, c_k)', \tag{5.3}$$

where λ is a balancing parameter, ranging from 0 to 1, that let us control the weight of the contextual similarity score relative to the weight of the initial recommendation score, $Sim(\vec{s}_j, \vec{GM}_{c_k})$ is the similarity between the audio features vector of song j, $\vec{s}_j$, and the audio feature vector, which can be either the global or personalized model representation, ($\vec{GM}_{c_k}$ or $\vec{PM}_{c_k,u}$), of contextual condition c_k and $Rec(u, s_j, c_k)'$ is the normalized recommendation score for user u generated by an initial recommender, song s_j and condition c_k if a context-aware recommender is used.

One factor to consider is which similarity measurement to use. In this research we only consider the Euclidean distance. It is a widely used method to calculate the similarity between two vectors, which are the two lists of audio features in our case. The normalized Euclidean distance (d) is defined as follows:

$$d(\vec{s}_j, \vec{GM}_{c_k})' = \sqrt{\sum_{i=1}^{n} (a_{i,s_j} - GM_{c_k,a_i})^2}, \tag{5.4}$$

where a_{i,s_j} represents audio feature a_i for song s_j and GM_{c,a_i} is the average audio feature a_i within the global (or personalized) model for contextual condition c_k, as calculated by either Equation 5.1 or 5.2. This gives us the distance between the song and condition, so in order to obtain a similarity value between 0 and 1 we use:

$$Sim(\vec{s_j}, G\vec{M}_{c_k}) = 1 - d(\vec{s_j}, G\vec{M}_{c_k})'. \tag{5.5}$$

Both $d(\vec{s_j}, G\vec{M}_{c_k})$ and $Rec(u, s_j, c_k)$ are normalized through min-max normalization. This is done in order to have a similarity score between 0 and 1, where 1 means that the two vectors are exactly identical (indicating a suitable match of a song and a condition) and 0 that they are completely opposite (which indicates that a song is not suitable for a condition). Min-max normalization is defined as follows:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{5.6}$$

where, in our case, x represents the similarity value that is being normalized, x_{min} the smallest not normalized similarity value in the set of all calculated similarities and x_{max} the largest similarity value in the same set.

As can be derived from Formula 5.3 to Formula 5.6, a more similar song to a contextual condition (similarity value closer to 1 than 0) gets assigned a higher score than a less similar song (distance closer to 0 than 1). This also depends on the value of λ of course. The Rec score is normalized through the same min-max normalization as well. This is to make sure that when λ has a value of 0.5, the 2 scores are actually equally weighted.

5.4. TWEAKABLE VARIABLES

Previous sections have explained the global and personalized model and how it is used to calculate the re-ranking score. What is missing is an overview of the various variables and parameters that are involved and how they are adjusted. Moreover, since the re-ranking algorithm uses the recommendation lists of the initial recommendation system, the final re-ranked results are also affected by parameters like the amount of re-ranked top songs in the initial recommendation list. This section gives an overview of all these variables in different parts of the pipeline, how they impact the results and how we adjust them.

5.4.1. INITIAL RECOMMENDATION RELATED

These parameters are related to the initial recommendation system and the recommendations that it outputs. This output of recommendations is subsequently used as input for the re-ranking system. Also the accuracy of the re-ranked recommendation will be compared to the accuracy of these initial recommendations, so it is important to consider all variables that influence the output.

RECOMMENDATION ALGORITHM SPECIFIC PARAMETERS

The very first choice that impacts the re-ranking results is selecting which initial recommender algorithms are used in the experiments, as described in Chapter 4. Each recommender algorithm has its own specific parameters that will influence how recommendations are created and thus the recommendation output. For the k-nearest neighbour algorithms, for example, the amount of nearest neighbours must be selected. For matrix factorization based algorithms, the amount of latent factors, also known as dimensionality, needs to be selected for the rating matrix decomposition. All iterative recommender algorithms need to have an amount of iterations or a specific threshold that will tell when the iterations need to be stopped etc.

RECOMMENDATION LIST SIZE

Another variable that needs to be determined is the size of the initial recommendation lists. In the case of the InCarMusic (see Chapter 4 dataset, there are only 139 songs in total. For these amounts of songs, it is no problem to include all 139 for each recommendation to be re-ranked. The #NowPlaying-RS dataset, however, is significantly larger. It is necessary to use a limit for the amount of songs in each recommendation to re-rank. The largest recommendation list size we use here is 200.

Next to re-ranking all 139 and 200 songs for each initial recommendation, it is interesting to take smaller recommendation list sizes and see how the list size influences the re-ranked results. Only ranking the top 25 songs for each recommendation list might give a better or worse performance than re-ranking the top 200 songs. Therefore, we use recommendation list sizes of 25, 50 and 139 for the InCarMusic dataset and 25, 50, 100 and 200 for the #NowPlaying-RS dataset.

5.4.2. RE-RANKING RELATED

These parameters are related to how the re-ranking system models contextual conditions and calculates the re-ranking scores that will be combined with the initial recommendation scores. These directly influence the new scores of the songs and thus their rank and ultimately the accuracy of the re-ranked recommendation list.

CONTEXTUAL DIMENSION

The re-ranking algorithm calculates similarity scores based on the audio features of a song and the audio features representation of a contextual condition. The algorithm can take in any condition as input, provided that there are enough historical ratings available in that specific condition to build an accurate representation. One thing to consider here is which conditions and thus which dimensions the algorithm should use. Some conditions might have a stronger impact on user preference than others. Or perhaps some conditions in the dataset have limited historical ratings available, which make their representations less trustworthy. It is essential here to not randomly pick conditions to use, but rather to select them based on logical reasoning.

To keep things simple, we only use one dimension for each re-ranking execution. We will be running the re-ranking algorithm with different dimensions for the InCarMusic dataset to compare them, but these will be independently run. The last section in this chapter will give some suggestion on how to run the re-ranking algorithm with multiple dimensions, but for this research we only use one dimension at a time.

λ COEFFICIENT

As explained in Chapter 5.3, the λ value balances the the initial scoring with the re-ranking scoring. The goal of this parameter is to measure how the added value of the re-ranking scoring compares to the value of the initial recommender score. The value of λ ranges from 0 to 1. Where for a value of 0, the re-ranking algorithm will only use the initial recommender algorithm's score and for a value of 1 only the re-ranking score is used. In our evaluation we want to compare different values ranging from low importance of the re-ranking score to using the re-ranking score exclusively. To do this we use lambda from 0 to 1 using steps of 0.1 in our experiment.

5.5. OPPOSITE RE-RANKING SCORING CALCULATION

Section 5.3 explained how exactly the re-ranking algorithm works. In this section we introduce an alternative version that does the opposite of the re-ranking algorithm. The formula for this opposite re-ranking algorithm is as follows:

$$opposite\_score = \lambda * d(\vec{s_j}, G\vec{M}_{c_k})' + (1-\lambda) * Rec(u, s_j, c_k)', \tag{5.7}$$

where all variables are the same as described previously. The main difference is that here $d(\vec{s_j}, G\vec{M}_{c_k})'$ is used directly instead of using the similarity by subtracting it from 1. So in this version of the algorithm, songs that are highly similar to a contextual condition get a low score, which results in a lower ranking than songs that are less similar. All other variables and evaluation configurations are kept the same to have a fair comparison with the normal re-ranking algorithm.

The goal of this opposite version is to support the original hypothesis of the re-ranking algorithm. In this case, our hypothesis is that the re-ranking performance goes down compared to the original re-ranking algorithm. Also we expect that the performance is worse compared to the initial recommender algorithm. If this is the case, it shows two things. The first being that the results of our re-ranking algorithm did not happen by chance. The second is that the performance does not always increase because of additional information, audio features in our case, that has been introduced in the recommendation process. In our experiment we test this algorithm and compare the results as well.

5.6. OTHER IDEAS

This section provide additional ideas for the re-ranking algorithm that are interesting to test and might improve performance. However, since we already have quite some features to implement, variables to tweak and a limited scope, we leave these here for interested readers.

WEIGHTED AUDIO FEATURE REPRESENTATION

With both current re-ranking models, the average of all audio features of songs that have been listened to in a certain contextual condition is taken to represent that condition. For an implicit dataset not much else can

be done here. For a explicit dataset, however, it might be interesting to look at a weighted version. Since in explicit datasets, the positive songs can be divided again into how positive they are. A song that has been rated with 5 stars out of 5 is more positive than a song that has been rated with a 4, but both are still positive songs. In such a case, the model can use a weighted audio features representation based on how positive songs were rated in a given condition.

AUDIO FEATURE SELECTION

As seen in Chapter 3 not all audio features are equally descriptive for specific contextual conditions. For example, the audio feature energy is a good in differentiating between happy and sad songs, but the audio feature tempo is, surprisingly, not. Since we saw in the analysis that key and liveness are generally bad differentiators, we left the out. For the eight other audio features, however, we do not distinguish between them. This idea is like the previous idea, except that in some situations it might be better to leave an audio feature out completely.

AVERAGING INDIVIDUAL SIMILARITY SCORES

This model basically operates the same as the global-model average audio features, but with 1 distinct difference. Whereas the previously mentioned model averages all the audio feature of songs that have been positively rated in a certain context, this method will not average these values. Instead of averaging the values and then calculating the distance between the given song audio features and the average audio features of a context, this method first calculates the distances between the song audio features and the audio features for the positively rated songs for all positively rated songs in a given context. After having all these separate distances, we take the average distance to use as representation for the re-ranking part. Although this gives much more overhead, we expect this method to be more accurate.

DIFFERENT SIMILARITY/DISTANCE METRICS

Another interesting change can be to use a different similarity or distance measurement than the Euclidean distance. One other distance metric is the Cosine similarity. Qian et al. [88] has shown through both theoretical analysis and experiments that the Euclidean distance and Cosine similarity are very similar when used in high dimensional settings. Increasing the dimensionality from 2 to 8 makes the two measurements significantly more similar. So here we do not expect the differences to be significant. However, the Cosine similarity is not the only alternative out there. Other potential metrics include the Chebyshev distance, Manhattan distance and the Jaccard distance, among others.

6

EXPERIMENT

Having defined the methodology, re-ranking algorithm and two user preference models, it is time to carry out the experiment. This chapter describes all steps within the experiment, elaborates on choices made and evaluates the re-ranked results. First, Section 6.1 gives a complete overview of all steps that are taken in the experiment. Section 6.2 gives an overview of how the data is processed and formatted before it can be used as input for the initial recommendation algorithms. The system that is used to execute these algorithms is described in Section 6.3, while Section 6.4 gives an in depth description of the structure of the output, which forms the input for the re-ranking system. Section 6.5 describes all steps within the re-ranking system and Section 6.6 motivates the selected evaluation metrics. Section 6.7 concludes this chapter by describing and evaluating the experimental results.

6.1. PIPELINE

Figure 6.1 shows the whole pipeline of the experiment. The selected datasets are the starting point. First they are gathered and cleaned before they can be used as input to the initial recommender system. Different recommendation methods can be used in this system. This recommender system proceeds to output initial recommendation lists, which is used as input for the re-ranking system. Next to this, also the initial contextual data and audio features of the songs in these datasets are used as input for the re-ranking. The re-ranking system then produces the final recommendation lists, which are evaluated against the initial recommendation lists. Next sections describe the different components within the pipeline in more details.

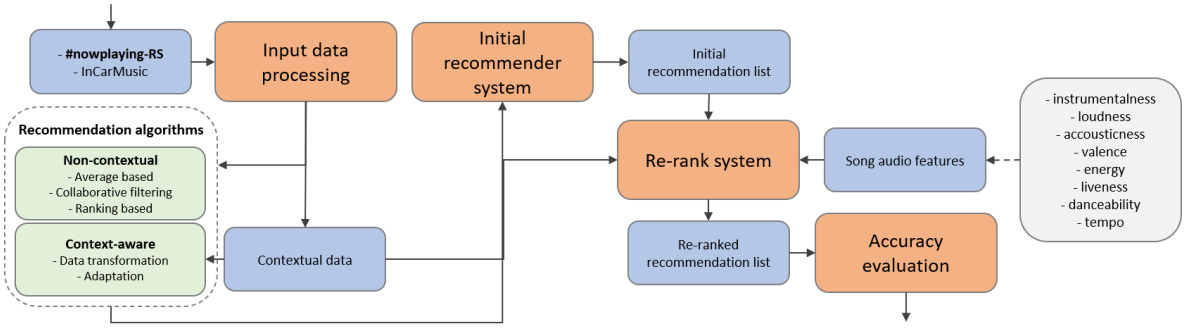


Figure 6.1: An overview of the experiment pipeline from input data to the re-ranked recommendation list.

6.2. INPUT DATA PREPROCESSING

As elaborated in Chapter 4, the #NowPlaying-RS and InCarMusic datasets are the most suitable datasets that could be found for our research. Both datasets were supplied in completely different formats and contain a lot of metadata. The InCarMusic dataset contains a diverse list of phone variables, while the #NowPlaying-RS contains hashtags and tweet language for example. Before these dataset can be used as input for both the initial recommender as well as the re-ranking system, they need to be properly filtered and formatted. All the

following steps described in this section have been performed using Python 3.7 on a Windows machine. The full code can be found on GitHub.[1]

#NOWPLAYING-RS

The full #NowPlaying-RS dataset contains more than 11 million listening events and 4 million users, which is too much for our experiment to handle. The whole file is bigger than 2.3 GB, which means that loading it already takes many minutes, let alone running algorithms on it. Next to this, not all tweets include time stamps or user location. Also, not all song in the dataset have a complete list of audio features. So it makes sense to filter out the missing data points and take a subset that is suitable for our experiment needs.

So first what we did is only keep users that have at least 3000 listening events in total and songs that have been listened to at least 300 times. These filters make sure that the users and songs that are selected have enough interactions to be used in the initial recommendation and re-ranking system. Not all information within the dataset is required, so we only take the user, song, created_at, time_zone and audio features for each listening event. When looking up the Twitter API, it says that the all the created_at values are GMT values to avoid issues with clock changes in different regions. Luckily the tweets also contain the time_zone variable. Based on the time zone of the user, the local time of the tweet can be calculated by either adding or subtracting the amount of hours that the time zone differs to the GMT. In this step we also group the times of the tweets to either morning, afternoon, evening or night. Morning represents all tweets that have been created between 6:00 and 11:59, afternoon between 12:00 and 17:59, evening between 18:00 and 23:59 and night between 00:00 and 5:59.

Now we have a subset of the dataset with the relevant variables and right creation times grouped in the contextual condition *time of day*. Users often listened and tweeted about the same song during the same contextual conditions. Since we do not make any distinction between on which date users have listened to songs (only to the time within a day), we remove all duplicate listening events. All listening events that contain songs of which the audio features are empty are excluded as well. Next to this, we map all the user and song ids that are left to a regular number. The #NowPlaying-RS dataset used long strings for song ids and large numbers for users. We saved the mapping in CSV files so later on in the pipeline we can refer back to the original users and songs if necessary.

