A COMBINING APPROACH TO COVER SONG IDENTIFICATION

Julien Osmalskyj

A thesis submitted for the degree of Doctor of Philosophy

Montefiore Institute
Department of Electrical Engineering and Computer Science
University of Liege

June 2017

Jean-Jacques	Embrechts	*Supervisor*
Marc	Van Droogenbroeck	*Co-Supervisor*
Simon	Dixon	*Jury Member*
Geoffroy	Peeters	*Jury Member*
Stephane	Dupont	*Jury Member*
Pierre	Geurts	*President*

A Combining Approach to Cover Song Identification

Julien Osmalskyj

Submitted for the University of Liege in fulfilment for the degree of Doctor of Philosophy (Computer Sciences)

Montefiore Institute
Department of Engineering, Electronics and Computer Science
University of Liege, Belgium

June 2017

Abstract

This thesis is concerned with the problem of determining whether two songs are different versions of each other. This problem is known as the problem of cover song identification, which is a challenging task, as different versions of the same song can differ in terms of pitch, tempo, voicing, instrumentation, structure, etc.

Our approach differs from existing methods, by considering as much information as possible to identify cover songs. More precisely, we consider audio features spanning multiple musical facets, such as the tempo, the duration, the harmonic progression, the musical structure, the relative evolution of timbre, among others. In order to do that, we evaluate several state-of-the-art systems on a common database, containing 12,856 songs, that is a subset of the Second Hand Song dataset. In addition to evaluating existing systems, we introduce our own methods, based on global features, and making use of supervised machine learning algorithms to build a similarity model.

For evaluating and comparing the performance of 10 cover song identification systems, we propose a new intuitive evaluation space, based on the notions of pruning and loss. Our evaluation space allows to represent the performance of the selected systems in a two dimensional space. We further demonstrate that it is compatible with standard metrics, such as the mean rank, the mean reciprocal rank and the mean average precision. Using our evaluation space, we present a comparative analysis of 10 systems. The results show that few systems are usable in a commercial system, as the most efficient is able to identify a match at the first position for 39% of the analyzed queries, which corresponds to 4,965 songs. In addition, we evaluate the systems when they are pushed to their limits, by analyzing how they perform when the audio signal is strongly degraded.

To improve the identification rate, we investigate ways of combining 10 systems. We evaluate rank aggregation methods, that aim at aggregating ordered lists of similarity results, to produce a new, better ordering of the database. We demonstrate that such methods produce improved results, especially for early pruning applications.

In addition to evaluating rank aggregation techniques, we propose to study combination through probabilistic rules. As the 10 selected systems do not all produce probabilities of similarity, we investigate calibration techniques to map scores to relevant posterior probability estimates. After the calibration process, we evaluate several probabilistic rules, such as the product, the sum, and the median rule. We further demonstrate that a subset of the 10 initial systems produces better performance than the full set, thus showing that some systems are not relevant to the final combination.

Applying a probabilistic product rule to a subset of systems significantly outperforms any individual systems, on the considered database. In terms of direct identification (top-1), we achieve an improvement of 10% (5,460 tracks identified), and in terms of mean rank, mean reciprocal rank and mean average precision, we respectively improve the performance by 40%, 9.5%, and 12.5%, with respect to the previous state-of-the-art performance.

We further implement our final combination in a practical application, named *DISCover*, giving the possibility for a user to select a query and listen to the produced list of results. While a cover is not systematically identified, the produced list of songs is often musically similar to the query.

Résumé

Cette thèse tente de déterminer automatiquement si deux chansons correspondent à deux versions d'un même titre. Ce problème est connu en anglais sous le titre de "cover song identification", soit, identification de différentes versions d'une même chanson. C'est un problème difficile, car deux versions d'un même titre peuvent différer en termes de tonalité, de tempo, de chant, d'instrumentation, de structure, etc.

L'approche que nous proposons dans cette thèse diffère de ce qui est traditionnellement appliqué. Nous proposons de tenir compte de toute l'information disponible dans le signal audio pour identifier des versions. Nous proposons donc d'extraire un ensemble de caractéristiques du signal audio, notamment le tempo, la durée, la progression harmonique, la structure musicale et l'évolution relative du timbre. Pour cela, nous implémentons et évaluons plusieurs systèmes existants dans la littérature, sur une base de données unique, contenant 12,856 titres. Cette base de donnée est un sous-ensemble d'une base de données existante, appelée le Second Hand Song dataset. Afin d'apporter plus d'information, nous introduisons nos propres méthodes de comparaison, basées sur des caractéristiques globales de la musique, en utilisant des algorithmes d'apprentissage automatique.

Nous évaluons et comparons ainsi 10 méthodes d'identification de versions, basées sur des caractéristiques musicales différentes. Pour l'évaluation, nous proposons un nouvel espace d'évaluation intuitive, basé sur les notions d'élagage (pruning) et de perte (loss). Cet espace d'évaluation permet de visualiser la performance d'un système dans un espace à deux dimensions. De plus, nous démontrons que cet espace est compatible avec des métriques standards, telles que le rang moyen, le rang moyen réciproque, et la moyenne des précisions moyennes. Nous utilisons cet espace pour comparer 10 méthodes. Les conclusions de l'analyse comparative démontrent que peu de systèmes sont pleinement utilisables à ce stade dans un système commercial. En effet, le système le plus performant à ce jour est capable d'immédiatement détecter d'autres versions pour 39% des chansons testées, soit 4,965 titres. Nous proposons de plus une analyse complémentaire qui considère des signaux audio fortement dégradés, afin d'évaluer les systèmes testés dans des conditions acoustiques difficiles.

Afin d'améliorer la performance, nous étudions divers moyens de combiner nos 10 méthodes, de manière à considérer l'information apportée par chaque système individuel. Nous expérimentons ainsi des méthodes d'aggrégation de rang, dont l'objectif est de combiner des listes de résultats afin de produire un résultat plus pertinent. Nous démontrons ainsi que ces techniques améliorent les performances.

Nous expérimentons également des méthodes de combinaisons probabilistiques. Étant donné que nos 10 systèmes ne produisent pas nécessairement des probabilités, nous étudions d'abord comment transformer des scores quelconques en probabilités à posteriori. Une telle transformation peut se faire par des techniques de calibrations. Après calibration, nous évaluons plusieurs règles de combinaisons probabilistiques, notamment la règle du produit, de la médiane et de la somme. Nous démontrons que la régle du produit génère les meilleurs résultats. De plus, nous démontrons qu'utiliser un sous-ensemble des 10 méthodes de base peut améliorer la performance.

Ainsi, en appliquant une règle de produit probabiliste sur un sous-ensemble de nos 10 méthodes, nous améliorons de manière significative la performance des systèmes de reconnaissances actuels. En termes d'identification immédiate (top-1), nous améliorons les performances de 10% (5,460 titres identifés), et en termes de rang moyen, rang moyen réciproque et moyenne des précision moyennes, nous améliorons les scores respectivement de 40%, 9.5% et 12.5%n par rapport au meilleur système actuel.

Nous implémentons notre combinaison finale dans une application pratique, appelée *DISCover*, qui permet à un utilisateur de sélectionner un titre et d'écouter la liste des chansons identifiées comme autres versions. Bien que le système ne reconnaisse pas toujours de version différente, on se rend compte que la plupart du temps, les chansons identifiées sont musicallement sembables aux chansons testées.

Contents

Acknowledgments

I would like to first thank my supervisors, Jean-Jacques Embrechts and Marc Van Droogenbroeck, for their unconditional support, their numerous valuable advice, and pertinent suggestions.

I further would like to thank everyone reading this thesis, in particular the members of the jury. As I had the pleasure to spend part of my research time in London, I would like to specially thank Simon Dixon, for being my supervisor for a short time at the Queen Mary University of London. Also, I thank all the friends and colleagues I met at the Center for Digital Music in London, in particular Peter, Maria, Giulio, Yading, Antonella, Pablo and everybody else in the lab.

A special thanks goes to Benjamin Laugraud, who spent a lot of time helping debugging and optimizing the code. I'm also grateful to Sébastien Piérard, who has always helped me, from the beginning of this thesis, with numerous ideas, advice and Matlab tricks.

Thanks also to Thomas Schmitz, who supported me throughout this thesis, despite my mood swings. Thanks to my good friends Vincent, Alexandra, Fred, Thomas, Ben and Arsène, with whom I had the pleasure to share so many enjoyable moments. Special thanks to Vincent for the numerous philosophical discussions. This thesis would not be finished without them.

I also would like to thank the members of the Montefiore Institute, especially the members of the acoustic department and the CEDIA, with whom I spent most of my time during my PhD. Special thanks to Angel Calderon Jimenez for his good vibes and interesting conversations.

Thanks also to my family, my sister and parents for their support and encouragements.

And finally, I thank so much Lucie for always supporting me and encouraging me during the last year of this thesis.

CHAPTER 1

Introduction

Music has been part of our lives for ages. It carries out emotions and feelings and has the power of making someone feel differently while listening to it. Whether in the car, at the office, in the bathroom, in a restaurant, or anywhere else, most of us enjoy listening to some music. With the widespread availability to the public of new technology, music listeners have been given access to a constantly growing catalog of music from all around the world. Throughout history, people have been listening to music through numerous media, from the phonogram, presented by Thomas Alva Edison in 1877, to vinyl discs, audio tapes, compact discs, digital files, and finally online music streaming services such as Spotify[1] or Deezer[2].

Nowadays, an increasing proportion of the population listens to music through these web services as it allows an easy way of accessing and managing music. With the widespread availability of mobile devices such as smartphones and tablets, desktop and mobile applications are designed to get access to the music collection, get a list of recommended tracks or to discover unknown artists. Such automated systems are becoming essential to manage a collection of songs growing every day. For example, the iTunes service reports a collection of more than 40 million tracks[3], while the streaming service Spotify reports a collection of more than 20 million tracks. Such web services are becoming more important not only for the consumers, but also for the artists and the music industry in general. According to Foster [40], the Recording Industry Association

[1]`https://www.spotify.com`
[2]`https://www.deezer.com`
[3]`htpp://apple.com/itunes/`

of America reports respective revenues of $1305 million and $859 million for download and streaming services for the first half of 2014. These download and streaming services account respectively for 41% and 27% of all revenues, compared to 28% of all revenues for physical formats (CD, Vinyl).

In order to constantly allow access to such huge collections of music, new computational methods are needed to instantly extract the requested information. Intelligent and fast algorithms are also necessary on the server side to keep the collection of songs organized and to update it rapidly, without losing access to the service. The research field of Music Information Retrieval (MIR) deals with the investigation of methods for managing such huge collections and giving access to them [24]. MIR investigates how to deal with musical characteristics of songs such as the pitch, the rhythm, the harmony, the timbre, the lyrics, the performance information, etc., as described by Downie [29]. MIR seeks at obtaining computable representations of these musical characteristics to ease the development of algorithms used to get access to the large music collections. As an example, one can consider a music streaming service such as Spotify. Such a system has been relying on textual metadata information to provide the users an easy way of selecting music. Such metadata include artist names, titles of the songs and other release information. To allow users to perform more specific searches, audio textual tags, encoding general and expressive annotations of the musical content [54], were included. They ultimately encode information about the musical characteristics MIR has to deal with. Using such music representation, streaming services allow a user to access a selection of recommended tracks that match their needs. One disadvantage of such a textual tags approach is that the metadata must be encoded manually by human operators. Furthermore, manual annotation requires a good knowledge of musical theory in order to produce robust tags. In addition, manually annotating musical data is an extremely time-consuming task. One objective of MIR is to automatically compute these tags. In order to do so, different sources of information can be considered: the sampled audio signal or the musical score representation. The former constitutes the field of content-based MIR, while the latter is referred to as symbolic MIR.

In this thesis, we focus on content-based music information retrieval. That part of MIR distinguishes tasks spanning audio fingerprinting, genre identification, mood classification, music retrieval applications, structural segmentation as well as versions identification, which constitutes the subject of this thesis. We focus in particular on the problem of cover song identification (CSI), which consists in identifying different versions of a song in a large collection of songs. This chapter describes and summarizes the work done in this thesis.

1.1 Content-based music analysis

Most tasks related to content-based MIR have to deal with the development of computational tools to analyze and extract audio features. The audio features are related to the specific goal of one application. For instance, in order to automatically organize a music collection by musical genre, one has to compute corresponding suitable features such as the so called MFCC coefficients and process them to extract the genre information. One can also imagine an application whose goal is to categorize the music by musical keys. In order to do this, appropriate features, such as chroma features must be extracted. MFCC and chroma features will be detailed later in this thesis.