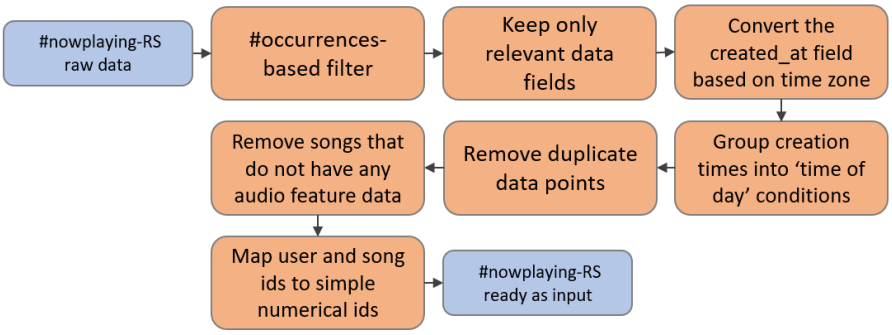


Figure 6.2: An overview of the input data processing pipeline for the #NowPlaying-RS dataset from raw data to more structured ready as input data.

After all the processing and filtering, the final subset of the #NowPlaying-RS dataset that is used as input contains 333 users, 7304 songs and 108,202 unique contextual interactions. The average amount of ratings per user per contextual condition, like Table 4.6 gives for the InCarMusic dataset, is shown in Figure 6.1 for this subset. This subset should be a fair representation of the whole dataset. In Section 6.5 we describe how multiple folds are used in order to fully utilize the subset and reduce bias and variance. The full data processing pipeline of the #NowPlaying-RS dataset is shown in Figure 6.2.

INCARMUSIC

Luckily the InCarMusic dataset is more manageable, mostly because of its size compared to the #NowPlaying-RS dataset. One drawback here is that the dataset itself does not contain any audio features of the songs that have been rated. So the biggest challenge is to extract the right audio features for each song. For this we used the search function in the Spotify API. It can take both song author as well as title as input. So

[1] boninggong/DataPreProcess

Contextual Condition	Avg #ratings/user
morning	213.16
afternoon	216.16
evening	215.3
night	201.7

Table 6.1: The average amount of ratings per user in a given contextual condition for each condition.

we took all song authors and titles, formatted them as consistent as possible, and used it to extract audio features. For the ones that did not return any results, we manually looked them up on Spotify, noted their Spotify id and looked up the audio features separately. Next to extraction of the audio features, we made sure that all contextual conditions from the eight dimensions are consistent. The specific dimensions and their conditions are described in Section 4.1.

Now after having all the right data in a consistent format, the next step is to format the data to be suitable as input for the initial recommender system. Table 6.2 shows the input format that is accepted by this system, which is described more in-depth in the next section. Under the hood, it converts the input to a binary format for quicker and more efficient processing. After all the pre-processing we proceeded to format both filtered input datasets in the loose format so it is ready to be used for the initial recommender system.

(a) The loose input data format

User	Item	Rating	Dimension	Condition
1	item_1	2	Activity	Working
1	item_1	2	Time_of_day	Morning
2	item_2	5	Activity	Running
2	item_2	5	Time_of_day	Evening

(b) The compact input data format

User	Item	Rating	Activity	Time_of_day
1	item_1	2	Working	Morning
1	item_2	4	Reading	Evening
2	item_2	5	Running	Evening
2	item_3	1	Working	Evening

Table 6.2: The two possible data input formats for the recommender system.

6.3. INITIAL RECOMMENDATION SYSTEM

In order to use the re-ranking system, an initial recommendation list is needed as input. For our research the CARSKit[2] by irecsys on GitHub is used to create the initial recommendation lists. CARSKit is a recommendation engine that has been created in Java and is open-source. It can be used, modified and distributed under the GNU General Public License. It contains both context-aware as well as traditional recommendation algorithms, but the focus of it is on context-aware recommendations. The traditional recommendation algorithms within CARSKit are taken from LibRec-V1.3.[3]

As described in the previous section, CARSKit takes input datasets in specific formats, either in the loose format or the compact format. It does not matter which format is used, as long as the data is consistent. See figure 6.3 below for a complete overview of how exactly CARSKit is designed.

CARSKit also has the option to use a simple k-fold cross validation. When this is enabled, the system divides the dataset in k approximately equal sized subsets. When training a model, it is trained on (k-1) subsets and the performance is evaluated on the last subset that has not been used for training. This can be done k times, where each time another subset is left out of the training phase, but is used as test set. The average results of all these k executions is subsequently averaged to get a general performance for the model. This is especially useful to increase the robustness of results for sparse and smaller datasets, like the InCarMusic dataset. When computational duration is significant and there is a large enough dataset, e.g. more than 10,000 data points and > 1 hour of runtime, it is encouraged to use 5-fold or 10-fold cross validation [89]. Since our main dataset contains over 100,000 listening events, we will be using 5-fold cross validation during our experiment.

In our research we have made some small modifications to the original CARSKit system to fit our needs. These include the following:

- Added a Random recommendation algorithm, where items are assigned a random integer between 1 and 1000 and are ranked based on this number.

[2] https://github.com/irecsys/CARSKit
[3] https://www.librec.net/

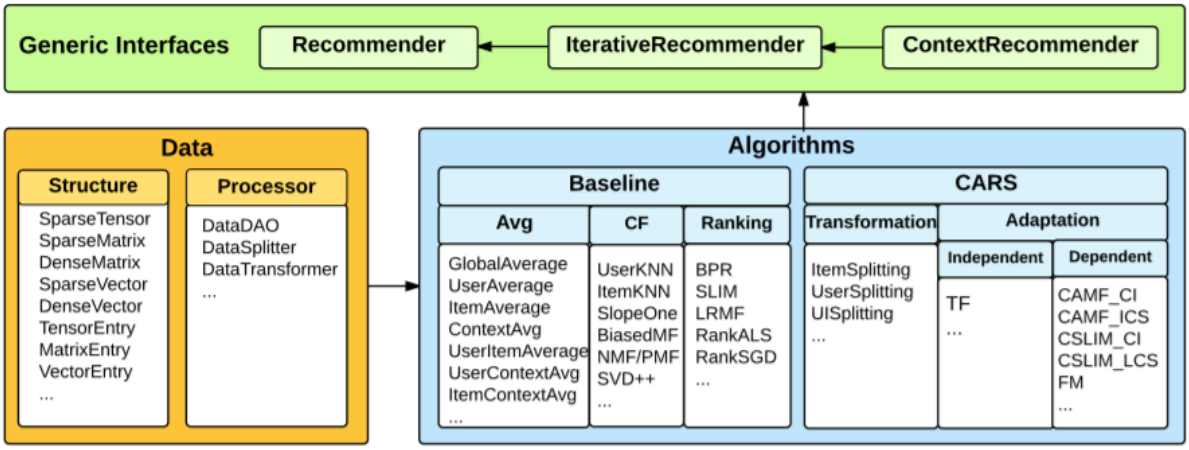


Figure 6.3: An overview of the architecture of the CARSKit library as described in [1]

- Added a ItemPopularity recommendation algorithm, where items are ranked based on the amount of ratings/interactions that they have had.
- Modified that the first line of any output recommendation list contains an integer, which represents the amount of recommended items for each user-song(-context) combination.
- Modified that the header of the recommendation lists contains unique column names for each recommended item, e.g. 'p1' for the recommended item on position 1, 'p2' for the second etc.
- Modified that the system also outputs the fold that has been left out in each iteration over all the folds. Our re-ranking system needs to know for each iteration which ratings/listening events are used for evaluation of performance.
- Modified that next to the specific fold, the system also outputs another file containing the index of all positive items across all recommendation lists. This data is used to visualize the distribution of positive items for the initial recommender algorithm to get a general idea about the recommendation accuracy.

Besides the above mentioned modifications, no other changes were made. All the recommendation algorithms are used as is, in their original forms. The modified version that we created and used in our research is accessible through GitHub.[4]

ALGORITHMS USED

Many types of both traditional as well as context-aware recommendation algorithms are available in the CARSKit library. A subset of them are shown in Figure 6.3. Since the InCarMusic dataset contains explicit feedback, we are able to select any initial recommendation algorithm. However, this is not the case for the #NowPlaying-RS dataset, since it only exists of implicit feedback. So that is why different initial recommendation algorithms are used for each dataset. For each dataset, we have selected three algorithms to create the initial recommendation list. We proceed to elaborate on the selected algorithms for each dataset.

The algorithms that are selected are a mix of context-aware and non-context aware recommender algorithms. The reason for using using a mix of the two types was explained in Section 4.2. In summary, both are used to create baseline recommendation lists and for the context-aware recommendation it is interesting to compare how our context and audio features based re-ranked recommendation compares to existing context-aware recommender algorithms. Each of them have been described more in-depth in the literature review.

The following three initial recommendation algorithms were used for the #NowPlaying-RS dataset to create the initial recommendation list:

1. **BPR**: A simple ranking based algorithm using Bayesian probabilities [35].

[4] https://github.com/boninggong/CARSKitModified

2. **UserSplitting-BPR (US-BPR)**: A contextual algorithm that first splits users in sub profiles based on contextual conditions and then executes the BPR algorithm [54].

3. **CAMF_ICS**: A specific type of ranking based matrix factorization that takes context into account based on underlying similarities of conditions within the same dimension [90].

All three of them are ranking based algorithms, which do not require explicit ratings to create recommendations. BPR does not use context, while the other two do. The subset that we are using has thousands of songs that can be recommended. It would be too computational expensive and useless to output every recommended song for each user-song-(condition) combination. So for each initial recommendation we limit the output to the top 200 songs per each user-song(-context) combination. The re-ranking system, described later on, splits these lists of 200 songs to subsets with smaller amount of recommendations. Each of those is re-ranked separately. Since we are using 5-fold cross validation, we have 5 separate files containing initial recommendations for each row that is made up of 200 recommended songs each.

Since the InCarMusic dataset has explicit feedback, we are free to choose any initial recommender algorithms. The following three initial recommendation algorithms were used for the InCarMusic dataset to create the initial recommendation list:

1. **UserKNN**: A simple collaborative filtering algorithm based on user similarity [24].

2. **BiasedMF**: A specific type of matrix factorization that takes a bias into account [28].

3. **CAMF_CU**: A specific type of matrix factorization that takes context into account based on their influence on users [53].

These three initial algorithms are selected based on their varying mechanisms and characteristics associated. UserKNN is the most simplistic one that is re-ranked and compared. This gives use insights into what kind of impact the re-ranking has on simplistic collaborative filtering methods. Matrix factorization is more complex form of collaborative filtering, so it would be interesting to see how it will be impacted by the re-ranking and how it compares to the results of the more simplistic form of collaborative filtering. Last, but not least, we have included an initial recommendation algorithm that uses context itself. In this case, we output the full recommendation lists, because the dataset only contains 139 songs in total. These lists will also be split into smaller subsets to be re-ranked individually.

6.4. INITIAL RECOMMENDATION LIST

Table 6.3 shows the format of the initial recommendation lists. The outputs start with just an integer in the first row. This integer indicates how many recommendations are given on each row and is used by the re-ranking system to correctly process the input. The second row contains the header. The first column exists of our simplified user identifiers. The second column contains all positive or 'correct' songs for a given user in a given context. These positive songs are gathered from the test set and thus are not used in training the recommender model.

The correct songs are connected by a semicolon in between each. After that, the contextual situation is described in the third column. Since we limit our research to only one condition at a time, it only outputs one dimension followed by one of its conditions. The contextual column is followed by all columns of recommended songs. For each song, the prediction/score is appended to the song identifier with a semicolon in between.

200						
userId	**correctItems**	**contexts**	**p1**	**p2**	**...**	**p200**
12	1106;1650;2248;293	daytime:morning	118;9.113	544;8.830	...	1583;6.958
12	91;1554;293;140	daytime:night	118;9.113	544;8.830	...	1583;6.958
...	...	...	...	...	...	...
702	3287;2549	daytime:afternoon	167;9.367	1127;9.208	...	3287;7.232

Table 6.3: Format of the initial recommendation lists that the initial recommendation system gives as output.

6.5. RE-RANKING SYSTEM

Figure 6.4 gives an overview of the main components and processes within the re-ranking system. The blue boxes indicate CSV files while white boxes represent certain components/processes. In our experiment we ran the whole re-ranking system for each initial recommendation algorithm and each of the sizes of the initial recommendation lists. So in our case with the 3 algorithms of BPR, UISplitting-BPR and Random and 4 list sizes of 25, 50, 100 and 200, we have run this a total of 12 times. The system has been built using Python 3.7 on a Windows machine. The implementation is publicly available on GitHub.[5]

The starting point for the re-ranking system is by reading in the output, the initial recommendation list, of the initial recommender system and use it as input. Next to this, the system also takes the song audio features and formatted version of the original dataset as inputs. In order to reduce the total run-time, differently formatted versions of the formatted dataset are also used as input. For example, there is a file filtered on user-contextual condition pair, where each user-contextual condition pair is followed by all the songs that that user has listened to while in the specific condition. These additional input files are all based on the original dataset, no new data has been introduced.

The first step is of the system is to create the different re-ranking models. Since we are using 5-fold cross validation, where each fold has different training and test data, we have to create the models separately for each fold. For the global model, for example, we go through all the known listening events in the training data for each fold. Based on the songs that have been listened to in the specific conditions, a global model is made through an representation of audio features as described in more detail in Section 5.1. For the personalized model, the same is done for each user for each fold. A more detailed description was given in Section 5.2. The required amount of calculations grow exponentially with the amount of users, ratings and contextual conditions. That is another reason where the pre-processed files, as described in the previous paragraph, come handy.

After having the models ready, it is time to calculate the $DIST(I_{af}, C_{af})$ part of Equation 5.3. For this, the Euclidean distance is applied to measure the distance between the audio feature vector for each song and the audio feature vector representation of a given contextual condition. In the case of the personalized model, the audio feature vector representation of a given contextual condition for a given user is used. These resulting scores represent the re-ranking scores.

Now we have the initial recommendation score and our newly calculated re-ranking score based on the underlying audio features and context in which they are rated. The re-ranking equation combines them and output a new score for each song for a given user-contextual condition pair. Since both the initial recommendation score as well as the re-ranking scores can have greatly varying values, we normalize them separately using also Equation 5.6. So now both scores are in a range between 0 and 1 and they have an equal weight when combining them using the balancing parameter λ. The initial recommendation list is then ordered in an ascending order based on this new combined score. These newly ordered lists form the re-ranked/final recommendation list.