Throughout the years, MIR thus led to the development of many of such applications. Among these applications, one can identify music recommendation systems, automatic playlist generation systems and music retrieval applications. The latter set of applications includes the task of cover song identification. Note that content-based MIR systems and applications are not only designed for the end consumer. It is also used by researchers and musicologists to extract patterns from music that help drawing conclusions about musical knowledge. For example, one could imagine using a musical chords recognition system to analyze common patterns in western music [74]. Content based MIR thus does not only allow consumers to gain access to novel recommendation systems that enable them to search and discover music, but also musicologists, artists and therapists [40].

This section describes the general principles of content-based MIR applications. Following that description, we analyze examples of such applications. In particular, we focus on music classification applications, which seek at performing a mapping between audio tracks and specific labels. Among these tasks, we distinguish in particular music retrieval tasks, and categorize them with respect to the concept of specificity and granularity, explained in this section.

1.1.1 Principles

Figure 1.1 provides a schematic overview of the principles of a content based MIR application. The first step consists in extracting features from the audio signal. Once the audio samples have been acquired by a content based system, they are processed using signal processing and statistical techniques in order to compute automatically musical characteristics. Using these features, audio tracks can be represented, for example by a sequence of time ordered features. These representations are next used to classify tracks. Track-wise distances can be computed to identify suitable tracks, with respect to the goal of an application. Alternatively, more complex

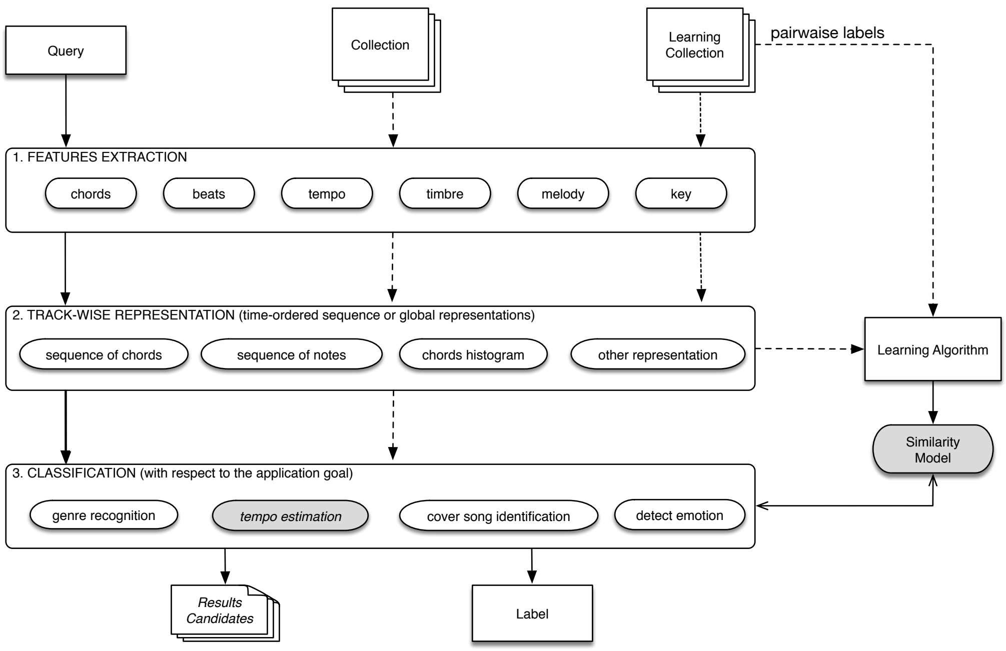


Figure 1.1: General pipeline of a content-based MIR application. The query and the reference audio collection must be processed similarly. The first step consists in extracting requested features. Then a track-wise representation, which can be global, or time-ordered is necessary. If a learning algorithm is used at some point, these representations must be sent to the learner in order to build a similarity model. The model can next be used in the final classification step to output a list of result candidates, or a single label, depending on the final application's goal. Alternatively, distance-based or matching algorithms can be used rather than machine learning techniques.

models can be used to classify the audio tracks, with respect to their computed representation. Such models can make use of machine learning algorithms, or others heuristics that allow to classify the music.

Most content based MIR applications internally rely on a concept of *music similarity*. For example, mood or genre identification systems seek at identifying songs that match the requested mood or genre. The system must therefore be able to identify tracks that are similar to each other, with respect to the requested musical genre. Likewise, for versions identification, we seek at identifying a set of songs that match a provided audio query, that is songs musically similar to the query track. The query in this case could be a short audio excerpt recorded by a user with a smartphone, for instance. This is the case in an audio identification system such as the popular Shazam application[4]. The system aims at identifying a song being recorded with a smartphone. Precisely, it returns the release information corresponding to the query. The

[4]`https://www.shazam.com` (accessed January 17, 2017)

system compares the query to a collection of songs and needs to find the track that is the most similar to the query. The concept of music similarity is therefore essential in content-based MIR applications. Among these applications, we focus in particular on music retrieval systems. Such systems follow the *query-by-example* paradigm to retrieve tracks in the collection, with respect to the query. Music retrieval systems span applications like audio identification, audio alignment, and versions identification.

1.1.2 Music retrieval systems

Music retrieval applications are intended to find music in large databases with respect to a specific criterion. The criterion can be a single parameter, such as the tempo, the duration of the songs, the musical key or at least one instrument that should be present in the returned polyphonic tracks. The criterion could also be more complex, for example, a short excerpt of musical content for an audio identification scenario. In that case, one has to record a few seconds of a song and send it to the application. The latter in turn processes the audio and returns the exact release information corresponding to the song. Instead of being a short fragment of music, the criterion could correspond to a full length song. For instance, let us consider the problem of versions identification. The system requires a candidate song as an input, which acts as the criterion of the retrieval system. The application processes the audio and somehow compares it to a collection of songs. It eventually returns a list of songs that should correspond to other versions of the query. In that case, the system needs a full length song for a query in order to analyze the entire sequence of audio data.

From these examples, we can derive two observations. First, audio retrieval applications return sets of results with different cardinalities. Some of them return an exact match while others return a list of similar tracks. In other terms, depending on the application, that is, the context, we do not expect the same cardinality results. Grosche et al. [54] identify that observation as the *specificity* of the retrieval application. According to them, "*the specificity of a retrieval system refers to the degree of similarity between the query and the database document to be retrieved*". For example, an audio identification system such as Shazam returns either the exact match, or nothing (cardinality is one or zero). On the other hand, low-specific systems return a list of matches that are similar to the query. For example, retrieving songs corresponding to the jazz musical genre requires a low-specificity as a lot of existing tracks can be classified in that category (cardinality ≥ 1). In that categorization, version identification can be classified as a *mid-specific* system: we do not require a single match, but a relatively reduced set of tracks that are musically similar to the query.

The second observation is that some applications request short audio fragments as an input, whereas others require full length songs as an input. Again, depending on the context, the requested information is different from one case to another. Grosche et al. [54] name that observation the *granularity* of the retrieval system. In other terms, systems requesting fragment of audio can be considered to have a low granularity, whereas systems considering full length songs have a high granularity. For example, fragment-level retrieval scenarios consider short excerpts of audio as a query. The goal is to retrieve parts of the songs of the music collection that are similar to the excerpt query. One example application with low granularity would be to find all the songs in a database that *contain* the query at some temporal moment in the songs. This corresponds to identifying songs that *quote* the query. As opposition to fragment level scenarios, document-level retrieval makes use of entire documents instead of fragment. In that case, the query is an entire track, and the retrieval system considers full length tracks in the search database. Versions identification fits in the second category and is considered to have a high granularity. As different versions can be different in terms of musical structure, complete tracks are necessary. One key difference between both scenarios (low granularity and high granularity) is that in the former case, one has to consider local similarity, while global similarity can be considered in the latter.

1.1.3 Version identification

In this thesis, we are interested in the problem of identifying different versions of a song, with respect to an audio query. In comparison to tasks such as audio identification (fingerprinting) or audio alignment, the task has a relatively mid-level specificity. The goal is to return a single track, or a set of tracks (cardinality ≥ 1) that are considered as different versions of the query. From a musical point of view, this corresponds to identifying a list of tracks that are considered similar to the query, while allowing a certain degree of flexibility. Whereas different versions of one song should refer to each other, they might be quite different in terms of audio content. Indeed, versions can differ from the query in terms of tempo, musical key, duration of the song, musical structure, lyrics, and other characteristics. It is therefore necessary to be able to derive a measure of similarity between cover songs that can deal with such musical differences.

Version identification incorporates different variants of the problem. The first one corresponds to the identification of remasters [23] of a recording, where a remaster only differs from the original song in terms of musical timbre and noise characteristics [40]. The second variant is the most popular one in MIR and corresponds to identifying *cover versions* of a song, where versions can differ from the original song in terms of tempo, structure, harmony, lyrics and other musical facets [92].

This thesis considers the second variant of the versions identification problem, named *Cover Song Identification* (CSI). We experiment with a novel approach to the task that makes use of multiple audio features and multiple retrieval algorithms. The next section addresses the research questions related to this thesis.

1.2 Research topics

Typical approaches to the problem of cover song identification are based on the extraction of mid-level audio features that usually represent the harmonic content of the music. Given an audio query, CSI systems aim at identifying other versions of the query by computing pairwise similarities between the query and the tracks in the audio collection. For instance, the matching tracks can be identified by comparing models of features. Usually, we can distinguish two categories of features to describe the music for CSI. The first one is a temporal description of the music. In other words, we have a time-ordered description of the music using some audio feature computed at regular time points in the audio. The second one follows a global description of the musical content. No temporal information is retained, and the music is described using a set of properties that characterize the music globally. Such description can be classified in the category of *bag-of-features* approaches. Time-ordered features usually require computationally expensive algorithms to compare audio tracks. On the other hand global descriptions allow much faster comparisons of tracks due to the low dimensionality of the music descriptions. According to results of the literature, global feature do not perform as well as time ordered features. However, they are more scalable and allow to compare a query to a large collection of songs.

While such systems based on time-ordered and global features have been used extensively in the past, we include new descriptions of the music based on simple informations such as the tempo, the duration of the music, the beats, etc. We thus evaluate several systems based on various features, not necessarily related to each other. We investigate how simple CSI systems based on rather simple audio descriptions perform. The reason is that if two systems are built upon different features, it should be possible to take advantage of both features to improve the performance. Thus, we are interested in the performance of existing systems on a rather large scale, and in the information brought by each system. We are also interested in designing simple CSI systems based on simple features in order to bring as much information as possible. Finally, we seek at identifying different ways of combining all these systems to quantify the performance of a global combined CSI system.

In this section, we address the main research questions that we seek at answering to in this thesis. For each of them, we present a summary of what has been done and refer to the remaining of this thesis for further details.

1.2.1 Quantifying the performance of CSI systems

From the perspective of content-based music retrieval systems, one of the fundamental question we consider in this thesis is how to accurately evaluate and quantify the performance of a CSI system. This question is of high importance as we need a uniform way of comparing CSI systems together, on a common collection of music.

In order to do that, it is mandatory to make use of an evaluation procedure that is able to quantify (1) the general performance of a system, not bound to a restricted set of databases, or (2) the performance on a specific dataset, used in a production environment, for example. To our knowledge, there is no standard database publicly available that would allow researchers to evaluate and quantify the performance of a CSI system. Traditionally, CSI systems are evaluated on individual small datasets, often private collections of digital files, containing at most a few hundreds songs (see Chapter 3). Few research works have been evaluated on a common database, as will be pointed out in Chapter 3. While scores such as the mean reciprocal rank, the mean rank, and the mean average precision are commonly used to quantify the performance of a system, we demonstrate that such metrics correspond to specific areas in an evaluation space. We show that plotting a performance curve allows to interpret the performance of a CSI system more accurately.

We address the problem of the evaluation of a CSI system by providing an evaluation based on the Second Hand Song Dataset (SHS[5]) containing approximately $18,000$ songs. While the dataset does not allow access to audio data, it provides a collection of features pre-computed by the EchoNest analyzer[6]. We thus show that several state-of-the-art CSI systems, originally evaluated on private or small collections, can be evaluated on a bigger dataset, provided by a third party. Using the SHS dataset, we provide an evaluation of the performance of 10 CSI systems in a novel evaluation space and in terms of standard metrics and evaluation spaces (see Chapter 5).