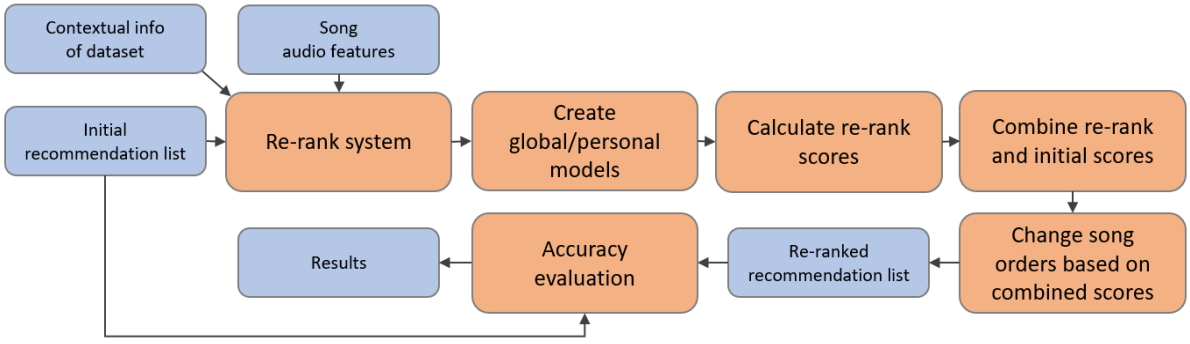


Figure 6.4: An overview of the main steps that happens within the re-ranking system.

6.6. EVALUATION METRICS

Now that we have the re-ranked recommendation lists, it is time to evaluate them. Section 4.4 gave an extensive list of evaluation metrics that can be used for this. Since the #NowPlaying-RS dataset is the more

[5] https://github.com/boninggong/Re-rankSystem

reliable of the two, we decide based on its properties. The dataset contains implicit feedback data, which means that prediction accuracy metrics can not be used. So we are left with the decision support accuracy metrics and the ranking based accuracy metrics groups. We decided to use the precision at k (Prec@k) and mean average precision at k (MAP@k) metrics for our experiment. The ranking of positive songs within a recommendation list matters. These two metrics take the ranking into account by measuring the top k songs instead of the full recommendation list. Precision is the more simple metric and looks at how many songs in the top k are positive songs. The mean average precision does the same, but also takes the relative position of the positive songs into account.

Having selected two different evaluation metrics, the next step is to decide on a useful number for k. This number should not be too small, but neither too big. If a k value of 3 is used, for example, the evaluation scores will be very close to 0. The #NowPlaying-RS subset that we use has thousands of songs that are considered for the recommendation. For a given user-song-condition triplet there are only tens of positive songs in the test set. If only the top 3 songs of each recommendation list are evaluated, the chance is very high that none of the top 3 recommended songs are a positive song. On the other hand, a k value of 50 does not make much sense either. The lower a song is ranked in a recommendation, the smaller the chance that users will interact with it in practice. Spotify, for example, recommends 20 songs based on your previous listening behavior. They show only 6 at a time. Users can use a small arrow to browse further through the other songs if they do not like the first 6. Figure 6.5 shows how this looks like in their user interface. Based on these factors, we have decided to evaluate the top 10 and top 25 recommended songs for both the precision as well as mean average precision metrics. The precision evaluated over the top 10 songs is from now on denoted as Prec@10 and mean average precision over the top 25 as MAP@25 etc. So in the end we evaluate each configuration using MAP@10, MAP@25, MAP@all, Prec@10 and Prec@25.

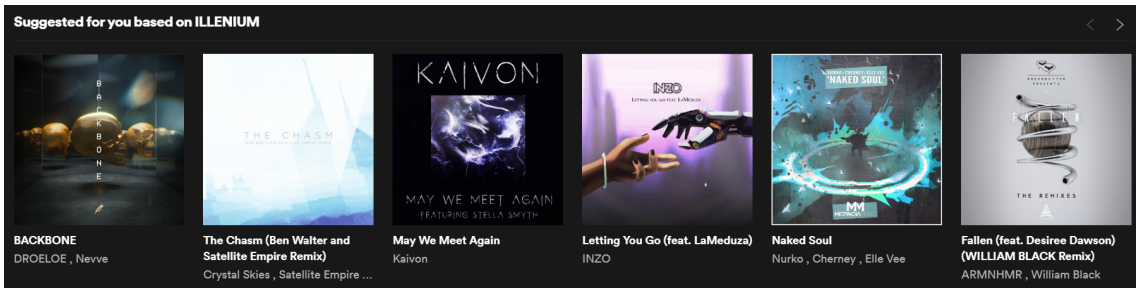


Figure 6.5: An example of recommendations made by Spotify based on your previous listening behavior.

6.7. RE-RANKING RESULTS

This section presents and elaborates on the results of the experiments described in the sections above. First the results of the InCarMusic dataset are presented, which are then followed by the #NowPlaying-RS re-ranking results.

6.7.1. INCARMUSIC

Section 4.1.3 outlined the rationale behind using the InCarMusic dataset [63]. Even though it is limited in size and data of individual ratings, it does provide a wide variety of contextual dimensions that can compared be used to compare to one another. The wide variety of variables (initial recommendation algorithm, global/personalized mapping, initial recommendation list size, contextual dimension etc.) yield many results. So that is why the full results have been put into the appendices. Appendix A contains various results that were obtained from re-ranking recommendation lists that were created using the InCarMusic dataset as input.

The first thing that is noticeable when looking at these results is that the precision and mean average precision scores all are relatively low. Normally a low performance score is a bad result. In our case, however, the re-ranking performance score itself does not matter too much. It is the re-ranking performance score relative to the initial recommendation list performance score that is important. We do not have control over the performance of our baseline, the initial recommendation performance, except for some model specific parameters. An explanation for the low score is that there are often only one or two positive songs for a given user-song-condition triple. The recommendation algorithm has to rank those higher than the other 100+

songs that it can choose from in this case. In many cases the positive song(s) are not placed in the top 10 or top 25 recommended songs, which gives a MAP and Prec score of 0. This in turn lowers the average MAP/Prec score over all recommendation lists.

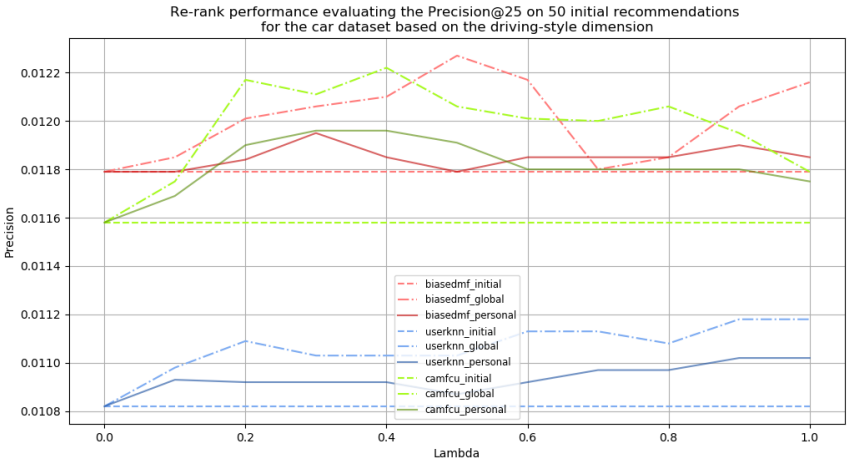


Figure 6.6: **InCarMusic driving-style** dimension based re-ranking results measured using **Prec@25** over the top **50** songs.

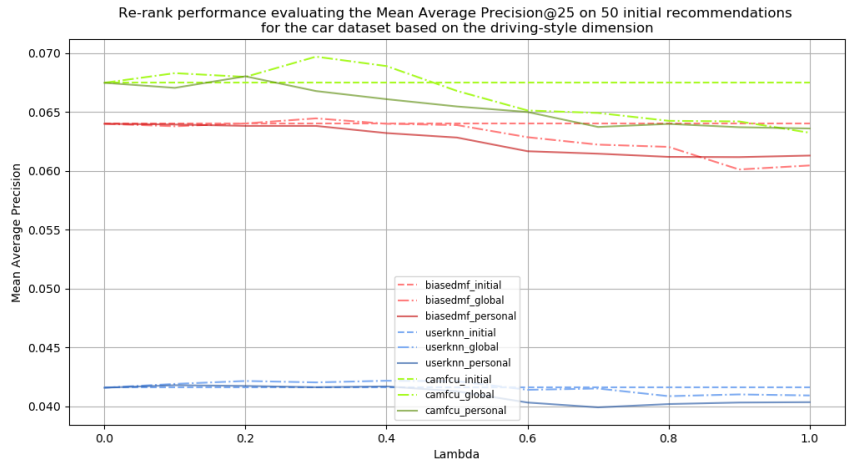


Figure 6.7: **InCarMusic driving-style** dimension based re-ranking results measured using **MAP@25** over the top **50** songs.

There are many different results for different variables, so we visualized a selection of them for more clarity. Figures 6.6 and 6.7 are such visualizations and show the performance of the initial recommendation list (dotted line), the re-ranked recommendation list using the global model (dash-dotted line) and the re-ranked recommendation list using the personalized model (solid line) measured using the Prec@25 and MAP@25 metrics respectively.

The first noticeable observation in these visualizations is that the re-ranked results measured using Prec@25 give an increase in performance compared to the initial recommendation list, while the results when measured using MAP@25 decreases with increasing lambda values. This means there are more positive songs in the top 25 songs after re-ranking, but they are not necessarily placed higher. So what happens is that positive songs that are placed in spots 26 to 50 are re-ranked to one of the top 25 spots, but the rank for these positive songs are not necessarily higher.

Another noticeable observation is that the global mapping based re-ranking algorithm outperforms the personalized mapping based algorithm. At first sight, this contradicts our hypothesis regarding the last sub-research question. However, if we go back to Table 4.6, we see that there are very little ratings per user per contextual condition available. This means that the personalized models, for the users that have enough

historical rating data, were only based on a few ratings and thus highly unreliable. The global model, on the other hand, uses all previous ratings of all users for each condition, so the representation is more accurate. This explains the observed results.

For comparison we have also plotted the same results for the roadtype and weather contextual conditions. They are shown in Figures 6.8, 6.9, 6.10 and 6.11. The main reason for using this sparse dataset is to find out how different contextual dimensions influence the re-ranking results. We chose to use roadtype since it has the second highest average amount of ratings per user, as seen in Table 4.6, which might provide some insight for the personalized model. Weather is included, because we were interested in its potential influence and it has a decent amount of historical ratings available.

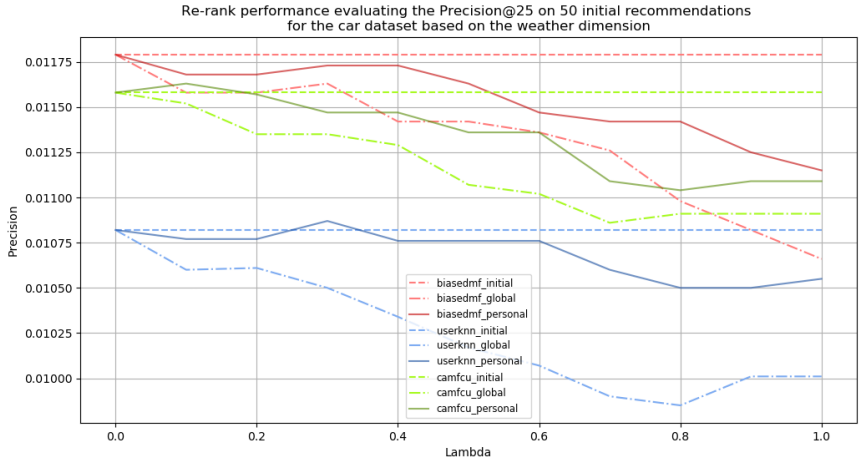


Figure 6.8: **InCarMusic weather** dimension based re-ranking results measured using **Prec@25** over the top **50** songs.

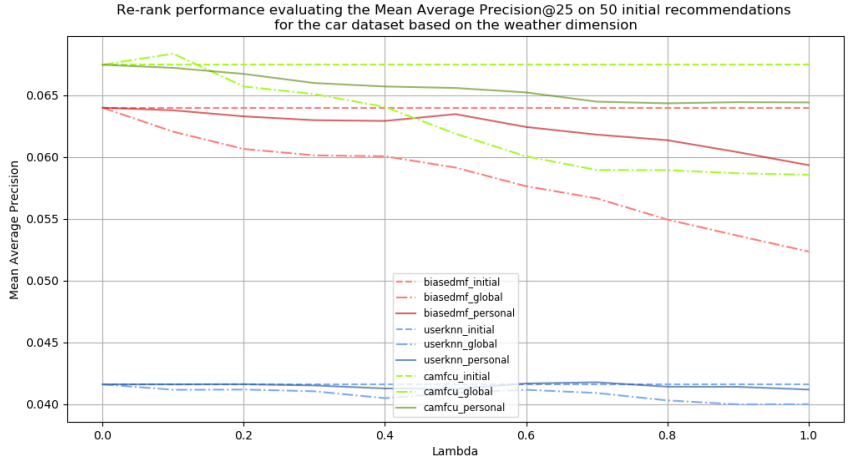


Figure 6.9: **InCarMusic weather** dimension based re-ranking results measured using **MAP@25** over the top **50** songs.

The re-ranking visualizations for the weather and roadtype dimensions show different results compared to the driving-style based re-ranking. Both re-ranking results show a decrease when measured using Prec@25. This means that generally there are less positive songs in the top 25 songs after re-ranking. Which gives an opposite effect compared to the driving-style dimension. There are two main reasons that can explain these results. The first is that both the weather as well as roadtype dimensions have a very occurrences within the ratings. As shown in Table 4.6, the weather conditions have an average of 1.05 ratings per person over all conditions, while the roadtype dimension only has 1.31 per person. Both are lower than the driving-style dimension, which has an average of 1.62 ratings per user. The second possible explanation is that these contextual dimensions might just not be representative for user preference.

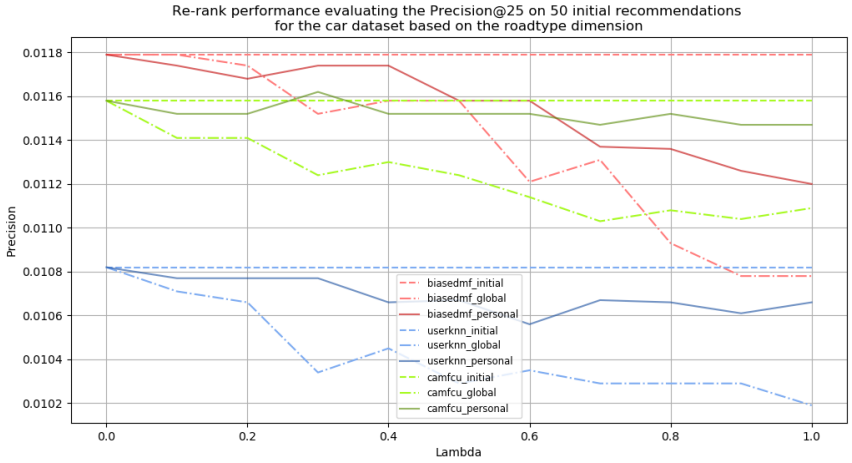


Figure 6.10: **InCarMusic roadtype** dimension based re-ranking results measured using **Prec@25** over the top **50** songs.