1.2.2 Interest of global features

Traditional features used for CSI principally incorporate descriptors derived from the harmonicity of the songs. The harmonicity can be musically described by musical chords. Chords constitute an interesting mid-level description of the music as they are relatively independent of the instrumentation and rhythm of the music. Musical chords are usually described in a computer by chroma features, that encode the harmony content of the music in a series of 12-dimensional vectors [45]. Music is thus described as a temporal sequence of chroma features. Existing

[5]`http://labrosa.ee.columbia.edu/millionsong/secondhand` (accessed January 17, 2017)
[6]`http://the.echonest.com/` (accessed January 17, 2017)

methods have been using chroma features extensively. Techniques were developed to align chromas on the beat of the music, rendering the track representation tempo-invariant. Likewise, chroma features allow a track to be rendered pitch-invariant using various transposition techniques. Such interesting properties are the reason why chroma features have been used as the main music representation for CSI.

Other representations have been suggested in the literature, based on timbral sequences [105] and melody sequences [85]. Most of them are derived from chroma features or other spectral representations of the audio signal. Such features make sense from a musical point of view, and have been proven to produce a decent performance for CSI. However, to our knowledge, few works have considered the use of simpler features, yet showing a musical meaning. Indeed, features such as the tempo, the beats, the average chroma vector, or even spectral features close to the signal have not been investigated for CSI. While it seems logical that such features would not perform well in a CSI setup, they are likely to contribute to the overall performance, when combined with existing harmonic and timbral features. We therefore investigate how some of these features perform for cover song recognition (see Chapter 5). In particular, we measure musical similarity using classifiers based on such features, using machine learning algorithms to learn the similarity function.

1.2.3 Robustness of CSI systems to audio degradations

While several CSI systems have been designed and demonstrated to produce results, none of them was fully experimented in real life, noisy environments. Most of the time, systems are evaluated in a controlled research environment, with ideal conditions. As we measure the performance of 10 CSI systems, in terms of standard metrics and performance space, one interesting question is to understand how these systems perform in various noisy conditions. Are the selected audio features robust against acoustic perturbations in the audio signal? Answering this question would allow to select the right systems and audio features depending on the use context of a commercial application. In this thesis, we address this question (see Chapter 6) and select a subset of CSI system to be evaluated in various environmental noise conditions.

1.2.4 Investigating ways of combining systems for an improved performance

Having evaluated the performance of 10 systems under various conditions, the key topic studied in this thesis is how to take advantage of all systems to build a composite CSI system that takes advantage of the individual systems. This question is essential, and has not been answered yet, to our knowledge. Although several existing works consider mixing information at the feature level, no work has studied how to mix the output of independent systems to build a robust composite system. This topic of high interest raises many questions: Do all systems return the same kind of output? Should the outputs be normalized? If not, how to combine outputs on different scale while maintaining an interpretable performance? Answering these questions leads to the topic of classifiers combination, which surprisingly has not been studied so much in the literature. In this thesis, we answer these questions (see Chapters 7 and 8), showing that methods for combining classifiers, or ordered list of results allow to improve the performance of CSI systems.

1.3 Contributions

This section summarizes the novel contributions of this thesis.

1. **Methods for estimating cover song similarity based on global features.** This thesis introduces a set of CSI systems based on low-dimensional and global features. While most systems make use of complex features extraction and matching algorithms, we show in this thesis that systems based on features as simple as the tempo, the duration, the beats, and the average chroma vectors are useful. We show that although such systems perform rather poorly when considered individually, they contribute to an improved performance when combined with other systems. The provided systems are based on machine learning algorithm, specifically random forests, which produce cover song similarity models. Evaluating such simple systems, we demonstrate that any feature is useful, as long as it can be combined with other systems, hence as long at it shows some degree of independency with respect to the other considered systems.

2. **Theoretical and practical evaluation framework for estimating and comparing the performance of CSI systems.** We provide a new framework for evaluating the performance of CSI systems. We introduce a relaxation of the main problem by considering that at least one match to the query should be identified for the system to be successful. Based on that definition, we define a new evaluation space based on the concept of pruning and loss. We demonstrate that our space is more useful and more interpretable than other spaces such as the Receiver Operating Characteristics (ROC), or the Precision-Recall space. In addition, it allows to compare the performance of systems evaluated on different databases, as long as the characteristics of the databases are known. We show that this framework is also compatible for evaluating systems on a specific database, and we demonstrate its compatibility with some standard metrics such as the mean rank or the top-k statistics. In addition, we demonstrate that our framework allows to predict the performance of a system on any database, without having to run a time consuming experimental evaluation.

3. **Experimental evaluation and comparison of 10 CSI systems on a large database.** We provide an evaluation of ten different CSI systems on a large database containing approximately $13,000$ songs. Some of the systems are implementations of state-of-the-art methods from the literature, and others are our own methods based on global features. We provide an evaluation based on a subset of the Second Hand Dataset, which is the largest dataset available for cover song identification research. As the features are pre-computed and no audio data is available, our evaluation had to deal with uncontrolled features, that is, features as they are provided with no possibility of adjusting them to match the algorithms. Therefore, our setup is quite realistic as a commercial application would likely have to deal with provided databases. We provide a comparison of the methods in terms of our evaluation space, and in terms of standard metrics such as the mean rank, the mean reciprocal rank and the mean average precision. To the best of our knowledge, we are the first to provide such a comparison on a common large database.

4. **Analysis of the robustness of multiple systems to acoustic audio degradations.** Because a hypothetical commercial CSI system would be used in potential noisy environments, we provide an evaluation of the robustness of several systems under multiple acoustic degradations conditions. In particular, we alter the audio signals with increasing surrounding noise levels, and multiple levels of quadratic distortion. Furthermore, various degradations, such as added reverberation and convolution with a smartphone speaker, are applied to the systems. We demonstrate that most studied systems show a decent degree of robustness against most of applied audio degradations.

5. **Evaluation of multiple combination methods to aggregate the outputs of multiple systems.** Using ten CSI systems, we provide an evaluation of multiple combination algorithms to consider the features and similarity estimation algorithms used by each individual system. In particular, we investigate parameter free methods, based on the positions of the tracks in the lists of results returned by the systems. Such methods are named *rank aggregation* methods and do not require any score normalization. We demonstrate that rank aggregation methods allow to achieve an increased performance for cover song identification. Following rank aggregation methods, we evaluate probabilistic combining rules to build an improved composite system. Because all systems do not return probabilities, we investigate how to map scores to probabilities through calibration techniques. We provide an evaluation of three probabilistic rules and demonstrate that the product rule in particular provides an huge improvement in terms of identification rate. Finally, we show that removing some systems from the final combination further improves the overall performance of the combined system. We thus demonstrate that, for some combinations, some systems do not contribute to the combined performance

and are thus not relevant for cover song identification. Combining a subset of systems using probabilistic rules allows to create a composite system that outperforms current state-of-the-art performance.

6. **Design of a prototype application for identifying cover songs.** Having studied several ways of improving the performance of CSI system, we implement our final combination of 10 systems in a prototype web application. We demonstrate that our combined technique allows to identify a lot of queries taken as an input. The application is named *DisCover* and is available publicly online[7].

1.4 Publications

This thesis has led to the following peer-reviewed publications:

- **J. Osmalskyj**, S. Piérard, M. Van Droogenbroeck, J.J. Embrechts. Neural Networks for Musical Chords Recognition. *Journées d'Informatique Musicale (JIM)*, pp 39-46, 2012.
- **J. Osmalskyj**, S. Piérard, M. Van Droogenbroeck, J.J. Embrechts. Efficient Database Pruning for Large-Scale Cover Song Recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp 714-718, 2013.
- **J. Osmalskyj**, M. Van Droogenbroeck, J.J. Embrechts. Performances of Low-Level Audio Classifiers for Large-Scale Music Similarity. *IEEE International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp 91-94, 2014.
- **J. Osmalskyj**, P. Foster, S. Dixon, J.J. Embrechts. Combining Features for Cover Song Identification. *International Society for Music Information Retrieval Conference (ISMIR)*, pp 462-468, 2015.
- **J. Osmalskyj**, M. Van Droogenbroeck, J.J. Embrechts. Enhancing Cover Song Identification with Hierarchical Rank Aggregation. *International Society for Music Information Retrieval Conference (ISMIR),* pp 136-142, 2016.
- **J. Osmalskyj**, J.J. Embrechts. Effects of Acoustic Degradation on Cover Song Identification Systems. *International Congress on Acoustics (ICA)*, 2016.

In these publications, J. Osmalskyj realized all scientific work, from questions formulation, code implementation, design of the experiments to the manuscript writing. The paper written with P. Foster and S. Dixon is the result of a research collaboration with the Queen Mary University

[7]`http://auralie2.montefiore.ulg.ac.be/`

of London (QMUL), precisely with the Center for Digital Music (C4DM) laboratory, under the direction of S. Dixon. The collaboration was part of a research visit of 6 months at the QMUL. The main supervisor of this thesis is J.J. Embrechts, and the co-supervisor is M. Van Droogenbroeck.

1.5 Outline

The remainder of this thesis is organized as follows. In Chapter 2, we begin by explaining the general principles of cover song identification. We explain the different steps involved in a CSI system and describe how each steps has been studied in the literature. Following the general principles, we briefly introduce our way of dealing with the problem and give some insights for the remainder of the thesis.

In Chapter 3, we formally define the problem of CSI, and we introduce a relaxation to the problem. We formulate, mathematically, how a CSI system works, and we introduce a new evaluation space, based on the notions of pruning and loss, for evaluating CSI systems.

Chapter 4 is an extension of Chapter 3: it brings a slight modification of the evaluation space introduced in the previous chapter so that it becomes compatible with some standard evaluation metrics, as used in the literature. We further demonstrate how to predict the performance of a system on another database, without ever running a physical evaluation, thus saving computational time.

Having explained how to evaluate the performance of CSI systems, we provide in Chapter 5 an evaluation of 10 CSI systems based on various audio features and matching algorithms. For each of them, we explain the algorithm and the corresponding performance, on a rather large database. In order to understand which features and algorithms are robust to acoustic degradations, we further perform an analysis of a subset of these 10 systems in various noisy conditions. The latter study is provided in Chapter 6.

These six first chapters constitute the first part of this thesis, focused on the principles and performance of cover song identification. In the second part, we introduce combination and analyze several ways of mixing the output of individual CSI system in order to build a robust composite system that outperforms current existing systems. We begin in Chapter 7 with a brief introduction to combination. In particular, we review the small amount of work in the field of combination for cover song identification.

Chapter 7 introduces rank aggregation techniques. Such techniques are interesting to combine ordered lists of results provided by individual systems. They show the advantage of being parameter free and they do not require any parameters tuning. We explain how such techniques work and how they perform with 10 systems. In Chapter 8, we consider another combination technique based on probabilities. We review existing methods for combining probabilities. We demonstrate how to convert classifier's scores to probability estimates, and experiment with multiple combination rules. We conclude the chapter by showing that in the final combination, some methods are not necessary to improve the performance. Thus, using a subset of methods is sometimes more interesting than using all initial systems. Finally, Chapter 9 concludes this work.

CHAPTER 2

The Principles of Cover Song Identification

The goal of a CSI system is to automatically retrieve different versions of an audio track (the query), by comparing it to a collection of reference musical tracks. The query is usually issued to the system as an audio recording, for instance recorded by a smartphone. Considering the query, the CSI system runs an identification algorithm that computes a similarity score for each track of the reference collection. The system should then produce a list of the tracks considered the most similar to the query.

Identifying cover songs is not an easy task, because there is no clear definition of what a cover song should be. In some cases, a cover version of a song is musically similar to the query, for example in terms of melody. In some other cases, a cover version could be an electronic remix of a classic rock song. In that case, the similarity between that version and the original track is much more difficult to identify, even for a human listener. However, as pointed out by Foster [41] and Serra [87], CSI systems commonly assume that the sequential pitch is preserved among the different versions of the same piece of music. The sequential pitch can be represented for instance by the melody of the song, or the list of chords of the song. However, cover versions of a song are often different from the original track in terms of pitch, tempo, musical structure, instrumentation, etc., which makes cover song identification a difficult task.

In order to identify cover songs in a large database, one can easily imagine that the raw audio signal is not suitable to identify similar tracks. Somehow, a representation of the music is necessary to compare the query to the collection of audio tracks. That representation should describe all the tracks in the database, as well as the query, such that both are comparable. This is typically done in a step of *audio features extraction*. Audio features encode the characteristics of the music such that they can be processed by a computer. Audio features are extracted from the audio signal by computational methods, and correspond, for example, to a list of musical chords, or a list of notes, expressed as fundamental frequencies. Once they are computed, the features should be stored in data structures that can be accessed rapidly at the identification stage. Many audio features have been studied in the litterature, and this chapter reviews some of them that are used for CSI.

Following the features extraction process, we need to compare the query to the tracks of the music collection: this is the *retrieval* stage. Most CSI systems are based on the principle of pairwise comparisons between the query and each track of the reference collection. A score is calculated for each pair <*query, reference track*> so that the collection can be sorted, with respect to the similarity with the query. This allows to identify the most similar tracks at the top of the returned list.

Depending on the selected features, different comparison algorithms have been investigated in the litterature. Such algorithms can make use of global features, that do not take into account the temporal evolution of the music (*eg* the musical key), or features sampled over time in the audio signal (*eg* the list of chords from the begining of the song to the end of it). For global features, similarity scores can be computed as simple distances such as the Euclidean distance, or the cosine similarity [96]. For time-ordered features, multiple strategies have been investigated, and will be described later in this chapter.

In the following, we begin with a description of common audio features used for the task of CSI. Then, a review of the CSI litterature is provided to have a global overview of the research field. In particular, we split our litterature review in a part specific to audio features, followed by a part on identification algorithms.

2.1 Stages of a cover song identification system

As pointed out in the introduction of this chapter, CSI systems involve different steps. Figure 2.1 shows the different elements that are involved in a CSI system: (1) the features extraction step, (2) a features post-processing step and (3) a similarity estimation step. The features extraction step extracts audio features from the audio signals. Such features are usually computed based on the power spectrum of the songs and correspond to the base representation of the songs. Following the features extraction step, post-processing algorithms are applied to the features to build a higher-level description of the audio tracks. Such a higher-level description is then encoded in a higher-level audio features. In addition to building a track-wise representation of the music, the post-processing step generally involves rendering the raw features tempo,

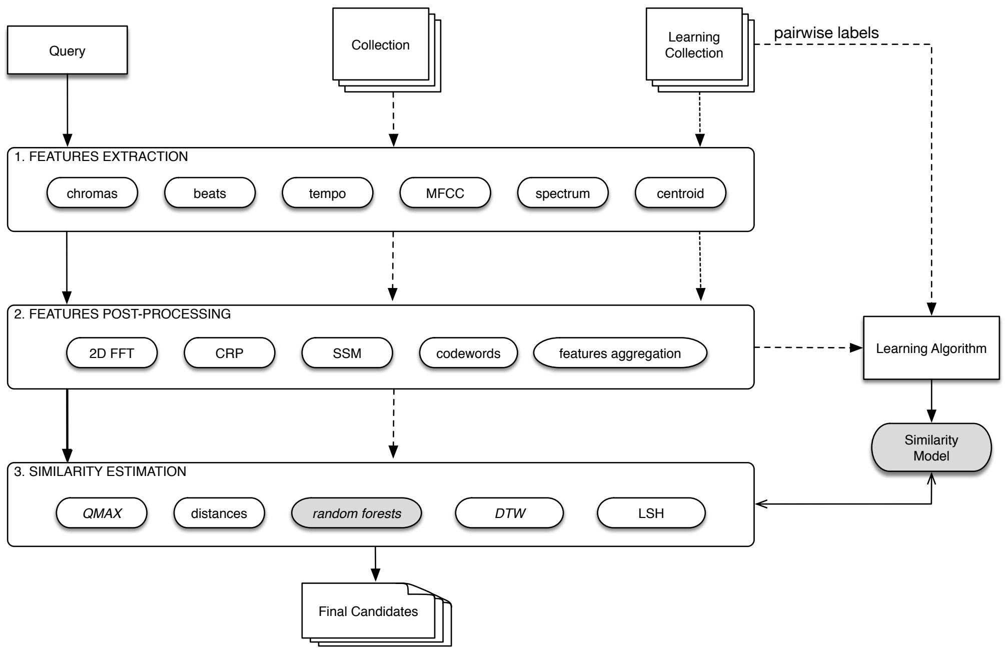


Figure 2.1: General pipline for a cover song identification system. Features are first extracted from the query, the audio collection and the learning collection if needed. They are next post-processed by a specific algorithm (examples in the boxes) in order to build a track-wise, alternative representation of the music. Finally, a similarity algorithm is used, potentially with a learned similarity model, to output a list of matching candidates.

structure and key-invariant, as these musical characteristics can change from one version to another. Finally, once a suitable features is available for the query, and for the collection of songs, one needs a similarity estimation algorithm to score the songs of the collection, with respect to the similarity with the query. Then, a set of output candidates is returned. The similarity estimation step can be as simple as a Euclidean distance computation, or more complex. For instance, some systems make use of machine learning algorithms to build a similarity model to rank tracks, with respect to an audio query. In that case, part of the reference collection, or an alternative collection can be used as a learning set to build a supervised similarity model. This model building process is illustrated in Figure 2.1 by the *random forests* algorithm pictured in gray in the similarity estimation step. Note that a list of true pairwise labels must be issued to the learning algorithm in order to build the model.

In this section, we review the litterature for each of these steps. For a in depth review of the litterature on CSI, we refer to Foster [41]. We base our discussion on the work realized by Foster and extend it.

2.1.1 Features extraction

Audio features are commonly extracted from the time-frequency representation of the audio signal and can be categorized in (1) low-level, or (2) mid-level features, as explained by Fu et al. [43]. Low-level features are computed using spectral analysis techniques on short audio frames (10ms - 100ms). Statistics can then be computed to aggregate the features on a track level. Examples of low-level features extracted from the magnitude spectrum are the *spectral centroid*, the *spectral flux* and the *spectral rolloff* [43]. The centroid is defined as a weighted average of the magnitude spectrum. The spectral flux is defined as the sum of squared errors between successive magnitude spectra (between frames) and finally, the rolloff corresponds to a specific percentile of the magnitude spectrum.

Alternatively, one might characterize the music with musical timbre. To do that, Tzanetakis et al. [109] use the Mel-frequency cepstrum coefficients (MFCC). MFCCs are obtained from log-magnitude spectra extracted using the short time discrete Fourier transform (STFT). The spectrum is divided in sub-bands and weighted to give weighted spectra. The spectrum is divided such that the sub-bands are linear at low frequencies, and logarithmic at high frequencies, following a perceptually motivated scale. The resulting signal is finally cosine-transformed to obtain MFCCs. As pointed out by Foster [41], the cosine transformation can be considered as a decorrelation step that approximates principal component analysis (PCA). Therefore the resulting coefficients in ascending order explain a decreasing amount of variance in the log-transformed magnitude spectrum. For music analysis, the use of 13 MFCC coefficients is common.

Mid-level features are alternative representations that are closer to the musical aspect of the music. They are designed to describe musical aspects of the music, such as the harmony, the melody, tempo, etc. Mid-level features are usually more challenging to compute [71]. The most typical mid-level features are chroma features [45], also known as Pitch Class Profiles (PCP). They represent the harmonic content of the music. Chroma features have been widely used in MIR tasks. They map octave-folded frequency bands to 12 pitch classes in the chromatic scale, thus representing the energy distribution across pitch classes independently of the octave for each pitch of the chromatic scale. Chroma features were introduced in 1999 by Fujishima [45], who computes an STFT for short audio frames and sums spectral energy in octave-folded frequency bands, to obtain 12 components. An example of the PCP representation of a C major chord is shown in Figure 2.2. Several other implementations have been developed [26, 76], and a comprehensive review of chroma features implementations can be found in [49]. Probably the most popular at this time is the Harmonic Pitch Class Profile (HPCP) introduced by Gomez [51]. The HPCP is a pitch class distribution (or chroma) feature computed in a frame basis using only the local maxima of the spectrum within a certain frequency band. It considers the presence of

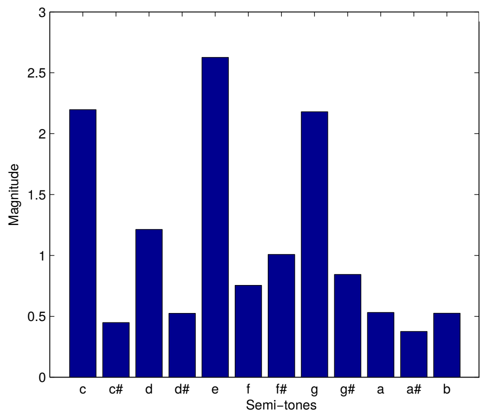


Figure 2.2: Example of the Pitch Class Profile (PCP) corresponding to a C major chord. The three main peaks correspond to the main notes of the chords: C, E, G.

harmonic frequencies, as it is normalized to eliminate the influence of dynamics and instrument timbre (represented by its spectral envelope). Other popular chroma features are CENS features, introduced by Kurth et al. [65]. They were designed to absorb variations in dynamics by applying an appropriate normalization. They further avoid the energy distributions that occur during passages of low energy. Finally, they are quantized using appropriate quantization functions and normalized with respect to the Euclidean norm.

Recently, Chen et al. [26] introduced a perceptually motivated chroma feature called the cochlear pitch class profile (CPCP). Their main idea is to combine concepts of psychoacoustics, such as the equal loudness contour and critical band, with the conventional PCP descriptor to enhance its discriminative power. The authors demonstrate the effectiveness of the CPCP in a CSI system (See Section 2.1.2). Another approach was suggested by Walters et al. [110] with a new feature derived from the chroma features, called the *intervalgram*. In their work, they do not compute chroma features from the audio spectrum directly, but from the stabilized auditory image obtained using a 64 channels filters bank. From the auditory image, they extract a sequence of 32-dimensional chroma features and they average them using different time-offsets to build their intervalgram. They successfully applied their feature in a CSI setup [110].

Besides chroma features, other mid-level features attempt at characterizing the musical rhythm. Tzanetakis et al. [109] detect periodicities by selecting peaks in the sample autocorrelation of the audio signal envelope. Audio features characterizing the audio tracks can also be as simple as the tempo, the number of beats and the duration of the song. These can be easily computed from the audio samples and do not require much processing. We demonstrate later that such simple features contribute to improve the performance of a CSI system.

Some authors consider the use of perceptually motivated audio features. Van Balen et al. [3]

introduce cognition-inspired descriptors for CSI. In particular, they consider *pitch Bihistograms*, which essentially capture how often two pitches p_1 and p_2 occur less than a distance d apart. The second feature they provide encodes which pitches simultaneously appear in a signal. They name their features *chroma correlation coefficients*. Finally, they build a *harmonization feature*, which is a set of histograms of the harmonic pitches $p_h \in \{1, \ldots, 12\}$ as they accompany each melodic pitch $p_m \in \{1, \ldots, 12\}$.

Features introduced in this section are considered raw, meaning they do not require a huge processing pipeline to be computed. For CSI, these features are usually post-processed to obtain a track-wise representation of the audio. Also, some sequences of features (*eg* chroma or MFCC sequences) can take time to be processed and thus require some sort of summarization for fast processing, for example in a CSI system. The next section covers the literature for CSI, and describes existing features post-processing methods.

2.1.2 Features post-processing

CSI systems generally assume that the sequential pitch, such as the melody, is preserved among the different versions of the same piece of music. Therefore, one can consider that a correct criterion for defining cover songs is that they match the same melodic progression. Several research works are based on the extraction of the predominant melody in a polyphonic signal. Tsai et al. [107] consider that the vocal content in music is relevant for identifying cover songs. Using a classifier based on MFCCs, they segment the audio into vocal and non vocal regions. They extract the predominant melody using the STFT and applying a peak picking algorithm. They thus sum energies across harmonics. If pitch bands are repeated across harmonics, they are discarded as it is likely that they are produced by a musical instrument, and not a singing voice. In another approach, Marolt [71] presents an approach in which he introduces a beat-synchronous melodic representation consisting in multiple salient melodic lines. He uses a two dimensional shift invariant transformation to extract shift invariant melodic lines. The melodic lines are then pruned by thresholding the salience values associated to the melodic lines, thus forming the mid-level representation.

Because extracting a reliable melody from a polyphonic signal is a challenging and yet unsolved task, most CSI system rely on mid-level representations characterizing the harmony of a song, such as the chroma features (see Section 2.1.1). CSI systems based on chroma features can be split in two categories: systems considering continuous chroma features, that is raw features, and systems considering quantized chroma features. The latter needs a post-processing step to build a higher-level feature robust for CSI.

2.1.2.1 Continuous chroma features

Systems using raw continuous chroma features take advantage of the entire information stored in the features. As chroma features are at least 12-dimensional, techniques based on continuous chromas tend to be slower than systems based on codebooks. However, they also tend to perform better due to the gain of information compared to quantized features.