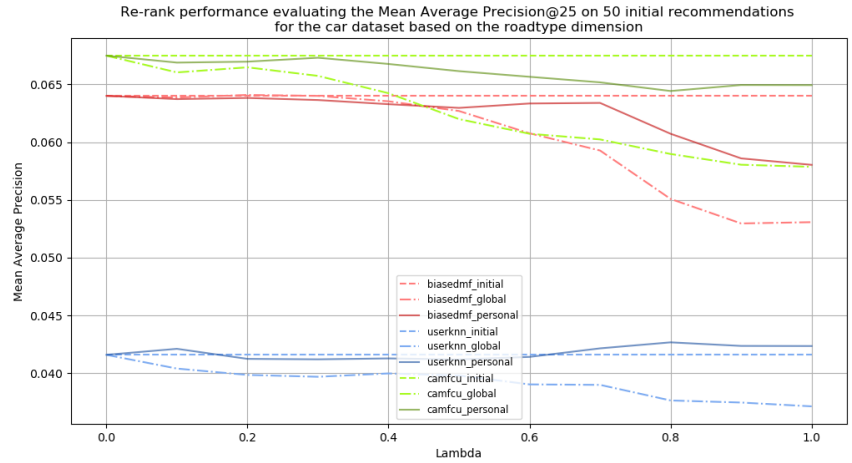


Figure 6.11: **InCarMusic roadtype** dimension based re-ranking results measured using **MAP@25** over the top **50** songs.

Another interesting observation is the fact that all personalized model based re-ranking recommendations outperform the global model based ones. This is also opposite of the results of the driving-style dimension based re-ranking, even though roadtype and weather have a lower average amount of ratings per user.

Since the InCarMusic dataset is so sparse, it is hard to draw sound and reliable conclusions. So we decided to limit the number of experiments that use it and focused more on testing with the #NowPlaying-RS dataset. The next section shows and elaborate on results that were gathered from re-ranking the recommendations obtained from the #NowPlaying-RS dataset, which is a significantly larger dataset, but more limited regarding contextual dimension variety.

6.7.2. #NOWPLAYING_RS

The #NowPlaying-RS dataset is significantly larger than the InCarMusic dataset. Here however, there is only one clear contextual dimension, namely time of day/daytime. For this dataset we have run more configurations compared to the InCarMusic dataset, because the amount of data gives more robust results. All regular re-ranking results can be found in Appendix B, while Appendix C depicts all opposite re-ranking results.

Also here we visualized the results in the appendices for a more clear overview. A selection of these visualization is used and discussed. An overview of all visualizations for each result can be found on my GitHub.[6]

[6]https://github.com/boninggong/Re-rankSystem/tree/master/res/nprs

First we give an explanation on how to read the visualizations, because they contain so much information. Each visualization shows the results of all three initial recommendation algorithms that have been used as input. These are Bayesian Personalized Ranking (BPR), UserSplitting-based BPR (US-BPR) and Independent Context Similarity based Context Aware Matrix Factorization (CAMF_ICS). They can be distinguished by their colors, which are red, blue and purple respectively. For each of these algorithms there are three line plots. The flat dotted line represents the performance score of the initial recommendation list, without any re-ranking. The dash-dotted line represents the score of the global model based re-ranking recommendation list, while the solid line represents the score of the personalized model based version. The x-axis represents the tested lambda values, which ranges from 0 to 1 in steps of 0.1, while the x-axis represents the performance evaluation score, so either the Prec or MAP score. How this can be interpreted in layman terms is that the more to the right you go, the more the final recommendation list has been influenced by the re-ranking algorithm. A lambda value of 1 means only the re-ranking scoring is used to create recommendations, while a value of 0 means only the initial recommender algorithm has been used. The higher the value on the y-axis, the better the performance of the recommendation list.

Figure 6.12 shows the results of the same evaluation metrics that have been shown in the previous section for the InCarMusic dataset. Figure 6.13 shows the same evaluation, but this time for the results of the opposite re-ranking algorithm. Many things are noticeable and each of them is described below.

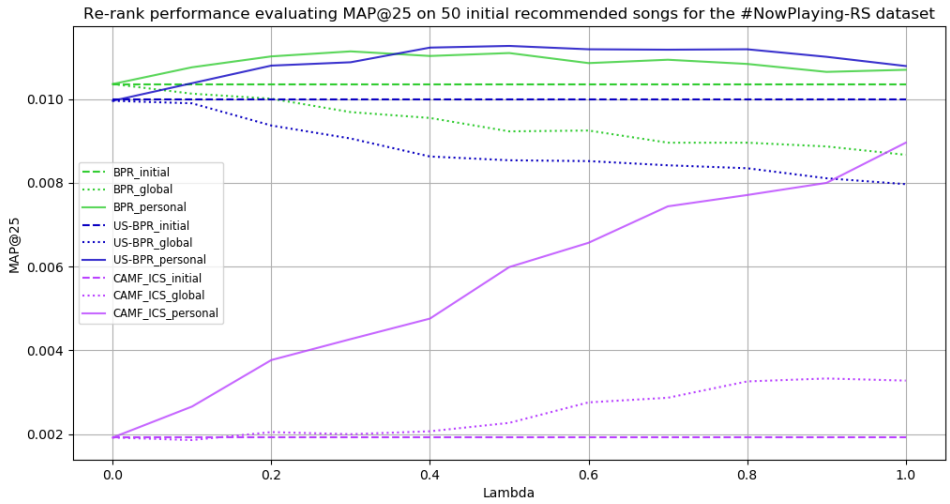


Figure 6.12: #NowPlaying-RS re-ranking results evaluated using **MAP@**25 over the **top 50** recommended songs.

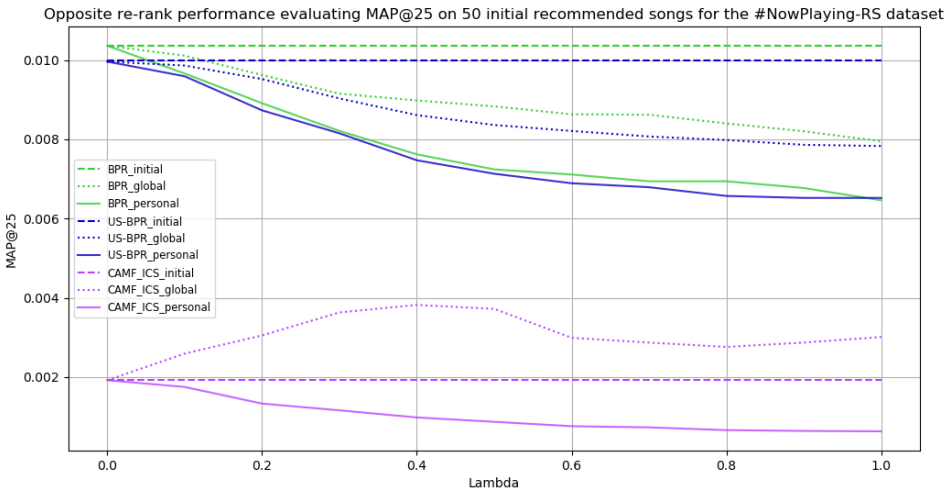


Figure 6.13: #NowPlaying-RS **opposite** re-ranking results evaluated using **MAP@**25 over the **top 50** recommended songs.

The first observation is that the baseline performance of BPR and US-BPR differs significantly from the

performance of CAMF_ICS. BPR performs more than 4 times as well and US-BPR even almost 5 times. US-BPR, which uses the contextual data as extra information in the recommendation process, performs better than BPR only. This is in according to what we expect, since the input is more granular after applying User-Splitting techniques.

Another of them is that the recommendation accuracy of the personalized model generally beats the recommendation quality of the initial recommendation, especially for the CAMF_ICS algorithm. The CAMF_ICS personalized model re-ranking results even outperforms all three BPR results using the same lambda value. An explanation here could be that CAMF_ICS placed relatively more positive songs in spots 26 to 50 than BPR did. The re-ranking algorithm then proceeded to place them in the top 25 songs, which greatly improved the MAP@25 score. That would also explain why the initial MAP@25 score is relatively low. The personalized model, furthermore, consistently beats the performance of the global model, which varies widely for each recommender algorithm. This is in line with our hypothesis that the personalized model is a more accurate model than the global one.

We can also observe that in the opposite ranking, the personalized model greatly underperforms and creates a worse recommendation list than both the initial recommender algorithm and global model. This further strengthens the value of the re-ranking algorithm, since re-ranking on opposite scores yields opposite performance. The global model decreases the accuracy for the BPR and US-BPR algorithms, but, surprisingly, increases accuracy when utilizing the CAMF_ICS algorithm. This increase is relatively high for lambda values between 0.3 and 0.5. We do not have a clear explanation why this is happening and more research would be needed to explain this phenomenon.

Last, but not least, a higher lambda value does not always result in increased accuracy, even for the well performing personalized model re-rankings. It also differs per initial recommender algorithm. For the personalized model of CAMF_ICS a higher lambda value does provide a better performance, but for BPR we observe that the accuracy actually goes down with lambda values higher than 0.3. This means that if this re-ranking algorithm would be implemented in practice, it should be tweaked based on the underlying recommender algorithm. There is no lambda value that always give the best re-ranking results.

Figure 6.14 and Figure 6.15 show the same algorithms, but evaluated using MAP@10 over the top 25 songs. As explained in Section 6.6, there is a higher chance that users interact with songs that are placed in the top spots in a recommendation. Having positive songs in the top 25 is good, but in the top 10 rankings is even better.

These results are fairly comparable to the MAP@25 evaluation of recommendation lists consisting of 50 songs. Here the personalized model results consistently outperform the initial baseline with every lambda value. The global model based re-ranking results for BPR and US-BPR are relatively better than the MAP@25 evaluations. Also the opposite re-ranking results show a steep decrease in accuracy for BPR and US-BPR and again show an unexpected increase for the global model for CAMF_ICS.

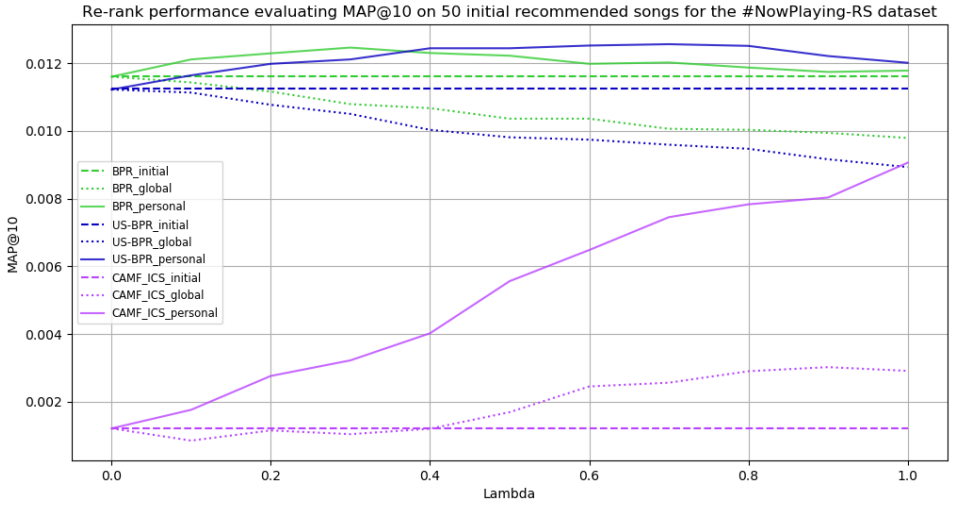


Figure 6.14: #NowPlaying-RS re-ranking results evaluated using **MAP@**10 over the **top 25** recommended songs.

All previous four figures showed the MAP@k evaluations, which tells both how many positive songs can be found within the top k recommendations as well as how they are ranked relatively to each other. Looking at

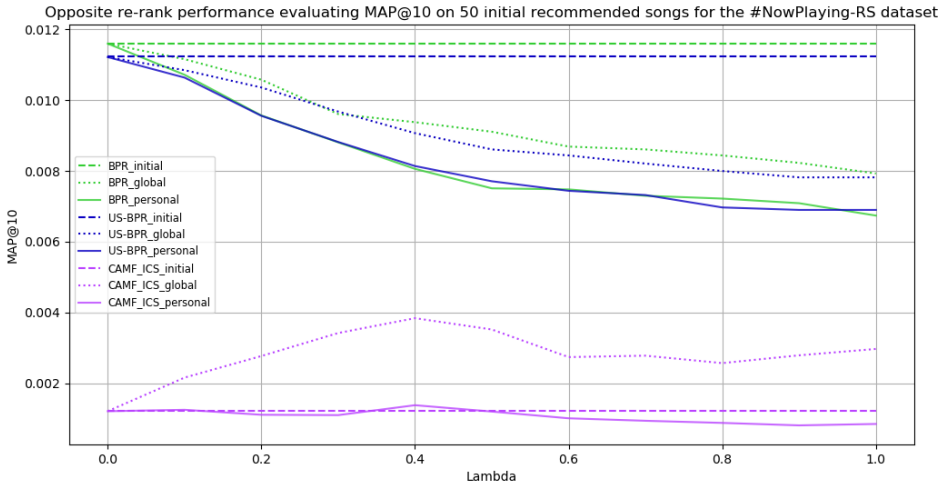


Figure 6.15: #NowPlaying-RS **opposite** re-ranking results evaluated using **MAP@**10 over the **top 25** recommended songs.

the scores, however, it does not tell whether the amount of positive songs in the top k positions have increased or decreased. That is why the precision evaluation metric has been included in our experiments.

Figures 6.16 and 6.17 show the same experiments evaluated using the Prec@10 metric. These results differ a bit from the MAP@10 results. One of the observations here is that BPR itself is actually performing slightly better than US-BPR. So when looking at the amount of positive songs in top 10 positions, the UserSplitting did not manage to rank more positive songs there. On the contrary, it decreased it slightly. Another observation is that both global and personalized model based re-ranking results for BPR are showing generally decreasing accuracy. In Figure 6.14 we saw that for the MAP@10 metric, these gave an improvement. Combining these two figures, we know that there are actually less positive songs in the top 10 recommendations after re-ranking, but the songs that are there do get a higher position. Also here in the opposite re-ranking results we can observe that the performance decreases for all re-ranking results for both the BPR and US-BPR algorithms. One interesting and different result is the opposite performance for CAMF_ICS. Here, the global model based opposite-re-ranking performance is not improving like it does in the cases of Figures 6.13 and 6.15. This tells us that apparently the amount of positive songs does not change compared to the baseline when running the opposite re-ranking algorithm. However, positive songs are placed relatively higher.