As the different renditions of a song can vary in terms of tempo, techniques are needed to render the chroma features tempo-invariant. The predominant method is to use beat-synchronous features [5, 12, 13, 37, 58, 89]. The technique relies on an external beat tracker that estimates the times of the predominant beats. The chroma features can then either be computed at the beat times, or resampled in a post-processing step by applying an interpolation method. Using beat-synchronous features makes the music representation invariant to local and global tempo changes. They further allow a speed up in the recognition process as they usually reduce the length of the sequence of chromas. However, it also leads to a loss of information, and thus a potential decrease of the performance [5, 89].

In order to deal with changes in pitch, it is necessary to have key-invariant features. Chroma sequences corresponding to tracks played in different keys can be seen as sequences of circularly shifted chroma features. Indeed, because the pitch class is a modulo 12 representation of a chromatic pitch (see Section 2.1.1), a transposition implies applying a circular shift.

To address the issue of key invariance, several approaches have been investigated. One possibility is to consider all 12 possible circular shifts of the chroma sequence at the recognition step [37, 65, 71]. Comparing a track to 12 versions of the query might be time consuming. For this reason, it might be possible to reduce the number of shifts in a good trade-off between performance and efficiency.

Another technique, called the Optimal Transposition Index (OTI), was introduced by Serra et al. [88]. It consists in averaging both chroma sequences across time, then computing the inner product between the resulting chroma vectors. The inner product is then maximized with respect to rotations of the query average chroma. Once the optimal OTI is found, all original chroma features of the query are circularly shifted by the OTI amount. A number of works have been relying on the OTI technique [12, 13]. An alternative approach consists in estimating the key of one song, and to transpose the other one to the key of the first using circular rotation. This technique, suggested by Gomez et al. [51], is however less robust, compared to the OTI. Finally, some authors have been relying on the two-dimensional magnitude spectrum of the chroma sequences, which has the property of shift-invariance [13, 56]. The resulting representation is then invariant to pitch, as the 2D Fourier spectrum is invariant to pitch shift.

2.1.2.2 Quantized chroma features

Systems based on quantization aim at classifying chroma features in a reduced set of labels, compared to the universe of possible continuous chroma features. Such techniques usually rely on the construction of a codebook on which raw chroma features are mapped. They present the advantage of simplifying the system and often speed up the recognition rate. On the other hand, mapping continuous features to discrete labels inevitably leads to a loss of information, potentially reducing the recognition rate in a functional system. Casey et al. [22] make use of the K-Means clustering algorithm to build a codebook with a size varying in the range [8, 64] interval. In another approach, Kurth et al. [65] use musical knowledge to build a set of 793 chroma codewords. Instead of relying on a clustering algorithm, one can make use of a chords recognition algorithm to convert a song to a sequence of chords labels. In that case, the set of codewords is relatively small. Bello [5] uses a 24-state Hidden Markov Model (HMM) based on two chord types (major and minor) and represents a song as a string of chords labels. Lee [68] uses a more complex HMM with 36 states trained on labelled chord sequences. Some research works try to match chroma vectors to binary templates in order to extract chords labels [58, 72].

2.1.3 Similarity estimation

Once the features have been computed and post-processed, one needs methods to compare a query to a set of reference tracks. Techniques depend on the use of continuous, or discrete features. Similarity estimation between two audio tracks can be done through alignment of sequences of time-ordered features, which is inspired by time-series literature. To allow fast retrieval in a large database, several fast methods have also been developed, as pictured in Figure 2.3. The latter can be divided in three specific techniques, respectively based on index-based methods, features summarization and fast alignment algorithms. Finally, some authors attempt at mixing features and distance measures using multiple methods.

2.1.3.1 Alignment of sequences of features

One way of evaluating similarity between two songs is to consider a sequence of features sampled over time in both songs. Such features can correspond to chroma features, or chords labels, or even timbral features. Techniques based on dynamic programming can then be used to compare both sequences and compute their similarity. Comparing sequences of features is not a trivial task, however, because sequences corresponding to different audio tracks often have a different length.

To compare two sequences of chroma features, alignment algorithms are typically used. Gomez et al. [51] apply Dynamic Time Warping (DTW) to chroma sequences. DTW computes a

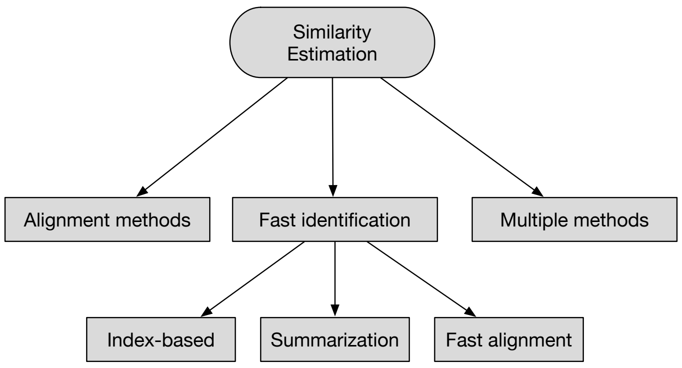


Figure 2.3: Similarity estimation step of the general CSI pipeline.

similarity matrix D between both sequences, with each point corresponding to the computed similarity between two features vectors. The similarity can be computed using a Euclidean distance, or cosine similarity for example. An optimal cumulative score is then computed between both sequences, by setting costs for inserting or deleting features vectors from one of the sequences. The number of deletions and insertions constitutes a global alignment between both sequences. DTW thus builds a similarity score while being invariant to changes in tempo, by inserting or deleting feature vectors. This presents the advantage that it does not require to rely on beat-synchronous features. Figure 2.4 shows an example of global alignment path between two time-series using DTW [25]. As the value at point $D(i, j)$ is the minimum distance between both series, then if minimum distances are known for smaller portions in the matrix D that are one point smaller than i and j, then $D(i, j)$ is the minimum distance of all possible paths for series that are one data point away from (i, j), plus the distance between both points x_i and y_j (see Figure 2.4). The distance can thus be evaluated recursively as

$$D(i,j) = D(i,j) + \mathrm{min}(D(i-1,\,j),\,D(i-1,\,j-1),\,D(i,\,j-1)) \tag{2.1}$$

Before applying DTW, Gomez et al. [51] build a binary matrix by thresholding pairwise distances between feature vectors. They also use a local-alignment algorithm based on the Smith-Waterman method. As described by Foster [41], local alignment determines the optimal set of insertions and deletion from the query with respect to all contiguous subsequences of the query and comparison sequence. Local alignment is thus able to account for structural changes between versions, for example an omitted verse or chorus in a cover song. Note however that the use of the DTW or Smith-Waterman algorithms is time consuming and does not allow to quickly compare a song to a large collection of tracks.

The most efficient technique so far based on alignment was provided by Serra et al. [92]. Instead of considering individual chroma vectors, the authors embed a succession of chroma

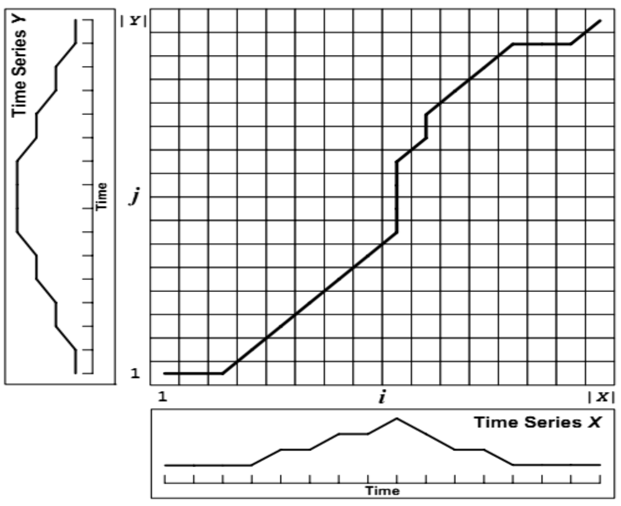


Figure 2.4: Global alignment path between two sequences of time-ordered features. This figure is borrowed from [25].

features to capture local information on temporal structure. They threshold pairwise distances between vectors of chroma sequences to obtain recurrence plots. Recurrence plots correspond to distance matrices between sequences of features, similarly to the cost matrix in the DTW algorithm. From the recurrence plots, the similarity between a query and a comparison track is quantified with a score computed from the diagonal path in the recurrence plot, similarly to DTW. The authors further account for local and global tempo changes by allowing variations in the path (in terms of curvature and angle). They include these variations of DTW in their own alignment algorithm, named QMax. This algorithm is detailed in Section 5.1.2. The same alignment algorithm was also successfully applied by Chen et al. [26] to evaluate their own features (CPCP, see Section 2.1.1) against the HPCP.

In further work, Serra et al. [90] build a predictive model from the query. The parameters of the models are obtained by assessing the prediction error with the query itself. They next use the model to obtain a sequence of prediction of a comparison track, that is they use the model built from the query to predict a sequence for a comparison track. That process is called *cross-prediction*. The cross-prediction error is then taken as a pairwise dissimilarity measure between cover-songs.

In a different approach, Ellis et al. [37] use two-dimensional cross-correlation between beat-synchronous chroma sequences as a measure of similarity between two songs (see Section 5.1.4). They measure similarity by applying a peak-picking algorithm to the sample cross-correlation. If two songs are similar, they yield a strong correlation at a specific time lag. The similarity measure is computed as the reciprocal of the maximum value in the cross-correlation signal. Jensen et al. [57] also use signal processing techniques to identify cover songs. Specifically, they use the two dimensional sample auto-correlation and sum the values of the resulting signal in 16 exponentially distributed bands to obtain a representation that is insensitive to different tempi. In a similar manner, Bertin-Mahieux et al. [13] compute 2-dimensional power spectra to obtain a key-invariant representation of the chroma representations. The Euclidean distance is used in both cases to compute a similarity score between two songs.

Rather than using slow alignment algorithms based on DTW, Foster et al. [42] makes use of an information-theoretic approach applied to both quantized and continuous chroma features. In particular, they consider approaches based on the normalized compression distance (NCD), which is a string compression method that quantifies similarity between two strings in terms of joint compressibility. Techniques based on compression methods make use of the number of bits required to encode a given string, where a string can be a sequence of quantized chroma features, for CSI. They successfully apply their technique on quantized and continuous features and produce state-of-the-art performance.

Instead of using chroma features, Tralie et al. [105] consider the use of timbre features based on self-similarity matrices of MFCC coefficients. They produce sequences of timbre based features and perform an alignment between two songs using the Smith-Waterman alignment algorithm. In contrast with other works related to MFCC, they innovate by examining the relative shape of the timbre coefficients. They demonstrate that based on such features, cover song identification remains possible, even if the pitch is blurred and obscured.

2.1.3.2 Fast identification

While alignment techniques for measuring cover song similarity produce good performance, they are usually too slow to be used in a large scale environment. In order to compare a query to millions of tracks, one needs fast audio representations and fast matching algorithms in order to quickly identify matching candidates. Among existing fast methods, one can distinguish between methods based on an inverted file, which allows almost instantaneous access to the matching tracks, methods based on features summarization, which reduces feature dimensionality, and methods based on fast alignment algorithms. In the first case, track representations are mapped to one or multiple index positions, at which one can find lists of potential matching tracks to the query. In the second case, sequences of features are usually compressed to some low-dimensional representation, which can in turn be compared to a large database by means of simple distances.

Index-based methods

Index-based methods rely on the concept of inverted file, which is widely used in the information retrieval literature. The concept is as follows. Assuming a list of documents, with each document being assigned a set of keywords or attributes, an inverted file is the sorted list of keywords (or attributes), with each keyword having links to the document containing that keyword. The use of such an inverted file improves search efficiency by several orders of magnitude.

Kurth et al. [65] suggest a method for identifying audio tracks containing excerpts similar to a provided query. While this is not exactly the same problem, their work could be applied for CSI. They absorb the variations between versions at the feature level by using their own version of chroma features, named CENS features (see Section 2.1.1). They quantize the CENS features to a fixed set of 793 codewords, with a codebook built manually based on musical knowledge. They finally identify matching tracks using an inverted file method, where each quantized sequence of CENS features is mapped to index positions containing the potential matching tracks. They further introduce a higher degree of robustness against variations by employing fuzzy and mismatch searches.

Alternative approaches for fast identification using index-based methods rely on locality-sensitive hashing (LSH) [97]. LSH is a projection operator that aims at reducing the dimensionality of the feature space. Given two features vectors (potentially high-dimensional), LSH maps the vectors to lower-dimensional vectors so that the similarity between the lower dimensional vectors relate to the similarity between the original vectors. In addition, a hash function maps the low-dimensional feature space to a set of hash values. LSH thus allows to map two similar features vectors to the same hash value. Similar tracks to a query can thus be retrieved almost instantaneously. Casey et al. [21] perform version identification using a nearest-neighbor search. They obtain two sets of chroma windows for a query and a comparison track. The similarity is then quantified by the number of windows in the query track for which there is a close fragment in the comparison track. They define similarity in terms of Euclidean distance and use LSH to reduce dimensionality. LSH is also used by Walters et al. [110], in combination with the Intervalgram feature described in Section 2.1.1.