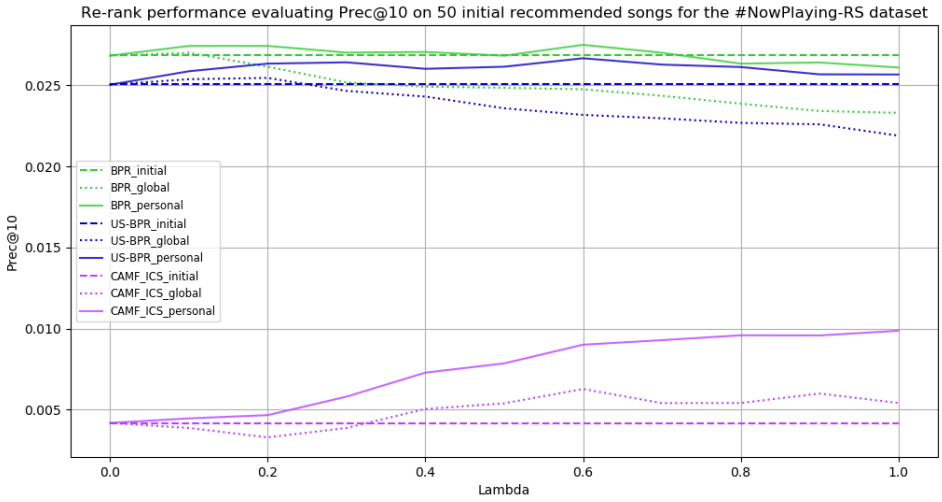


Figure 6.16: #NowPlaying-RS re-ranking results evaluated using **Prec@**10 over the **top 50** recommended songs.

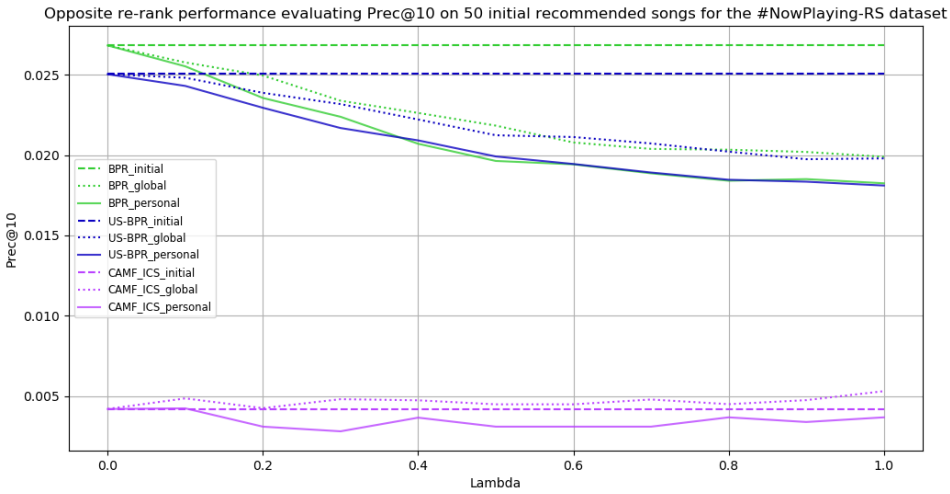


Figure 6.17: #NowPlaying-RS **opposite** re-ranking results evaluated using **Prec@**10 over the **top 50** recommended songs.

7

Conclusion & Future Work

This chapter gives a wrap-up of our research on contextual re-ranking of music recommendations through audio features based user preference models. In Section 7.1 we give a brief recap of the research questions and conclude our work by answering each. Next, we discuss some limitations of the research and decisions we have made in our work in Section 7.2. The chapter ends with Section 7.3, which points to potential directions for future work.

7.1. Conclusion

Before concluding this research, we first recap the goal, which was to answer the following research question:

- **Main Research Question:** How can the relation between audio features and contextual factors be used to improve music recommendation quality through re-ranking?

In order to answer this main research question, we defined three sub-research questions. During this research we carried out multiple analysis and experiments to address these questions. Each of the the sub-research questions are stated below together with a recap of our findings.

- **sub-RQ1:** How are contextual conditions of different contextual dimensions related to audio features?

In Chapter 3 we describe the extraction, visualization and analysis of audio features for various contextual dimensions and their respective conditions. Generally, audio features show a significant correlation when comparing conditions. This means that they are suitable for representing user preferences for different contextual conditions.

Even though a majority is significantly correlated, many differences exists among different dimensions and audio features. Some audio features are strong descriptors to distinguish between two conditions, while others are weaker, dependent on which two contextual conditions are compared to each other. For all the conditions that we analyzed, we conclude that the audio feature *key* is not a good descriptor and *liveness* only in a few cases. The effectiveness of the other audio features differ per condition. Table 3.4, for example, shows that *instrumentalness* is the audio feature that distinguishes afternoon and night strongest. A low value means it is more suitable to an afternoon condition, while a higher value suits night conditions better. Also in the comparison between night and evening in Table 3.7 a low *instrumentalness* value is more representative of the evening condition. Low *instrumentalness* both suits afternoon as well as evening conditions, so it is not a good audio feature to use when comparing afternoon to evening conditions, as can be seen in Table 3.2. In this case, *energy*, *valence*, *danceability* and *acousticness* are better. Another observation is that since the audio feature representation of conditions differ greatly, it is easier to compare some conditions than others. Table 3.8, for example, shows that the audio features of relaxing and sleeping are relatively similar. This makes it harder to distinguish between these two conditions. On the other hand, there is the happy and sad comparison, depicted in Table 3.14. Here, there are multiple audio features which are strong descriptors that can be used to distinguish between sad and happy moods.

- **sub-RQ2:** How does re-ranking, based on audio feature representations of user preferences in different contextual conditions, affect music recommendation accuracy?

We designed and evaluated a re-ranking algorithm that uses audio feature representations of user preference in contextual conditions, as described in Chapters 5 and 6. The re-ranking results differ strongly for different user preference models and initial recommender algorithms. Generally, after choosing the right models and parameters, the re-ranking algorithm is able to increase the recommendation accuracy. We proceed by pointing out a few examples from the results.

Figure 6.12 shows that the personalized model based re-ranking algorithm, with a lambda value of 1, managed to get a evaluation score that is more than 4 times as accurate than the initial algorithm did. On the other hand, if we take the same parameters and apply it to the BPR algorithm, the re-ranking results actually are less accurate than the initial recommendation's results. When comparing various Prec@k and MAP@k results, we observe that the Prec scores are relatively lower than the MAP scores for the re-ranked recommendations. This indicates that the re-ranking results rank positive songs relatively higher in a given recommendation lists, but it does not necessarily place more positive songs in the top 10 or top 25.

The opposite re-ranking results, as depicted in Figures 6.13, 6.15 and 6.17, also supports the same conclusion. The personalized model based opposite re-ranking results consistently underperforms the global model version and the initial recommendation lists for the BPR and US-BPR algorithms. This further strengthens the positive results of the regular re-ranking algorithm. CAMF_ICS forms a curious case, where the personalized model based opposite re-ranking decreases performance, but the global model increases performance. Altogether these experimental results, based on various initial recommender algorithms, show that there is merit in re-ranking recommendations using audio features as user preference representations in contextual conditions to increase the accuracy of music recommendations.

- **sub-RQ3:** How do global audio feature representations of user preferences in different contextual conditions affect the re-ranking results compared to personalized audio feature representations of user preferences in the same contextual conditions?

We defined a global and personalized model to represent user preferences in contextual conditions in Chapter 5 and evaluated them in Chapter 6. From the results we can conclude that the personalized model clearly outperforms the global model. It does this consistently across different initial recommender algorithms. In the opposite re-ranking results, the personalized model consistently performs worse than the global model. This further strengthens the accuracy of such a personalized user preference representation. All performance evaluations can be found in the appendices and a selection has been visualized in Chapter 6.

These results confirm our hypothesis that contextual conditions represented using audio features based on previously listened songs are more accurate when they are created for each user individually. Global representations using all historical listening events is less effective. One trade-off to consider is the significantly higher computational costs to create these personalized models. Where the global model creates preference models for each contextual condition, the personalized model does this for every user and condition. Fortunately, this only needs to be done once at the beginning for every user. Afterwards, the model only needs to be updated with new user-song interactions. Based on this and the great difference of re-ranking performance, we think it is a trade-off worth making.

Putting all answers and insights from above together, it is time to come back to the main research question, provide a final answer and conclude our work. The main research question was defined:

- **Main Research Question:** How can the relation between audio features and contextual factors be used to improve music recommendation quality through re-ranking?

First, we showed that audio features are significantly correlated to contextual conditions. Thus, they can be used to distinguish different conditions and represent user preferences. Based on this, we created a re-ranking algorithm that uses these audio features based user preference models to re-rank any given recommendation list. We have evaluated two of such user preference models, a global and personalized model. We re-ranked songs based on the similarity between their audio features and the audio feature representation of user preferences in a given contextual condition. The re-ranking algorithm has been evaluated using the #NowPlaying-RS and InCarMusic dataset using the precision at k and mean average precision at k accuracy metrics. Initial results show that such a re-ranking algorithm is able to increase recommendation accuracy. Especially the personalized model shows promising results and consistently outperforms the global model. This is further substantiated by the opposite re-ranking algorithm, where opposite results were obtained.

7.2. DISCUSSION & LIMITATIONS

Since our re-ranking algorithm uses audio features to represent contextual conditions, its use is limited to the musical domain. This would be a major limitation when looking from the high-level domain of recommender systems. The re-ranking algorithm would, for example, not be useful when trying to recommend books on an e-commerce platform to users, since books do not have audio features that can be used to represent them in specific conditions. It is also dependent on the presence of contextual information in the historical ratings. Without historical contextual information, the user preference models can not be created, thus the re-ranking algorithm would not function.

Another limitation in our study regarding audio features is that the audio features that we used in our analysis and experiment are defined by Spotify and were retrieved from the Spotify API. Even though they are formed based on low-level audio analysis, we are still dependent on how Spotify defines and creates each audio feature. Any conclusion that includes any audio feature is under assumption that the method that Spotify uses to create them is representative and accurate. We expect our re-ranking algorithm to be working with audio features created by other providers as well. As long as the audio features are created in a representative and consistent way. However, more research is needed to validate this.

The experiment is carried out on two existing datasets, which means only offline evaluation has been carried out. An online evaluation next to these offline evaluations would further strengthen the results. Carrying out such an online experiment with a significant amount of participants is complex and requires proper preparation. Especially since users would need to actively use an app and most probably indicate what contextual conditions they are in each time they interact or rate a song.

The #NowPlaying-RS and InCarMusic datasets also have their own limitations. The InCarMusic dataset is very sparse. Both re-ranking results are therefore not too reliable, especially the personalized model based re-ranking. Many users did not rate any songs in multiple conditions, so for many conditions no personalized models or unreliable models were built. The #NowPlaying-RS dataset does not have sparsity problem, but it only contained one consistent contextual dimension, time of day. It provided valuable for evaluating the re-ranking algorithm for that dimension, but did not allow us to evaluate the re-ranking algorithm for other contextual dimensions. #NowPlaying-RS is also an implicit feedback dataset, which greatly limits the amount of initial recommender algorithms that could be used to create initial recommendation lists.

The re-ranking algorithm uses all contextual conditions within a selected dimension. It is limited to only one dimension at a time. Combining multiple dimensions quickly becomes increasingly complex with each additional dimension. That is why we left it out of scope for this research. Furthermore, all audio features and all conditions are given an equal weight when calculating the audio feature similarities. We have seen in the analysis preceding the experiment that some audio features are more descriptive than others. So this might not be the most optimal set-up. Also, one can imagine that some conditions have a stronger influence on the preference of users than others. Despite these limitations in our research, the results are still significant and promising.

7.3. FUTURE WORK

This section gives suggestions for potential directions of future work to gain more insights on this research topic. Each subsection describes a specific direction and how it builds on top of our work.

OFFLINE EVALUATION

In our research we evaluated the two re-ranking models using the InCarMusic and #NowPlaying-RS datasets. As can be seen from the results of the InCarMusic dataset, various contextual conditions yield different results. One possible direction is to further research how various conditions impacts the re-ranking performance. This can be done by running a comparable experiment on a large enough contextual music dataset that contains a variety of dimensions and conditions. Furthermore, it would be interesting to try to understand why some conditions have a stronger or more positive influence on the results than others. This can be done by carrying out a correlation analysis between these conditions and the audio features of the songs that have been classified to be suitable or unsuitable for example.

Unfortunately, to the extent of our knowledge, no such dataset of considerable size exists. Another idea is to create such a dataset by setting up and running a closely monitored user study. This would require massive effort and monitoring of users for multiple months. We believe that the insights that can be extracted from such a study will make it worthwhile, which brings us to the next direction of online evaluations.

ONLINE EVALUATION

As described in the previous subsection, the creation of a dataset with the right properties will also create a valuable opportunity to carry out online evaluations. This would mean that different versions of the re-ranking algorithm will be run for different users, while some users will only get the initial baseline recommendations. After some time, when there has been enough data generated by the users to be used by the re-ranking algorithm, user satisfaction can be measured. This can be done by using a user survey, for example. This will give the added benefit of feedback from actual users, instead of relying on the feedback from pure offline data evaluations. This suggestion is easier said than done, since user studies music recommendation approaches is rare, let alone contextual music recommendations [91]. Challenges involved in such a study include:

- The cold-start problem, where it is very hard or even impossible to make recommendations when a new user is using the system without having previously listened to any song. This is especially troublesome for the personalized mappings based re-ranking algorithm, since it requires significant historical personalized user interactions in different contextual conditions.
- To gather a group of users that is big enough while also making sure that diversity is not being nullified in the process. A user study with tens of people is doable and has been done before, like the user study from the InCarMusic dataset. But to gather hundreds of thousands or even millions of listening events, it would require a large number of active users that are willing to participate for a long time.
- Deciding on which contextual dimensions and conditions to include and gather from the users. For many dimensions, explicit permission of the user is needed to gather the data. For weather, for example, the application would need to know the location of the user. For other dimensions like activity and mood, it is even much harder to gather this data without the need of users to constantly input their current conditions. This is one of the main reasons why only small controlled user studies have been carried out regarding context-aware recommender systems.

COMBINING CONTEXTUAL DIMENSIONS

Another direction that can be looked into is how to deal with contextual situations that consists of a multitude of conditions from different dimensions. In our research, the re-ranking algorithm only has been run using conditions from a specific contextual dimension. This was done to keep keep a clear overview of how different dimensions impacted the results and to keep the system and algorithm simple. However, in practice, users often have multiple conditions that can impact their musical preference. Someone that is running in a happy mood will probably have different preferences than when running in a sad mood and again prefers different kind of music when happy or sad while studying.

Since the re-ranking algorithm we have designed is able to work with any dimension, it is easily compatible with situations as well. This is a matter of combining the audio feature representation of multiple unique contextual conditions. The biggest question and most interesting part here is how to combine these conditions so as to represent user preferences as accurately as possible. Does a linear combination work? What if one contextual condition has a stronger influence compared to another? To measure these relative influences, how they impact user preference and what effects they have on the re-ranked results is beyond the scope of our research. It is, however, an interesting topic to further research.