Summarization methods

Methods based on dimensionality reduction are built upon a summarization of some higher-dimensional features. An approach introduced by Khadkevich et al. [58], considers tracks summarization by chords profiles, which correspond to distributions of chords within the songs, using 24 chords labels. They use LSH to identify nearest neighbors to the query. They further re-rank the list of results candidates using an edit-distance between the chords sequences of the query and results candidates. A different method based on hashing was provided by

Bertin-Mahieux et al. [12], in which they binarise chroma sequences using an adaptive method. Each fragment of the sequence is then mapped to a hash value, using chroma peak locations as a basis.

In further work, Bertin-Mahieux et al. [13] consider patches of 75 consecutive chroma features, and compute the 2D power spectrum of each patch, keeping the magnitude coefficients. They next aggregate the patches using a point-wise median aggregation rule, and project the resulting 900-dimensional vector on a 50-dimensional PCA subspace. Similarity is computed using the Euclidean distance. The advantage of that method is that the 2D power spectrum is invariant to shift, and thus renders the track representation pitch invariant. Based on the same representation, Humphrey et al. [56] improve upon previous work by using data-driven projections at different time-scale to capture local features. They next embed summary vectors into a semantically organized and high-dimensional space, using unsupervised learning. After aggregating the beat-chroma projections, they apply supervised dimensionality reduction to recover a low-dimensional space such that distances can be computed fast.

Fast alignment methods

As an alternative to index-based and summarization methods, Martin et al. [72] designed a CSI system based on a fast local alignment algorithm called BLAST. The algorithm relies on the observation that when querying a new sequence to a large database, there are only a small number of good alignments. So it filters out the database to avoid computing irrelevant alignments. Applied to CSI, this reduces the set of sequences to be considered for local alignment and thus speeds up the identification process. Similarly, Silva et al. [94] makes use of time series based techniques called *shapelets* [112], based on the hypothesis that they can characterize cover song similarity using only small segments to characterize the audio tracks, rather than full length tracks. The so-called shapelets are informally defined as the subsequence that is the most representative of a track. Shapelets computation first extract all subsequences of a track, and then select a subset of the subsequences using a quality assessment algorithm. It then constructs a decision tree to compare a query to a set of shapelets. The authors adapt the method to *musical shapelets*, based on CENS chroma features. They measure the distance between the query and all the shapelets found at the training phase for the database.

Finally, a recent method, called SiMPle, was introduced by Silva et al. [95], based on chromas subsequences similarity joins, which identify the nearest neighbors of each subsequence of a track A in a track B. Based on the similarity joins, they encode the distance to the nearest subsequence, for a given subsequence in track A, as well as the position of the nearest subsequence in B. They store that information in a compact representation, the *Similarity Matrix Profile*. The data structure thus encodes structural information, which identifies meaningful moments in a song. For similarity estimation, they compute the similarity matrix profile between the query and each track of the database and take the median of the profile as the distance measure.

2.1.3.3 Multiple features and distances

A few authors have investigated different ways of considering several features or distance measures, rather than considering a single feature and similarity estimation method. Among them, Ahonen et al. [1] investigate the combination of three representations based on discretized sequences of features extracted from chromagrams. The first representation is a set of labelled chroma features, discretized using a 12-states HMM. The second one considers the Manhattan distance between each successive chroma vector. The distances are discretized using the SAX method. The last representation considers a sequence of the dominant pitch class in each chroma vector, in order to extract melodic information. All three representations produce similarity scores between pairs of tracks using the NCD, which is a parameter free similarity metric based on data compression. A global score is produced as the average of the three NCD scores.

In another approach based on compression based distances, Foster et al. [42] consider the combination of five systems returning scores based on the NCD. As in the work of Ahonen, they consider chroma vectors as their feature. They compute a weighted average of the distances using max and min operators, which have been used as a mean of combining probability estimates by Kittler et al. [64].

Still based on chroma features, Chen et al. [27] consider three variants of the chroma features: the *Cochlear Pitch Class Profile* (CPCP) [26], *Harmonic Pitch Class Profile* (HPCP) [51] and beat-synchronous chroma vectors, as detailed in the work of Ellis et al. [37]. They use QMax [92] as a distance measure for the CPCP and HPCP, and cross-correlation for the beat-synchronous chroma vectors. They next fuse similarity matrices produced by the three systems using a technique herited from the biological field called *Similarity Network Fusion* [111].

Kim et al. [62] build a fingerprint for classical music audio tracks that encapsulates different informations based on chroma features. In that case, the combination is done at the feature level, and the final similarity score is computed using a template matching algorithm.

Recently, a new filter-and-refine method was studied by Cai et al. [20], using a two-layers algorithm based on the 2D Fourier transform magnitude coefficient of chroma patches, and structural segmentation. They first run the algorithm provided by Bertin-Mahieux et al. [13] to quickly discard unrelevant tracks. For each of the resulting candidates, they compute a similarity score with the query using structural segmentation. They first apply a structure detection algorithm, provided by Serra et al. [91]. For each isolated segment, they extract again 2D-FFT magnitude coefficients as the main feature. They next compute pairwise cosine similarity between segments of both the query and the comparison track, and keep the minimum distance for each segment. Finally, they use four simple algebraic operations for computing a final similarity score from the ordered list of minimum distances between the pairwise sections.

2.2 Discussion

From the literature review, one can conclude that existing CSI systems mainly rely on chroma features. Many chromas post-processing methods have been investigated, and multiple similarity estimation algorithms have been studied. However, despite all these existing methods, the problem of identifying cover songs is not solved yet. So far, the most promising technique was the one based on cross recurrence plots, introduced by Serra et al. [92]. However, the method was released in 2009, and it has been hard to tell if any newer method performs better today. The problem is that it is difficult to compare cover song identification systems, because of multiple reasons.

The main reason is that researchers tend to evaluate their systems on personal collections. Therefore, they report performance each on different databases, which are not necessarily comparable. Most systems are further evaluated on small databases, not competing with commercial databases, that claim to contains millions of songs. Although some initiatives like MIREX [29, 31] evaluate systems on a common databases, they only provide 3,300 songs at most.

Other authors consider various databases, not always available for everybody. For instance, Casey et al. [23] consider a set of 2,018 songs, spanning the collection of only two artists. Ellis et al. [37] evaluate on a small test set containing 94 pairs of songs, coming from various collections. They also submitted their work to MIREX, thus providing an evaluation on a database of 3,300 tracks. Kim et al. [62] only consider a database of 107 pieces of classical music. They later improved their technique and evaluated on bigger database containing 2,000 pieces of classical music [61]. In that specific case, it is difficult to compare a system evaluated with classical music with systems considering general western music.

Several authors evaluated their work on a small database, the Cover80, containing 160 songs in total, which is small compared to other databases [3, 36, 105]. The reason for such a small database is that it provides full audio content, which is not the case of some other datasets. This allows authors to compute and design their own audio features. In contrast, the Million Song Dataset (MSD) [14] is a database containing pre-computed features for one million songs. They do not provide audio data, but pre-computed features. Several authors evaluate their work on that database [13, 15, 42, 56, 72, 80].

In addition to the problem of databases, authors do not present their results using the same performance metrics or spaces. Most research works present results based on single performance metrics, which only consider the proportion of tracks identified at the first position, for instance [37, 53, 62, 69]. Sometimes, the use of an evaluation space is preferred [23].

Clearly, there is some work left to do in the field of cover song identification. In this thesis, we make an attempt at answering some of the questions raised by the former discussion. Thus, we consider a new evaluation space in order to compare multiple systems in the same exact conditions. We also studied a subset of existing systems, implemented them and evaluated them on a rather large database, as will be explained in the remaining of this thesis. And finally, we studied ways of combining multiple systems together, in order to improve current performance.

2.3 A combining approach

In some CSI systems, other features than chromas, such as the melody, or the timbre are considered. However, most systems only consider a single feature. Another observation is that although many systems are based on chroma features, they usually make use of different similarity estimation algorithms: some use simple distances, while some others use complex alignment algorithms. One can thus make the hypothesis that CSI systems based on different features and different similarity estimation algorithms produce different orderings of the results. For instance, a system based on the tempo will not identify the same tracks than a system based on the chords progression.

The approach we investigate is based on such different features and comparison algorithms. We consider as many input features and comparison algorithms as possible. Such features span simple musical facets such as the tempo, the duration, the number of beats, and more complex music characteristics, such as chroma features for harmony, MFCC features for timbre, melody, musical structure etc. As the implied musical characteristics are all different, any system based on these features should contribute to a performance improvement.

To be able to consider all that information for a CSI system, it is required to somehow aggregate the individual information to produce a composite system that outperforms any individual system. One way of doing that is to aggregate multiple features to build a composite feature, and then use some similarity estimation algorithm to compute a score, considering the aggregated feature. However, designing such a comparison function is not an easy task, as the dimensionality of the features might increase considerably. For that reason, we consider individual CSI systems, based on different features and different similarity estimation methods. Each system thus produces a different ordered list of results, with respect to a query. In that case, it is necessary to combine the output of individual CSI systems and build an aggregated list of results that outperforms the initial ones. Such an approach presents several advantages. As the the initial systems are considered individually, one can choose to run them on different computing nodes or cores, which might reduce the overall computational time. If there is a time constraint on the final application, one might also select only a subset of systems rather than running the full pipeline.

Therefore, we introduce an approach to cover song identification based on the combination of multiple initial CSI systems based on multiple input features. Figure 2.5 shows the pipeline (initially described in Figure 2.1) for a CSI system that includes our contributions. The features extraction step considers multiple features, including tempo, beats, duration, etc. We design several new CSI systems based on single dimensional features. We design them using learning algorithms to build a similarity function, as illustrated in the similarity estimation step, in Figure 2.5. We also provide novel methods based on chroma features, specifically based on the average chroma vector or a song, as well as a clustering of chroma features in a fixed set of codewords. These methods will be described in Chapter 5.

We also include several state-of-the-art methods as they have been proven to produce a good performance. We also include one method based on the musical structure of the songs. Each considered system therefore produces a single ordered list of results. We thus end up with multiple list of results that should be combined using some aggregation method. We provide an evaluation of multiple combination methods, based on ranks or probabilities. These will be respectively described in Chapters 7 and 8. The selected combining algorithm thus takes advantage of all lists produced by a each system to build a composite list of results in which all versions of a query should be ranked at the top of the list. In order to achieve that, we need to implement multiple CSI systems, and evaluate them on a common database, preferably large enough to be representative of a correct sample of existing music.

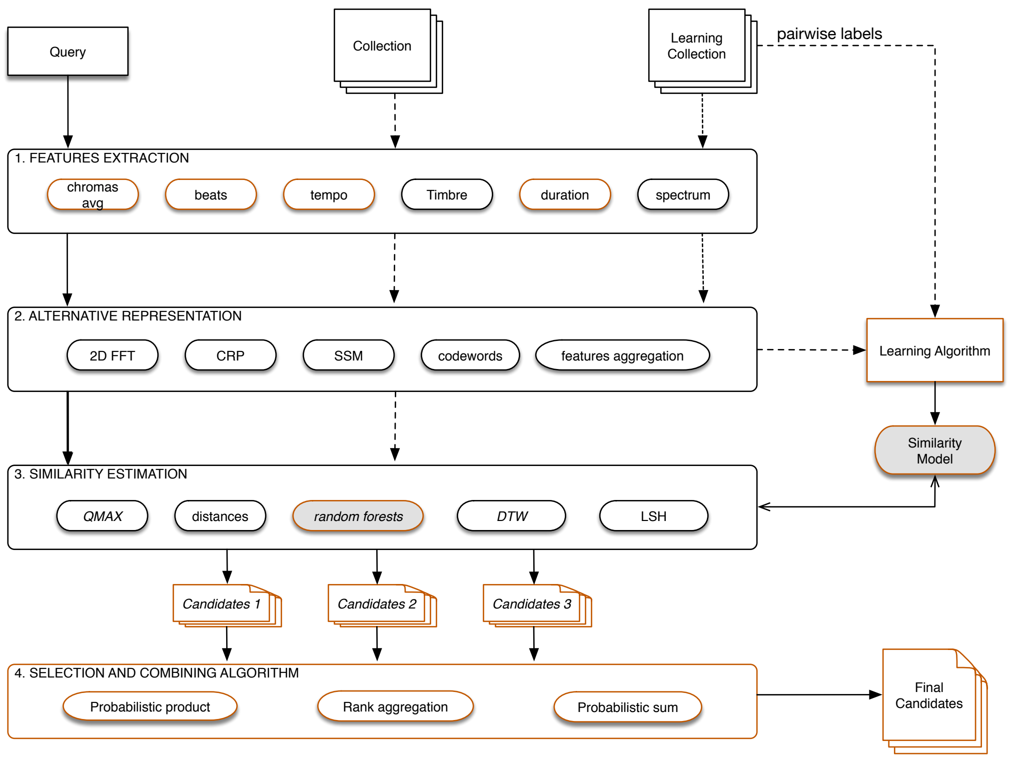


Figure 2.5: Our modified pipeline for cover song identification. The query is compared to the reference collection using multiple features and similarity estimation methods, thus producing multiple ordered list of results. All lists of results, or a subset of them are then selected and combined using an appropriate combination algorithm. Orange boxes represent our improvements compared to the literature.