RE-RANKING ALGORITHM IMPROVEMENTS

Other steps that can be taken are related to improving the design and implementation of the re-ranking algorithm. Section 5.6 presents an overview of ideas and/or improvements that came to our mind when designing and implementing the algorithm, but we did not have enough time to test them. These include, but are not limited to:

- Giving audio features weights based on how representative they are to specific conditions and/or users.
- Selecting and including audio features based on whether they have a positive influence on the results.
- Averaging all similarities between the audio features of the song with the most positively rated songs' audio features instead of using the average of all historical ratings to calculate only 1 similarity.
- Using different similarity/distance measurements for the comparison between the audio features representation of the condition and given song.

A

InCarMusic Dataset Re-ranking Performance Results

Driving-style dimension based re-rankings

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.01082	0.04159	0.01082	0.04159	0.01082	0.04159	0.01082	0.04159	0.01082	0.04159
Re-ranked Global	0.01109	0.04216	0.01103	0.04218	0.01113	0.04141	0.01108	0.04087	0.01118	0.04093
Re-ranked Personal	0.01092	0.04174	0.01092	0.0417	0.01092	0.04033	0.01097	0.0402	0.01102	0.04036

Re-rank results for the **UserKNN** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list using the **driving-style** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.01179	0.064	0.01179	0.064	0.01179	0.064	0.01179	0.064	0.01179	0.064
Re-ranked Global	0.01201	0.06403	0.0121	0.064	0.01217	0.06286	0.01185	0.06204	0.01216	0.06046
Re-ranked Personal	0.01184	0.06382	0.01185	0.06321	0.01185	0.06167	0.01185	0.06119	0.01185	0.0613

Re-rank results for the **BiasedMF** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list using the **driving-style** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.01158	0.06748	0.01158	0.06748	0.01158	0.06748	0.01158	0.06748	0.01158	0.06748
Re-ranked Global	0.01217	0.06798	0.01222	0.0689	0.01201	0.06513	0.01206	0.06425	0.01179	0.06324
Re-ranked Personal	0.0119	0.06803	0.01196	0.06609	0.0118	0.065	0.0118	0.06399	0.01175	0.0636

Re-rank results for the **CAMF_CU** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list using the **driving-style** contextual dimension

Weather dimension based re-rankings

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.01082	0.04159	0.01082	0.04159	0.01082	0.04159	0.01082	0.04159	0.01082	0.04159
Re-ranked Global	0.01061	0.04117	0.01034	0.04047	0.01007	0.04115	0.00985	0.04029	0.01001	0.03998
Re-ranked Personal	0.01077	0.0416	0.01076	0.04126	0.01076	0.04166	0.0105	0.0414	0.01055	0.04117

Re-rank results for the **UserKNN** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list using the **weather** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.01179	0.064	0.01179	0.064	0.01179	0.064	0.01179	0.064	0.01179	0.064
Re-ranked Global	0.01158	0.06066	0.01142	0.06007	0.01136	0.05764	0.01098	0.05494	0.01066	0.05235
Re-ranked Personal	0.01168	0.0633	0.01173	0.06293	0.01147	0.06244	0.01142	0.06137	0.01115	0.05935

Re-rank results for the **BiasedMF** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list using the **weather** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.01158	0.06748	0.01158	0.06748	0.01158	0.06748	0.01158	0.06748	0.01158	0.06748
Re-ranked Global	0.01135	0.06572	0.01129	0.06407	0.01102	0.06006	0.01091	0.05894	0.01091	0.05857
Re-ranked Personal	0.01157	0.06674	0.01147	0.06572	0.01136	0.06524	0.01104	0.06436	0.01109	0.06443

Re-rank results for the **CAMF_CU** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list using the **weather** contextual dimension

Roadtype dimension based re-rankings

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.01082	0.04159	0.01082	0.04159	0.01082	0.04159	0.01082	0.04159	0.01082	0.04159
Re-ranked Global	0.01066	0.03985	0.01045	0.03998	0.01035	0.03903	0.01029	0.03764	0.01019	0.03713
Re-ranked Personal	0.01077	0.04124	0.01066	0.04128	0.01056	0.04141	0.01066	0.04267	0.01066	0.04235

Re-rank results for the **UserKNN** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list using the **roadtype** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.01179	0.064	0.01179	0.064	0.01179	0.064	0.01179	0.064	0.01179	0.064
Re-ranked Global	0.01174	0.06408	0.01158	0.06353	0.01121	0.06076	0.01093	0.05508	0.01078	0.05307
Re-ranked Personal	0.01168	0.06382	0.01174	0.06328	0.01158	0.06334	0.01136	0.06072	0.0112	0.05803

Re-rank results for the **BiasedMF** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list using the **roadtype** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.01158	0.06748	0.01158	0.06748	0.01158	0.06748	0.01158	0.06748	0.01158	0.06748
Re-ranked Global	0.01141	0.06647	0.0113	0.06423	0.01114	0.06072	0.01108	0.05897	0.01109	0.05786
Re-ranked Personal	0.01152	0.06696	0.01152	0.06676	0.01152	0.06565	0.01152	0.06442	0.01147	0.06493

Re-rank results for the **CAMF_CU** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list using the **roadtype** contextual dimension

B

#NowPlaying-RS Dataset Re-ranking Performance Results

Daytime dimension based re-rankings

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116
Re-ranked Global	0.02616	0.01116	0.02507	0.01072	0.02477	0.01034	0.02465	0.01008	0.02424	0.00985
Re-ranked Personal	0.02742	0.01229	0.02698	0.01226	0.02709	0.01191	0.02674	0.01176	0.02621	0.01131

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116
Re-ranked Global	0.02614	0.01116	0.02491	0.01067	0.02475	0.01036	0.02386	0.01003	0.0233	0.00979
Re-ranked Personal	0.02742	0.01229	0.02705	0.0123	0.02749	0.01198	0.02633	0.01187	0.02609	0.01178

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116
Re-ranked Global	0.02614	0.01116	0.02491	0.01067	0.0247	0.01036	0.02361	0.00997	0.02191	0.00896
Re-ranked Personal	0.02742	0.01229	0.02705	0.0123	0.02744	0.01196	0.02656	0.01196	0.02589	0.0115

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116
Re-ranked Global	0.02614	0.01116	0.02491	0.01067	0.0247	0.01036	0.02352	0.00994	0.01896	0.00729
Re-ranked Personal	0.02742	0.01229	0.02705	0.0123	0.02744	0.01196	0.02661	0.01196	0.02377	0.01065

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036
Re-ranked Global	0.0242	0.01005	0.0242	0.00976	0.0242	0.00949	0.0242	0.00928	0.0242	0.00918
Re-ranked Personal	0.0242	0.01092	0.0242	0.01075	0.0242	0.01052	0.0242	0.01047	0.0242	0.01011

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036
Re-ranked Global	0.02439	0.01001	0.02409	0.00955	0.02375	0.00925	0.02337	0.00896	0.0229	0.00867
Re-ranked Personal	0.02489	0.01102	0.0254	0.01103	0.02511	0.01086	0.02515	0.01084	0.02475	0.0107

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036
Re-ranked Global	0.02439	0.01001	0.02409	0.00955	0.02387	0.00923	0.02249	0.00868	0.02144	0.0077
Re-ranked Personal	0.02489	0.01102	0.02545	0.01106	0.02538	0.01097	0.02526	0.01096	0.02414	0.01031

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036
Re-ranked Global	0.02439	0.01001	0.02409	0.00955	0.02387	0.00923	0.0224	0.00865	0.0191	0.00619
Re-ranked Personal	0.02489	0.01102	0.02545	0.01106	0.02538	0.01097	0.02529	0.01098	0.02292	0.00989

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01036	NA	0.01036	NA	0.01036	NA	0.01036	NA	0.01036
Re-ranked Global	NA	0.01005	NA	0.00976	NA	0.00949	NA	0.00928	NA	0.00918
Re-ranked Personal	NA	0.01092	NA	0.01075	NA	0.01052	NA	0.01047	NA	0.01011

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01122	NA	0.01122	NA	0.01122	NA	0.01122	NA	0.01122
Re-ranked Global	NA	0.01088	NA	0.01052	NA	0.01032	NA	0.0101	NA	0.00987
Re-ranked Personal	NA	0.0118	NA	0.0117	NA	0.01158	NA	0.0116	NA	0.01151

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01276	NA	0.01276	NA	0.01276	NA	0.01276	NA	0.01276
Re-ranked Global	NA	0.01243	NA	0.01202	NA	0.01171	NA	0.01129	NA	0.01036
Re-ranked Personal	NA	0.01338	NA	0.01333	NA	0.01329	NA	0.01331	NA	0.01284

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.0147	NA	0.0147	NA	0.0147	NA	0.0147	NA	0.0147
Re-ranked Global	NA	0.01437	NA	0.01395	NA	0.01359	NA	0.01298	NA	0.01045
Re-ranked Personal	NA	0.01533	NA	0.01532	NA	0.01535	NA	0.01543	NA	0.01436

Re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125
Re-ranked Global	0.02545	0.01077	0.02437	0.01011	0.02351	0.00991	0.02338	0.0097	0.02331	0.00957
Re-ranked Personal	0.02633	0.01198	0.02611	0.01222	0.02584	0.01211	0.02552	0.01202	0.02529	0.01164

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125
Re-ranked Global	0.02545	0.01077	0.0243	0.01003	0.02317	0.00974	0.02268	0.00947	0.02189	0.00893
Re-ranked Personal	0.02633	0.01198	0.02601	0.01244	0.02666	0.01252	0.02612	0.01251	0.02566	0.01201

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125
Re-ranked Global	0.02545	0.01077	0.0243	0.01003	0.02321	0.00974	0.02261	0.00939	0.02038	0.00841
Re-ranked Personal	0.02633	0.01198	0.02601	0.01244	0.02664	0.01255	0.02654	0.01268	0.02515	0.0112

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125
Re-ranked Global	0.02545	0.01077	0.0243	0.01003	0.02321	0.00974	0.02261	0.00938	0.01782	0.00715
Re-ranked Personal	0.02633	0.01198	0.02601	0.01244	0.02664	0.01255	0.02656	0.01271	0.02374	0.01064

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01
Re-ranked Global	0.02455	0.00952	0.02455	0.00898	0.02455	0.00896	0.02455	0.00894	0.02455	0.00883
Re-ranked Personal	0.02455	0.01071	0.02455	0.01093	0.02455	0.0108	0.02455	0.01076	0.02455	0.01039

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01
Re-ranked Global	0.02441	0.00937	0.02369	0.00863	0.02369	0.00852	0.02312	0.00835	0.02294	0.00797
Re-ranked Personal	0.02503	0.0108	0.0249	0.01123	0.02512	0.01119	0.02482	0.01119	0.02465	0.01079

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01
Re-ranked Global	0.02441	0.00937	0.02367	0.00861	0.02334	0.00846	0.02206	0.00807	0.02085	0.00721
Re-ranked Personal	0.02503	0.0108	0.02493	0.01125	0.02509	0.0113	0.02513	0.01139	0.02433	0.01026

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01
Re-ranked Global	0.02441	0.00937	0.02367	0.00861	0.02334	0.00846	0.02199	0.00806	0.01879	0.00608
Re-ranked Personal	0.02503	0.0108	0.02493	0.01125	0.02509	0.0113	0.02511	0.01144	0.02307	0.0098

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01	NA	0.01	NA	0.01	NA	0.01	NA	0.01
Re-ranked Global	NA	0.00952	NA	0.00898	NA	0.00896	NA	0.00894	NA	0.00883
Re-ranked Personal	NA	0.01071	NA	0.01093	NA	0.0108	NA	0.01076	NA	0.01039

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01082	NA	0.01082	NA	0.01082	NA	0.01082	NA	0.01082
Re-ranked Global	NA	0.01029	NA	0.00962	NA	0.00949	NA	0.00943	NA	0.0091
Re-ranked Personal	NA	0.01158	NA	0.01203	NA	0.01195	NA	0.01195	NA	0.01159

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01234	NA	0.01234	NA	0.01234	NA	0.01234	NA	0.01234
Re-ranked Global	NA	0.0118	NA	0.01107	NA	0.01083	NA	0.01053	NA	0.00974
Re-ranked Personal	NA	0.01313	NA	0.01363	NA	0.01364	NA	0.01369	NA	0.01263

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01415	NA	0.01415	NA	0.01415	NA	0.01415	NA	0.01415
Re-ranked Global	NA	0.0136	NA	0.01287	NA	0.01258	NA	0.01209	NA	0.0101
Re-ranked Personal	NA	0.01495	NA	0.01548	NA	0.01556	NA	0.01573	NA	0.01407

Re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121
Re-ranked Global	0.0033	0.00115	0.00391	0.00098	0.00385	0.00118	0.00391	0.00156	0.00391	0.00159
Re-ranked Personal	0.00467	0.00276	0.0061	0.00342	0.0064	0.00475	0.00606	0.00502	0.00606	0.00516

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121
Re-ranked Global	0.0033	0.00115	0.00505	0.0012	0.00628	0.00245	0.00542	0.0029	0.00542	0.00291
Re-ranked Personal	0.00467	0.00276	0.00729	0.00402	0.00901	0.00648	0.00959	0.00783	0.00987	0.00906

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121
Re-ranked Global	0.0033	0.00115	0.00505	0.0012	0.00658	0.00255	0.00851	0.00354	0.00961	0.00396
Re-ranked Personal	0.00467	0.00276	0.00729	0.00402	0.0096	0.00721	0.01395	0.0129	0.01359	0.01458

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121
Re-ranked Global	0.0033	0.00115	0.00505	0.0012	0.00658	0.00255	0.00851	0.00354	0.00817	0.00256
Re-ranked Personal	0.00467	0.00276	0.00729	0.00402	0.0096	0.00721	0.01454	0.0131	0.01704	0.01276

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192
Re-ranked Global	0.00509	0.0019	0.00509	0.00187	0.00509	0.002	0.00509	0.00208	0.00509	0.00209
Re-ranked Personal	0.00509	0.00337	0.00509	0.00391	0.00509	0.00493	0.00509	0.00526	0.00509	0.00536

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192
Re-ranked Global	0.00575	0.00205	0.0053	0.00207	0.00541	0.00276	0.00551	0.00326	0.0053	0.00328
Re-ranked Personal	0.00689	0.00377	0.00733	0.00476	0.00665	0.00657	0.00686	0.00771	0.00686	0.00896

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192
Re-ranked Global	0.00575	0.00205	0.00551	0.00208	0.00666	0.003	0.00687	0.00377	0.00658	0.00376
Re-ranked Personal	0.00689	0.00377	0.008	0.00497	0.00905	0.00779	0.01032	0.01305	0.01019	0.01466

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192
Re-ranked Global	0.00575	0.00205	0.00551	0.00208	0.00666	0.003	0.00701	0.00363	0.00714	0.00257
Re-ranked Personal	0.00689	0.00377	0.008	0.00497	0.00917	0.00782	0.01047	0.01329	0.01258	0.01313