CHAPTER 3

Single Match Retrieval Systems

A cover song identification system can be seen as an *Information Retrieval System* (IRS) relying internally on a *Content Based Retrieval System* (CBRS). Such a system finds the information requested about a queried object by retrieving the information associated to similar objects in a search collection. For CSI, the query object corresponds to a full-length audio track, and the retrieval system should return all other versions of the query track, that is audio tracks considered similar in the search collection. This is the traditional definition of the problem of CSI, as considered by the majority of existing research works in that field. Although recent CSI systems considerably improved the performance over previous work [27, 92], existing results tend to show that the problem is still far from being solved. Indeed, the best performing system so far is based on a temporal alignment algorithm of chroma sequences [92], and identifies less than 50% of the queries in a large collection, as will be shown in the upcoming chapters. In addition, few systems have been evaluated on search collections containing more than several thousand tracks [3, 23, 26, 27, 37, 53, 61, 62, 65]. This is partly due to the lack of standardized audio content datasets for the task of CSI. This issue regarding audio data access can be generalized to other MIR tasks, principally because of legal and copyright reasons. Owners of large databases of music are typically commercial companies which cannot release audio content for right management reasons, even for research purpose. It is therefore difficult to compare results of existing systems because their evaluation procedures are different from each other.

The question of the evaluation of MIR tasks has been discussed multiple times at meetings and conferences between researchers from the community. In 2012, at the *International Symposium on Music Information Retrieval* (ISMIR conference) in Porto (Portugal), people gathered together to discuss such evaluation questions, pointing out that there are indeed issues for evaluating MIR tasks that should be answered by the community [83]. Four years later, at the same ISMIR conference in New York (USA), similar questions were discussed, following the presentation of a novel sustainable plan for the evaluation of MIR tasks, proposed by Mc Fee et al. [75]. This shows that the problem of MIR evaluation is a complex one, and no standard evaluation methods have been accepted by the research community. The question of evaluation therefore remains an open question and should lead to new research and development in that area.

Having studied the literature on CSI, we observe that, at this time, no existing system is close to solving the problem of CSI. However, the CSI field has seen an increase of the performance over the years, and while the systems are not able yet to strictly identify cover songs, existing works could be used to reduce the size of the search collection, while maintaining a high probability of identifying a query. That is the reason why we introduce in this chapter a *relaxation* to the problem of CSI. We claim that if it is possible to identify *at least one match* to a query, than the latter should be identified by an external expert, computational or human. That problem is slightly different from the original problem of identifying *all* covers for a given query. We explain that a content based retrieval system based on that definition can be implemented in two steps. The first one (a pruning algorithm) identifies the samples of the search database that are potentially related to the query, and the second one (an expert) predicts the requested information based on the resulting set. We explain that CSI systems can be related to the pruning step mentioned earlier. For that step, we show that at least one single match to the query is necessary in the pruned set to get the requested information about the query. Therefore, we name *Single Match Retrieval* (SMR) system this kind of IRS based on a CBRS.

Having not found any satisfactory procedure for evaluating CSI systems based on SMR, we evaluate the pruning step in a new performance space named the *Prune-Loss* space. We demonstrate the compatibility of that evaluation space with other standard evaluation spaces and standard metrics used in the literature. We further show that the resulting performance curves are useful to compare the performance of multiple retrieval systems, even evaluated on different test collections.

The remainder of this chapter is organized as follows. Section 3.1 defines rigorously the task of cover song identification. It describes the pipeline of a SMR systems and defines the requested notions of pruning and loss. Section 3.2 explains how an SMR system should be evaluated based on the notion of pruning and loss. It further demonstrates that existing evaluation spaces and scores used in MIR and other fields (e.g. computer vision) are not suitable for the task of

CSI. The section however emphasizes a compatibility between the defined Prune-Loss space and existing spaces, in particular with the Receiver Operating Characteristic (ROC). In Section 3.3, we discuss the results and conclude this chapter.

3.1 Overview of the SMR pipeline

Let us denote the space of input objects by $\mathcal{I}$ and the space of output information by O. For example, an object can be a sound, and the information can be the author of the song. In the following, an object will always be an *audio track*. We introduce the function $f : \mathcal{I} \rightarrow O$ associating labels corresponding to the tracks, and the similarity function $s : \mathcal{I} \times \mathcal{I} \rightarrow \mathbb{B}$ such that $s(t_1, t_2) = 1 \Leftrightarrow f(t_1) = f(t_2)$, where $\mathbb{B} = \{1, 0\}$ denotes the set of booleans; it is common to use 1 to denote "true", and 0 to denote "false". These two functions are related to the ground truth, which associates the *true* label to a pair of tracks drawn from $\mathcal{I}$. Whereas an IRS aims at defining a function $\hat{f}$ as close as possible to f, a CBRS defines a function $\hat{s}$ approximating s. Indeed, finding the information related to a query $q \in \mathcal{I}$ is a difficult task because it involves a similarity measure which, in general, is unknown and difficult to estimate. In practice, s can only be approximated by a function $\hat{s}$.

The approach for constructing an IRS based on a CBRS is based on our observation that, in practice, an *expert* knowing s but not f is often available even if using it is time consuming. For example, consider the problem of version identification. A user requests some information about an unknown song, such as the title, or any other information related to the original track. A search database $\mathcal{D}$ contains audio tracks labeled with some information, which is identical for all versions of the same song. We assume that the user has the necessary psycho-acoustic knowledge to compare two audio excerpts, but he cannot compare his query to all songs in the database (considering that the database contains thousands of songs). However, he could identify his query based on a small subset of $\mathcal{D}$ obtained with a pruning algorithm based on a similarity measure. In this case, the user submitting the query is the expert, but an algorithmic expert can also be considered.

The pruning, which is based on the function $\hat{s}$, is the cornerstone of the systems studied in this chapter. Pruning has been widely used in CBRS as well as in *Text-Based Retrieval Systems* (TBRS). An example of TBRS method using pruning is WAND [18] which uses a two-level process that first identifies candidates documents using a preliminary evaluation and then performs a deeper analysis to return the final answer. Pruning has also been used in image retrieval systems, such as *Viper* [99] in which the authors analyze how the number of features used for pruning affects the ranking of the retrieved images. In [101], Summengen et al. presents a category pruning method for image retrieval. Another method using the Scale

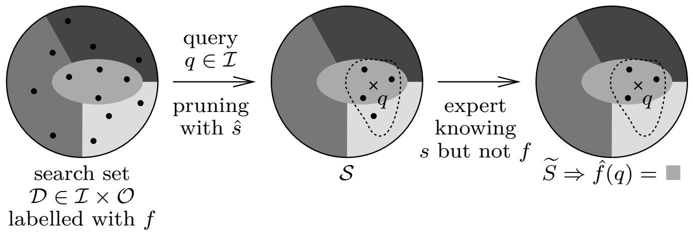


Figure 3.1: The processing pipeline of an IRS based on a CBRS. Gray levels indicate values in O whereas a point indicates a sample from the space $\mathcal{I}$ (an audio track, in the case of CSI). Samples within the same level of gray are associated to the same information. The first step is the SMR problem which aims at pruning the database. Next, an expert identifies the correct match in a small subset $\mathcal{S}$ of the database. The search engine succeeds if the expert finds at least one match similar to the query q in the pruned set.

Invariant Feature Transform (SIFT) is presented in [39]. All these pruning methods can take advantage of the evaluation technique presented.

Problem statement

An IRS based on a CBRS works as depicted in Figure 3.1. The search database $\mathcal{D}$ is a set of samples drawn from the space $\mathcal{I} \times O$. All samples $(t,\, l) \in \mathcal{D}$ are such that $f\,(t) = l$. For a query q, the CBRS prunes the search database and returns the subset

$$\mathcal{S} = \{(t,\, l) \mid (t,\, l) \in \mathcal{D} \,\wedge\, \hat{s}(t,\, q) = 1\}\,, \tag{3.1}$$

where $\wedge$ denotes the logical AND operator. Then, an *expert* knowing s (but not f), finishes the identification. It identifies the subset

$$\widetilde{\mathcal{S}} = \{(t,\, l) \mid (t,\, l) \in \mathcal{S} \,\wedge\, s(t,\, q) = 1\}\,, \tag{3.2}$$

and, if $\#(\widetilde{\mathcal{S}}) \neq 0$, where the symbol $\#$ denotes the cardinality of a set, it returns $\hat{f}\,(q) = l$ where $(t,\, l)$ is any element in $\widetilde{\mathcal{S}}$. Otherwise, the identification is not possible, maybe because the CBRS has dropped all the correspondences with the query during the pruning. The identification will succeed if and only if $\#(\widetilde{\mathcal{S}}) > 0$. If $\#(\widetilde{\mathcal{S}}) = 0$, then the identification fails and nothing should be returned. Having more than one element in $\widetilde{\mathcal{S}}$ is useless for the expert, since all the elements in $\widetilde{\mathcal{S}}$ share the same information. Therefore, we name this problem *Single Match Retrieval* (SMR). The aim of an SMR system (that is an IRS based on a CBRS) is to prune as much as possible the search set without dropping all the correspondences, in order to ease the expert's job.

The estimated similarity function $\hat{s}$ used for pruning is called a *similarity estimator*. That similarity function behaves such that $\hat{s}(t_1, t_2) = 0$ can be returned only when the classifier has a high confidence in the dissimilarity between t_1 and t_2. This behavior may be required in order to avoid loosing all the matches in the subset $\mathcal{S}$. In practice, the ultimate goal of a CSI system following the SMR principle is to build a function $\hat{s}$ that is able to classify a track into two classes: *similar* (denoted by the symbol 1) or *dissimilar* (denoted by the symbol 0), with respect to a given query q. Multiple options exist, as will be explained in the next chapters.

3.2 Evaluation of SMR algorithms

The evaluation of an SMR algorithm requires a test set $\mathcal{D}$ sufficiently large to contain many elements with the same associated information. We define a *clique* as a subset of $\mathcal{D}$ gathering the samples that share the same information. The function c gives the clique corresponding to any track $t \in \mathcal{I}$: $c\left(t\right) = \left\{\left(t, l\right) \in \mathcal{D} \,\mathrm{such\ that}\, l = f\left(t\right)\right\} \subseteq \mathcal{D}$. Note that the cliques may be of various sizes in the same dataset.

We define two ways of evaluating SMR systems, corresponding to different use cases. We name such a use case a *context*. The first one assigns the same weight to each clique of a dataset, thus giving the same chance to identify different queries corresponding to different cliques. Indeed, as many researchers still evaluate their CSI systems on personal collections, some songs might be more represented than others. Moreover, most databases are small in size and do not correspond to a representative sample of existing music. For this reason, in order to render research works comparable, even on slightly different databases, we normalize the evaluation with respect to the number of cliques of the collection. This is what we describe in the remainder of this chapter.

On the other hand, if one needs to evaluate a CSI system on a rather large collections, that might be representative of the existing music (consider for example the database of a company such as Spotify, or Pandora), it is necessary to take into account the size differences of the cliques, to report the real performance on that specific database. We describe that scenario in the next chapter, and suggest a slightly different evaluation.