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.00192	NA	0.00192	NA	0.00192	NA	0.00192	NA	0.00192
Re-ranked Global	NA	0.0019	NA	0.00187	NA	0.002	NA	0.00208	NA	0.00209
Re-ranked Personal	NA	0.00337	NA	0.00391	NA	0.00493	NA	0.00526	NA	0.00536

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.00221	NA	0.00221	NA	0.00221	NA	0.00221	NA	0.00221
Re-ranked Global	NA	0.0023	NA	0.0024	NA	0.00299	NA	0.00349	NA	0.00356
Re-ranked Personal	NA	0.0039	NA	0.00478	NA	0.00658	NA	0.00769	NA	0.00893

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.0027	NA	0.0027	NA	0.0027	NA	0.0027	NA	0.0027
Re-ranked Global	NA	0.00281	NA	0.00296	NA	0.00368	NA	0.00437	NA	0.00459
Re-ranked Personal	NA	0.00449	NA	0.00564	NA	0.00825	NA	0.0133	NA	0.01487

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.00318	NA	0.00318	NA	0.00318	NA	0.00318	NA	0.00318
Re-ranked Global	NA	0.0033	NA	0.00344	NA	0.00417	NA	0.00483	NA	0.00383
Re-ranked Personal	NA	0.005	NA	0.00619	NA	0.00893	NA	0.01443	NA	0.01406

Re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

C

#NowPlaying-RS Dataset Opposite Re-ranking Performance Results

Daytime dimension based opposite re-rankings

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116
Re-ranked Global	0.0253	0.01069	0.02417	0.0101	0.02382	0.01002	0.02326	0.00986	0.02336	0.00968
Re-ranked Personal	0.02381	0.00965	0.02249	0.00878	0.02179	0.00847	0.02154	0.00841	0.02126	0.00821

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116
Re-ranked Global	0.02495	0.01058	0.02262	0.00938	0.02078	0.00869	0.02033	0.00844	0.01989	0.00793
Re-ranked Personal	0.02355	0.00957	0.0207	0.00806	0.01942	0.00748	0.0184	0.00722	0.01824	0.00674

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116
Re-ranked Global	0.02495	0.01058	0.02183	0.00912	0.01912	0.00779	0.01805	0.00707	0.01753	0.00646
Re-ranked Personal	0.02355	0.00957	0.02024	0.00796	0.01761	0.00678	0.01682	0.00617	0.01602	0.00568

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116	0.02684	0.0116
Re-ranked Global	0.02495	0.01058	0.02183	0.00912	0.01865	0.0077	0.01754	0.00661	0.01524	0.00536
Re-ranked Personal	0.02355	0.00957	0.02024	0.00796	0.01714	0.00668	0.01591	0.00559	0.01373	0.00442

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036
Re-ranked Global	0.0242	0.00967	0.0242	0.00945	0.0242	0.00941	0.0242	0.00941	0.0242	0.00924
Re-ranked Personal	0.0242	0.00905	0.0242	0.00835	0.0242	0.00825	0.0242	0.00828	0.0242	0.00817

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036
Re-ranked Global	0.02398	0.00962	0.02383	0.00898	0.02341	0.00863	0.02304	0.0084	0.0228	0.00795
Re-ranked Personal	0.02367	0.00891	0.02269	0.00762	0.02178	0.00711	0.02134	0.00694	0.02099	0.00646

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036
Re-ranked Global	0.02396	0.00962	0.0233	0.00874	0.02111	0.00767	0.01986	0.00693	0.01872	0.0063
Re-ranked Personal	0.02365	0.0089	0.02238	0.0075	0.01968	0.00631	0.01783	0.00554	0.01686	0.00501

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036	0.0242	0.01036
Re-ranked Global	0.02396	0.00962	0.02328	0.00874	0.02036	0.00751	0.01713	0.00613	0.01596	0.00519
Re-ranked Personal	0.02365	0.0089	0.02238	0.0075	0.01909	0.00617	0.01557	0.0048	0.01413	0.00386

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01036	NA	0.01036	NA	0.01036	NA	0.01036	NA	0.01036
Re-ranked Global	NA	0.00967	NA	0.00945	NA	0.00941	NA	0.00941	NA	0.00924
Re-ranked Personal	NA	0.00905	NA	0.00835	NA	0.00825	NA	0.00828	NA	0.00817

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01122	NA	0.01122	NA	0.01122	NA	0.01122	NA	0.01122
Re-ranked Global	NA	0.01059	NA	0.00998	NA	0.00967	NA	0.0095	NA	0.00909
Re-ranked Personal	NA	0.0099	NA	0.0088	NA	0.00844	NA	0.00834	NA	0.00792

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01276	NA	0.01276	NA	0.01276	NA	0.01276	NA	0.01276
Re-ranked Global	NA	0.01214	NA	0.01136	NA	0.01048	NA	0.00978	NA	0.0092
Re-ranked Personal	NA	0.01142	NA	0.01013	NA	0.00915	NA	0.00844	NA	0.00789

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.0147	NA	0.0147	NA	0.0147	NA	0.0147	NA	0.0147
Re-ranked Global	NA	0.01407	NA	0.01328	NA	0.01223	NA	0.01093	NA	0.00974
Re-ranked Personal	NA	0.01333	NA	0.01199	NA	0.01077	NA	0.00926	NA	0.00793

Opposite re-rank results for the **BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125
Re-ranked Global	0.02422	0.0105	0.02387	0.00985	0.02366	0.00965	0.02331	0.00933	0.02327	0.00926
Re-ranked Personal	0.02311	0.00961	0.02228	0.00874	0.02205	0.00853	0.02182	0.00836	0.02184	0.00832

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125
Re-ranked Global	0.02387	0.01036	0.02221	0.00907	0.02112	0.00844	0.02021	0.008	0.01979	0.00782
Re-ranked Personal	0.02294	0.00956	0.02091	0.00814	0.01944	0.00744	0.01846	0.00697	0.0181	0.0069

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125
Re-ranked Global	0.02387	0.01036	0.02178	0.00891	0.0194	0.00788	0.01808	0.00694	0.01767	0.00697
Re-ranked Personal	0.02294	0.00956	0.02063	0.00805	0.018	0.00696	0.01673	0.00601	0.01594	0.00572

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125	0.02508	0.01125
Re-ranked Global	0.02387	0.01036	0.02178	0.00891	0.01908	0.00772	0.01649	0.00617	0.01499	0.00544
Re-ranked Personal	0.02294	0.00956	0.02063	0.00805	0.01777	0.00689	0.01558	0.00548	0.01392	0.00442

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01
Re-ranked Global	0.02455	0.00959	0.02455	0.00916	0.02455	0.00904	0.02455	0.00882	0.02455	0.0088
Re-ranked Personal	0.02455	0.0089	0.02455	0.00828	0.02455	0.00818	0.02455	0.00807	0.02455	0.00807

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01
Re-ranked Global	0.0244	0.00952	0.02361	0.00861	0.02295	0.00821	0.02256	0.00798	0.02217	0.00783
Re-ranked Personal	0.02401	0.00873	0.02263	0.00747	0.02132	0.00689	0.02091	0.00657	0.02047	0.00652

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01
Re-ranked Global	0.02438	0.00951	0.02306	0.00846	0.02137	0.00772	0.02001	0.0069	0.01911	0.00695
Re-ranked Personal	0.024	0.00873	0.02214	0.00732	0.01978	0.00627	0.01801	0.00543	0.01678	0.0051

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01	0.02455	0.01
Re-ranked Global	0.02438	0.00951	0.02305	0.00845	0.02055	0.00746	0.01799	0.0061	0.01603	0.00531
Re-ranked Personal	0.024	0.00873	0.02214	0.00732	0.01903	0.0061	0.01598	0.00484	0.01424	0.00384

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01	NA	0.01	NA	0.01	NA	0.01	NA	0.01
Re-ranked Global	NA	0.00959	NA	0.00916	NA	0.00904	NA	0.00882	NA	0.0088
Re-ranked Personal	NA	0.0089	NA	0.00828	NA	0.00818	NA	0.00807	NA	0.00807

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01082	NA	0.01082	NA	0.01082	NA	0.01082	NA	0.01082
Re-ranked Global	NA	0.01038	NA	0.00954	NA	0.00922	NA	0.00902	NA	0.00892
Re-ranked Personal	NA	0.00964	NA	0.0086	NA	0.00818	NA	0.00787	NA	0.00785

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01234	NA	0.01234	NA	0.01234	NA	0.01234	NA	0.01234
Re-ranked Global	NA	0.01189	NA	0.01094	NA	0.01027	NA	0.00958	NA	0.00961
Re-ranked Personal	NA	0.01112	NA	0.00989	NA	0.00895	NA	0.00812	NA	0.00783

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.01415	NA	0.01415	NA	0.01415	NA	0.01415	NA	0.01415
Re-ranked Global	NA	0.01369	NA	0.01273	NA	0.01181	NA	0.01051	NA	0.00961
Re-ranked Personal	NA	0.0129	NA	0.01162	NA	0.01041	NA	0.00893	NA	0.0077

Opposite re-rank results for the **UserSplitting-BPR** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121
Re-ranked Global	0.00428	0.00271	0.00541	0.00374	0.00541	0.00286	0.00513	0.00262	0.00513	0.00283
Re-ranked Personal	0.00257	0.00102	0.00285	0.00107	0.00315	0.00101	0.00339	0.00084	0.00339	0.00084

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121
Re-ranked Global	0.00424	0.00277	0.00473	0.00384	0.00448	0.00274	0.00449	0.00257	0.0053	0.00297
Re-ranked Personal	0.00309	0.00111	0.00365	0.00138	0.00309	0.00101	0.00367	0.00088	0.00367	0.00085

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121
Re-ranked Global	0.00424	0.00277	0.00478	0.00389	0.00506	0.00316	0.00536	0.00294	0.00479	0.00279
Re-ranked Personal	0.00309	0.00111	0.00311	0.00133	0.00309	0.00099	0.0031	0.00081	0.00194	0.00056

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10	P@10	MAP@10
Initial	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121	0.00419	0.00121
Re-ranked Global	0.00424	0.00277	0.00478	0.00389	0.0053	0.00324	0.00533	0.00305	0.0054	0.00532
Re-ranked Personal	0.00309	0.00111	0.00311	0.00133	0.00333	0.00107	0.00328	0.00082	0.0031	0.0021

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 10** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192
Re-ranked Global	0.00509	0.00315	0.00509	0.00402	0.00509	0.00321	0.00509	0.00311	0.00509	0.00334
Re-ranked Personal	0.00509	0.00159	0.00509	0.00161	0.00509	0.00159	0.00509	0.00149	0.00509	0.00148

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192
Re-ranked Global	0.00509	0.00305	0.00481	0.00382	0.00526	0.00299	0.00516	0.00276	0.00514	0.00301
Re-ranked Personal	0.00456	0.00133	0.00381	0.00098	0.0039	0.00076	0.00368	0.00066	0.00358	0.00063

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192
Re-ranked Global	0.0052	0.00307	0.0051	0.00406	0.00546	0.00343	0.00496	0.00317	0.00485	0.00324
Re-ranked Personal	0.00456	0.00133	0.00384	0.00095	0.00331	0.00071	0.00304	0.0006	0.00316	0.00056

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25	P@25	MAP@25
Initial	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192	0.00509	0.00192
Re-ranked Global	0.0052	0.00307	0.0051	0.00406	0.00511	0.00331	0.00535	0.00332	0.00543	0.0059
Re-ranked Personal	0.00456	0.00133	0.00384	0.00095	0.0034	0.00073	0.00373	0.00067	0.00361	0.00204

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top 25** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

List size	25									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.00192	NA	0.00192	NA	0.00192	NA	0.00192	NA	0.00192
Re-ranked Global	NA	0.00315	NA	0.00402	NA	0.00321	NA	0.00311	NA	0.00334
Re-ranked Personal	NA	0.00159	NA	0.00161	NA	0.00159	NA	0.00149	NA	0.00148

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **25 songs** using the **daytime** contextual dimension

List size	50									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.00221	NA	0.00221	NA	0.00221	NA	0.00221	NA	0.00221
Re-ranked Global	NA	0.00329	NA	0.00406	NA	0.00322	NA	0.00315	NA	0.00337
Re-ranked Personal	NA	0.00167	NA	0.00145	NA	0.00131	NA	0.00121	NA	0.00119

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **50 songs** using the **daytime** contextual dimension

List size	100									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.0027	NA	0.0027	NA	0.0027	NA	0.0027	NA	0.0027
Re-ranked Global	NA	0.00376	NA	0.00466	NA	0.00402	NA	0.00387	NA	0.00388
Re-ranked Personal	NA	0.00206	NA	0.00164	NA	0.0014	NA	0.00129	NA	0.00126

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **100 songs** using the **daytime** contextual dimension

List size	200									
	$\lambda = 0.2$		$\lambda = 0.4$		$\lambda = 0.6$		$\lambda = 0.8$		$\lambda = 1.0$	
	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all	P@all	MAP@all
Initial	NA	0.00318	NA	0.00318	NA	0.00318	NA	0.00318	NA	0.00318
Re-ranked Global	NA	0.00424	NA	0.00514	NA	0.00452	NA	0.00454	NA	0.00706
Re-ranked Personal	NA	0.00252	NA	0.00206	NA	0.00171	NA	0.00157	NA	0.00298

Opposite re-rank results for the **CAMF-ICS** initial recommendation for both global and personal model, evaluated over the **top all** songs in the recommendation list consisting of **200 songs** using the **daytime** contextual dimension

BIBLIOGRAPHY

[1] Y. Zheng, *A user's guide to carskit,* arXiv preprint arXiv:1511.03780 (2015).

[2] D. Hauger, M. Schedl, A. Košir, and M. Tkalčič, *The million musical tweet dataset: What we can learn from microblogs,* (International Society for Music Information Retrieval, 2013).

[3] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer, *The yahoo! music dataset and kdd-cup'11,* in *Proceedings of the 2011 International Conference on KDD Cup 2011-Volume 18* (JMLR. org, 2011) pp. 3–18.

[4] M. Taifi, *Mrr vs map vs ndcg: Rank-aware evaluation metrics and when to use them,* (2019).

[5] O. Celma, *Music recommendation,* in *Music recommendation and discovery* (Springer, 2010) pp. 43–85.

[6] G. Adomavicius and A. Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,* IEEE Transactions on Knowledge & Data Engineering , 734 (2005).

[7] Z. Cheng and J. Shen, *Just-for-me: an adaptive personalization system for location-aware social music recommendation,* in *Proceedings of international conference on multimedia retrieval* (ACM, 2014) p. 185.

[8] M. Schedl, A. Flexer, and J. Urbano, *The neglected user in music information retrieval research,* Journal of Intelligent Information Systems **41**, 523 (2013).

[9] A. C. North, D. J. Hargreaves, and J. J. Hargreaves, *Uses of music in everyday life,* Music Perception: An Interdisciplinary Journal **22**, 41 (2004).

[10] G. Reynolds, D. Barry, T. Burke, and E. Coyle, *Interacting with large music collections: Towards the use of environmental metadata,* in *2008 IEEE International Conference on Multimedia and Expo* (IEEE, 2008) pp. 989–992.

[11] J.-Y. Kim and N. J. Belkin, *Categories of music description and search terms and phrases used by non-music experts.* in *ISMIR*, Vol. 2 (2002) pp. 209–214.

[12] M. Kaminskas and F. Ricci, *Contextual music information retrieval and recommendation: State of the art and challenges,* Computer Science Review **6**, 89 (2012).

[13] M. Braunhofer, M. Kaminskas, and F. Ricci, *Location-aware music recommendation,* International Journal of Multimedia Information Retrieval **2**, 31 (2013).

[14] K. Haruna, M. Akmar Ismail, S. Suhendroyono, D. Damiasih, A. C. Pierewan, H. Chiroma, and T. Herawan, *Context-aware recommender system: A review of recent developmental process and future research direction,* Applied Sciences **7**, 1211 (2017).

[15] N. M. Villegas, C. Sánchez, J. Díaz-Cely, and G. Tamura, *Characterizing context-aware recommender systems: A systematic literature review,* Knowledge-Based Systems **140**, 173 (2018).

[16] Y. Zheng, *Context-aware mobile recommendation by a novel post-filtering approach,* in *The Thirty-First International Flairs Conference* (2018).

[17] D. Liang, M. Zhan, and D. P. Ellis, *Content-aware collaborative music recommendation using pre-trained neural networks.* in *ISMIR* (2015) pp. 295–301.

[18] D. Wang, S. Deng, and G. Xu, *Sequence-based context-aware music recommendation,* Information Retrieval Journal **21**, 230 (2018).

[19] Q. Lin, Y. Niu, Y. Zhu, H. Lu, K. Z. Mushonga, and Z. Niu, *Heterogeneous knowledge-based attentive neural networks for short-term music recommendations,* IEEE Access **6**, 58990 (2018).

[20] S. Zhang, L. Yao, A. Sun, and Y. Tay, *Deep learning based recommender system: A survey and new perspectives,* ACM Computing Surveys (CSUR) **52**, 1 (2019).

[21] X. Wang and Y. Wang, *Improving content-based and hybrid music recommendation using deep learning,* in *Proceedings of the 22nd ACM international conference on Multimedia* (2014) pp. 627–636.

[22] A. B. Melchiorre and M. Schedl, *Personality correlates of music audio preferences for modelling music listeners,* in *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '20 (Association for Computing Machinery, New York, NY, USA, 2020) p. 313–317.

[23] F. Ricci, L. Rokach, and B. Shapira, *Introduction to recommender systems handbook,* in *Recommender systems handbook* (Springer, 2011) pp. 1–35.

[24] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, *Grouplens: an open architecture for collaborative filtering of netnews,* in *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (1994) pp. 175–186.

[25] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, *Item-based collaborative filtering recommendation algorithms,* in *Proceedings of the 10th international conference on World Wide Web* (2001) pp. 285–295.

[26] D. kumar Bokde, S. Girase, and D. Mukhopadhyay, *Role of matrix factorization model in collaborative filtering algorithm: A survey,* CoRR, abs/1503.07475 (2015).

[27] D. Bokde, S. Girase, and D. Mukhopadhyay, *Matrix factorization model in collaborative filtering algorithms: A survey,* Procedia Computer Science **49**, 136 (2015).

[28] Y. Koren, *Factorization meets the neighborhood: a multifaceted collaborative filtering model,* in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, 2008) pp. 426–434.

[29] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, *Large-scale matrix factorization with distributed stochastic gradient descent,* in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (2011) pp. 69–77.

[30] A. Paterek, *Improving regularized singular value decomposition for collaborative filtering,* in *Proceedings of KDD cup and workshop*, Vol. 2007 (2007) pp. 5–8.

[31] X. Xie, W. Tan, L. L. Fong, and Y. Liang, *Cumf_sgd: Parallelized stochastic gradient descent for matrix factorization on gpus,* in *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing* (2017) pp. 79–92.

[32] R. M. Bell and Y. Koren, *Scalable collaborative filtering with jointly derived neighborhood interpolation weights,* in *Seventh IEEE International Conference on Data Mining (ICDM 2007)* (IEEE, 2007) pp. 43–52.

[33] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, *Large-scale parallel collaborative filtering for the netflix prize,* in *International conference on algorithmic applications in management* (Springer, 2008) pp. 337–348.

[34] Y. Hu, Y. Koren, and C. Volinsky, *Collaborative filtering for implicit feedback datasets,* in *2008 Eighth IEEE International Conference on Data Mining* (Ieee, 2008) pp. 263–272.

[35] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, *Bpr: Bayesian personalized ranking from implicit feedback,* arXiv preprint arXiv:1205.2618 (2012).

[36] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, *Recommender system application developments: a survey,* Decision Support Systems **74**, 12 (2015).

[37] B. N. Schilit and M. M. Theimer, *Disseminating active mop infonncition to mobile hosts,* IEEE network (1994).

[38] N. S. Ryan, J. Pascoe, and D. R. Morse, *Enhanced reality fieldwork: the context-aware archaeological assistant,* in *Computer applications in archaeology* (Tempus Reparatum, 1998).

[39] P. J. Brown, *The stick-e document: a framework for creating context-aware applications,* Electronic Publishing-Chichester- **8**, 259 (1995).

[40] R. Hull, P. Neaves, J. Bedford-Roberts, *et al., Towards situated computing* (Hewlett Packard Laboratories, 1997).

[41] A. K. Dey, G. D. Abowd, and D. Salber, *A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,* Human–Computer Interaction **16**, 97 (2001).

[42] G. Adomavicius and A. Tuzhilin, *Context-aware recommender systems,* in *Recommender systems handbook* (Springer, 2011) pp. 217–253.

[43] Y. Koren and R. Bell, *Advances in collaborative filtering,* in *Recommender systems handbook* (Springer, 2015) pp. 77–118.

[44] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, *Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering,* in *Proceedings of the fourth ACM conference on Recommender systems* (ACM, 2010) pp. 79–86.

[45] K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura, *Context-aware svm for context-dependent information recommendation,* in *Proceedings of the 7th international Conference on Mobile Data Management* (IEEE Computer Society, 2006) p. 109.

[46] B. Twardowski, *Modelling contextual information in session-aware recommender systems with neural networks,* in *Proceedings of the 10th ACM Conference on Recommender Systems* (ACM, 2016) pp. 273–276.

[47] H. Wu, K. Yue, X. Liu, Y. Pei, and B. Li, *Context-aware recommendation via graph-based contextual modeling and postfiltering,* International Journal of Distributed Sensor Networks **11**, 613612 (2015).

[48] L. Baltrunas and F. Ricci, *Context-based splitting of item ratings in collaborative filtering,* in *Proceedings of the third ACM conference on Recommender systems* (ACM, 2009) pp. 245–248.

[49] L. Baltrunas and X. Amatriain, *Towards time-dependant recommendation based on implicit feedback,* in *Workshop on context-aware recommender systems (CARS'09)* (Citeseer, 2009) pp. 25–30.

[50] P. G. Campos, I. Fernández-Tobías, I. Cantador, and F. Díez, *Context-aware movie recommendations: an empirical comparison of pre-filtering, post-filtering and contextual modeling approaches,* in *International Conference on Electronic Commerce and Web Technologies* (Springer, 2013) pp. 137–149.

[51] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone, *Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems,* in *Proceedings of the third ACM conference on Recommender systems* (ACM, 2009) pp. 265–268.

[52] A. Taneja and A. Arora, *Clu-pof-a novel post filtering approach for efficient context aware recommendations,* Procedia computer science **122**, 834 (2017).

[53] L. Baltrunas, B. Ludwig, and F. Ricci, *Matrix factorization techniques for context aware recommendation,* in *Proceedings of the fifth ACM conference on Recommender systems* (2011) pp. 301–304.

[54] A. Said, E. W. De Luca, and S. Albayrak, *Inferring contextual user profiles-improving recommender performance,* in *Proceedings of the 3rd RecSys Workshop on Context-Aware Recommender Systems* (2011).

[55] S. Rho, B.-j. Han, and E. Hwang, *Svr-based music mood classification and context-based music recommendation,* in *Proceedings of the 17th ACM international conference on Multimedia* (ACM, 2009) pp. 713–716.

[56] I. Andjelkovic, D. Parra, and J. O'Donovan, *Moodplay: Interactive mood-based music discovery and recommendation,* in *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization* (ACM, 2016) pp. 275–279.

[57] M.-K. Shan, F.-F. Kuo, M.-F. Chiang, and S.-Y. Lee, *Emotion-based music recommendation by affinity discovery from film music,* Expert systems with applications **36**, 7666 (2009).

[58] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, *Automatic multimedia cross-modal correlation discovery,* in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, 2004) pp. 653–658.

[59] M. Kaminskas and F. Ricci, *Location-adapted music recommendation using tags,* in *International conference on user modeling, adaptation, and personalization* (Springer, 2011) pp. 183–194.

[60] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, *An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model,* IEEE Transactions on Audio, Speech, and Language Processing **16**, 435 (2008).

[61] M. Schedl and D. Schnitzer, *Location-aware music artist recommendation,* in *International conference on multimedia modeling* (Springer, 2014) pp. 205–213.

[62] A. Sen and M. Larson, *From sensors to songs: A learning-free novel music recommendation system using contextual sensor data.* in *LocalRec@ RecSys* (2015) pp. 40–43.

[63] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, and R. Schwaiger, *Incarmusic: Context-aware music recommendations in a car,* in *International Conference on Electronic Commerce and Web Technologies* (Springer, 2011) pp. 89–100.

[64] R. Dias, M. J. Fonseca, and R. Cunha, *A user-centered music recommendation approach for daily activities.* in *CBRecSys@ RecSys* (2014) pp. 26–33.

[65] X. Wang, D. Rosenblum, and Y. Wang, *Context-aware mobile music recommendation for daily activities,* in *Proceedings of the 20th ACM international conference on Multimedia* (ACM, 2012) pp. 99–108.

[66] Q. Li, S. H. Myaeng, and B. M. Kim, *A probabilistic music recommender considering user opinions and audio features,* Information processing & management **43**, 473 (2007).

[67] Q. Li, B. M. Kim, D. H. Guan, *et al.*, *A music recommender based on audio features,* in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (ACM, 2004) pp. 532–533.

[68] Q. Li and B. M. Kim, *Clustering approach for hybrid recommender system,* in *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)* (IEEE, 2003) pp. 33–38.

[69] D. Bogdanov, M. Haro, F. Fuhrmann, A. Xambó, E. Gómez, and P. Herrera, *Semantic audio content-based music recommendation and visualization based on user preference examples,* Information Processing & Management **49**, 13 (2013).

[70] F. Lu and N. Tintarev, *A diversity adjusting strategy with personality for music recommendation.* in *IntRS@ RecSys* (2018) pp. 7–14.

[71] M. Kaminskas, F. Ricci, and M. Schedl, *Location-aware music recommendation using auto-tagging and hybrid matching,* in *Proceedings of the 7th ACM conference on Recommender systems* (ACM, 2013) pp. 17–24.

[72] K. Seyerlehner, G. Widmer, M. Schedl, and P. Knees, *Automatic music tag classification based on block-level,* Proceedings of Sound and Music Computing 2010 (2010).

[73] C.-M. Chen, M.-F. Tsai, J.-Y. Liu, and Y.-H. Yang, *Using emotional context from article for contextual music recommendation,* in *Proceedings of the 21st ACM international conference on Multimedia* (ACM, 2013) pp. 649–652.

[74] B. Lamche, Y. Rödl, C. Hauptmann, and W. Wörndl, *Context-aware recommendations for mobile shopping,* in *LocalRec@ RecSys* (2015) pp. 21–27.

[75] G. Adomavicius and Y. Kwon, *Toward more diverse recommendations: Item re-ranking methods for recommender systems,* in *Workshop on Information Technologies and Systems* (Citeseer, 2009).

[76] B. Chidlovskii, N. S. Glance, and M. A. Grasso, *Collaborative re-ranking of search results,* in *Proc. AAAI-2000 Workshop on AI for Web Search* (2000).

[77] Spotify, *Company info,* (2019).

[78] P. Snickars, *More of the same–on spotify radio,* Culture Unbound: Journal of Current Cultural Research **9**, 184 (2017).

[79] G. Jawaheer, M. Szomszor, and P. Kostkova, *Comparison of implicit and explicit feedback from an online music recommendation service,* in *proceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems* (2010) pp. 47–51.

[80] A. Poddar, E. Zangerle, and Y.-H. Yang, *#nowplaying-rs: A new benchmark dataset for building context-aware music recommender systems,* in *Proceedings of the 15th Sound & Music Computing Conference* (Limassol, Cyprus, 2018) code at https://github.com/asmitapoddar/nowplaying-RS-Music-Reco-FM.

[81] M. Schedl, G. Breitschopf, and B. Ionescu, *Mobile music genius: Reggae at the beach, metal on a friday night?* in *Proceedings of International Conference on Multimedia Retrieval* (ACM, 2014) p. 507.

[82] E. Zangerle, M. Pichl, W. Gassler, and G. Specht, *# nowplaying music dataset: Extracting listening behavior from twitter,* in *Proceedings of the First International Workshop on Internet-Scale Multimedia Management* (ACM, 2014) pp. 21–26.

[83] F. Å. Nielsen, *A new anew: Evaluation of a word list for sentiment analysis in microblogs,* arXiv preprint arXiv:1103.2903 (2011).

[84] M. Hu and B. Liu, *Mining and summarizing customer reviews,* in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (2004) pp. 168–177.

[85] M. Deshpande and G. Karypis, *Item-based top-n recommendation algorithms,* ACM Transactions on Information Systems (TOIS) **22**, 143 (2004).

[86] N. Zheng and Q. Li, *A recommender system based on tag and time information for social tagging systems,* Expert Systems with Applications **38**, 4575 (2011).

[87] T.-Y. Liu *et al., Learning to rank for information retrieval,* Foundations and Trends® in Information Retrieval **3**, 225 (2009).

[88] G. Qian, S. Sural, Y. Gu, and S. Pramanik, *Similarity between euclidean and cosine angle distance for nearest neighbor queries,* in *Proceedings of the 2004 ACM symposium on Applied computing* (2004) pp. 1232–1237.

[89] T. Fushiki, *Estimation of prediction error by using k-fold cross-validation,* Statistics and Computing **21**, 137 (2011).

[90] Y. Zheng, B. Mobasher, and R. Burke, *Similarity-based context-aware recommendation,* in *International Conference on Web Information Systems Engineering* (Springer, 2015) pp. 431–447.

[91] D. Jannach, I. Kamehkhosh, and G. Bonnin, *Music recommendations,* (2018).