3.2.1 Usual spaces used to evaluate the performance

Typically, search engines are evaluated using standard scores such as the number of *true positives* (TP), *true negatives* (TN), *false positives* (FP) and *false negatives* (FN). These elements define the *confusion matrix*. They are dependent on the decision threshold of the underlying classifier. The similarity estimator $\hat{s}$ classifies a track as True or False based on a threshold τ. If the similarity score is above τ, then the track is considered similar (True), otherwise it is

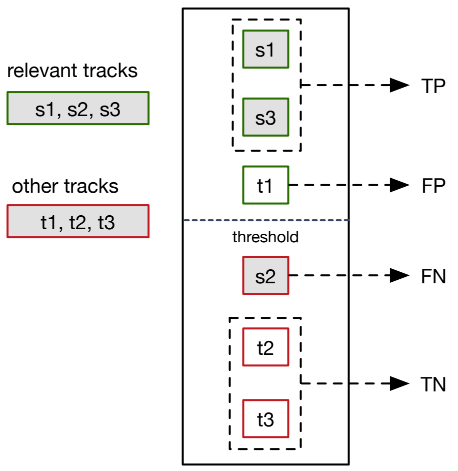


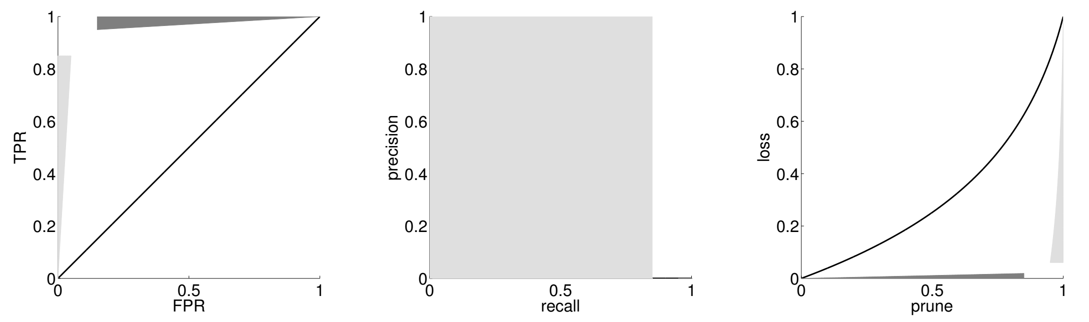
Figure 3.2: True positive, negatives and False positive, negatives for a given query with its corresponding relevant tracks. If the relevant tracks are considered similar, they are true positives, otherwise, they are false negatives. In the figure, there are 2 true positives, 1 false positive, 1 false negative and 2 true negatives.

dissimilar (False). Thus, for a given query q, if a relevant track (that is a track that belongs to the clique $c(q)$) t_i is considered to be similar, then it is a true positive for the query q. If the track is considered dissimilar, while it should be similar, then there is a false negative for the query q, as illustrated by Figure 3.2.The best threshold is application dependent. Therefore, in order to specify the performance of the search engine, one has to report the way it varies with the decision threshold. This is traditionally done graphically within the *Receiver Operating Characteristic* (ROC) or *Precision-Recall* (PR) spaces. We describe these two spaces below as well as their drawbacks for the SMR problem.

ROC space. The ROC space represents the effectiveness of a classifier in terms of its *True Positive Rate* (TPR) and its *False Positive Rate* (FPR).

$$\begin{aligned} \text{TPR} &= p\left(\hat{s}=1|s=1\right) \simeq \frac{TP}{TP+FN} \\ \text{FPR} &= p\left(\hat{s}=1|s=0\right) \simeq \frac{FP}{TN+FP} \end{aligned}$$

where $p\left(\cdot\right)$ denotes a probability. The probabilities are approximated by the elements of the observed confusion matrix. Alternatively, the performance can be expressed in terms of the *False Negative Rate* (FNR) and *True Negative Rate* (TNR) since TPR $+$ FNR $= 1$ and TNR $+$ FPR $= 1$. A ROC curve helps to predict the performance and to select the decision threshold in a context where maximizing the TPR and minimizing the FPR are simultaneously



(a) ROC Space (b) PR Space (c) PL Space

Figure 3.3: Relationship between ROC, PR and PL spaces (see Section 3.2). The black curves depict the performance of the random similarity estimator. The shaded areas represent the corresponding subsets of performances in the three spaces. One can observe that the PR space does not allow to see anything meaningful in a pruning context (dark shaded areas) due to the low positive prior.

required. In the context of SMR, maximizing the TPR is meaningless since we do not aim at retrieving all the database samples that are in the same clique as the query. Instead, we aim at giving the guarantee that at least one of these samples is retrieved. It follows that the interpretation of a ROC graph is not straightforward for SMR.

Precision - Recall (PR) space. The PR space represents the performance of a classifier in terms of the precision and the recall. PR space allows to select the decision threshold, with the aim of maximizing both the precision and the recall.

$$\begin{aligned} \mathsf{P} &= p(s=1|\hat{s}=1) \simeq \frac{TP}{TP+FP} \\ \mathsf{R} &= p(\hat{s}=1|s=1) \simeq \frac{TP}{TP+FN} \end{aligned}$$

The PR space has three main drawbacks for the SMR assessment:

1. The same drawback as the one explained above for the ROC space also applies to the PR space since R = TPR.

2. We consider the classification as a mean of pruning. In this context, it is much more important to have a high TPR (in order to avoid loosing all the matches in $\mathcal{S}$) than to have a low FPR (*i.e.* to prune aggressively $\mathcal{D}$). However, it turns out that only a very small part of the PR space corresponds to this goal, as shown in Figure 3.3 (explained in details in Section 3.3). In particular, the precision is expected to be very small since we expect $TP \lll FP$ for a pruning classifier (otherwise, the expert would be useless in the

second step).

3. For obvious practical reasons, the search set $\mathcal{D}$ does not contain samples drawn from the whole space $\mathcal{O}$ because most tracks in $\mathcal{O}$ may be unrealistic. For example, at a given time, all possible songs have not yet been composed. Therefore, the set of realistic samples (and the number of cliques in $\mathcal{D}$) is expected to grow with time for most applications. In this case, both FP and TN will increase in a similar way while TP and FN will remain unchanged. It follows that the precision is not a reliable measure of performance since it will drop.

3.2.2 Prune Loss space

Because of the previously mentioned drawbacks, we provide an evaluation space, named *Prune-Loss* (PL), to represent the performance of SMR retrieval systems. Our new space focuses on the *pruning* and *loss* rates, which are defined as follows. We make three assumptions:

[A_1] We assume the independence of the decisions taken by the estimator for several track t_i, conditionally to the knowledge of $s\,(q,\,t)$.

[A_2] We assume that the diversity of the tracks is equivalent for all cliques, and thus that the performance of the retrieval system is not clique dependent. We assume that, inside of any clique, the dissimilarity of the tracks is the same. In musical terms, we expect the different versions of a track to share common characteristics, for example, in terms of structure or sequential pitch. We could say that, inside a clique, there is no outlier tracks (*eg* an electronic remix of an original acoustic slow).

[A_3] The cliques are considered equiprobable in the following sense: the probability that a query issued by a user matches a clique is the same for all queries and all cliques. While this may not be correct for one given experimentation, it is a valuable assumption when no prior information is available. This is also a way to guarantee a fair evaluation for all the cliques or queries, and therefore we will assume an equal probability for all the cliques and queries in the following.
It is obvious however that, for a given application, where some information about the type of queries sent by a user are available, for example because the user has been profiled, we could take into account that knowledge in some way. This results in an evaluation procedure that would be user specific.

3.2.2.1 Definition of the loss

Single clique loss

Let $q \in \mathcal{O}$ be a query inside a clique $c(q)$ that contains $\#(q)$ tracks. q is compared to all tracks in a database $\mathcal{D}$ using a similarity estimator $\hat{s}$. The estimator considers some tracks in $\mathcal{D}$ similar and some others dissimilar, which allows to build a confusion matrix. One can thus compute the number of TP, TN, FP and FN for each query q.

The definition of the loss rate is context dependent. Using the SMR definition, one considers that there is a loss for a query if no other versions of that query have been identified. Knowing that a match is found if $TP > 0$, the query is considered lost if $TP = 0$.

One can thus consider the loss for a query as a random variable loss$(q, c(q))$ whose outcomes are *True,* if no other versions are found for the query q, or *False*, if at least one version is found for the query q. That random variable is resulting from a set of similarity tests between the query q and the remaining elements of the clique $c(q)$:

$$\text{loss}(q, c(q)) = \bigwedge_{k \in c(q)} (\hat{s}(q,k) = 0 | s(q,k) = 1), \tag{3.3}$$

where $\bigwedge$ denotes the logical AND operator.

The loss for a query is *True* if each similarity test is *True,* and *False* otherwise. We can thus evaluate the loss for a given query as the probability that loss$(q, c(q)) = 1$ for all possible queries q inside the same clique:

$$p\left(\text{loss}(q, c(q) = 1\right) = \prod_{k \in c(q)} p\left(\hat{s}(q,k) = 0 \,|\, s(q,k) = 1\right), \tag{3.4}$$

where $\prod$ denotes the product operator.

One can convert the logical AND to a product of probabilities because, from assumption $[A_1]$, the decisions taken by $\hat{s}$ for a given query are independent of the tracks to which the query is compared. Indeed, knowing that $\hat{s}(q, k_j) = 0$ does not help to determine whether $\hat{s}(q, k_i) = 0$ or 1. We can also reasonably assume that, according to the diversity of the tracks inside a clique, we have

$$p\left(\hat{s}(q, k_i) = 0\right) = p\left(\hat{s}(q, k_j) = 0\right) \; \forall k_i, k_j \in c(q). \tag{3.5}$$

Therefore, combining Equations 3.4 and 3.5 leads to

$$p\left(\text{loss}(q, c(q)) = 1\right) = p\left(\hat{s}(q,k) = 0 | s(q,k) = 1\right)^{\#(q)}. \tag{3.6}$$

It follows that the loss of a query within the same clique C_i depends only on the cardinality of the clique, with respect to a similarity estimator $\hat{s}$. Following Equation 3.6, the loss rate of a clique C_i can be defined as follows

$$\text{loss}(C_i) = p\left(\hat{s}(q,k) = 0 | s(q,k) = 1\right)^{\#(C_i)}. \tag{3.7}$$

One can observe that $p[\hat{s}(q,k) = 0|s(q,k) = 1]$ from Equation 3.7 corresponds to the expression of the FNR. Indeed, if we compare any query q to all tracks k inside a clique C_i, the FNR is the proportion of pairs (q,k) considered dissimilar by $\hat{s}$. This proportion can be evaluated for a single query q, but we can also evaluate it for a large set of tracks, which gives the following relation

$$\text{FNR}_i = p\left(\hat{s}(q,k) = 0 | s(q,k) = 1\right)^{\#(C_i)}, \tag{3.8}$$

which leads to

$$\text{loss}(C_i) = \text{FNR}_i. \tag{3.9}$$

Multiple clique loss

The evaluation for a dataset composed of several cliques is tricky, because the sizes of the cliques can be different or the distribution of the queries over the cliques might not match the size of the cliques. For example, we could have a clique that is over-represented in the dataset (by a large amount or tracks in that clique) or we could have queries that never challenge a given clique. Both problems could introduce biases in the evaluation process and need to be addressed.

For the question about the queries that challenge all the queries, we have assumed, by $[A_3]$, that all cliques are equiprobable. This assumption is necessary, as we do not have any knowledge about the prior probability that a query q matches a clique. Such a probability depends on the use case of the system, which might vary from one user to another. Although it is possible to compute such a prior from a specific database, there is no way of generalizing that to all possible use cases. We therefore assume that this prior probability is uniform for all cliques.

The other problem relates to the cardinality of the cliques. To avoid any bias due the cardinality of a clique, we fundamentally have two ways to proceed:

1. We normalize the cliques losses with respect to their cardinality, for example by duplicating or synthesizing some samples (that process is called "data augmentation" in the machine learning community). This makes the evaluation universal as it simulates an evaluation database with equal sized cliques, thus considering that the evaluation database is *canonical.* Such a normalization allows to compare the performance of systems evaluated on different databases, because we can evaluate the system for each clique and then just

average the loss over the cliques. Mathematically, the loss for a database $\mathcal{D}$ containing N cliques is then given by

$$\text{loss}(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\text{loss}(C_i) \in [0,1]. \tag{3.10}$$

As the cliques are equiprobable, we have, by combining Equations 3.9 and 3.10:

$$\begin{aligned}\text{loss}(\mathcal{D}) &= \frac{1}{N}\sum_{i=1}^{N}\text{loss}(C_i) && (3.11)\\ &= \frac{1}{N}\sum_{i=1}^{N}\text{FNR}_i. && (3.12)\end{aligned}$$

The loss is thus a function of the False Negative Rate (FNR) that can be written as $\varphi_{\mathcal{D}}(\text{FNR})$.

2. In some cases, one is interested in an evaluation that takes into account the size of each clique, as evaluations are often unbalenced in practice. We explain that specific use case in Chapter 4. In that case, the evaluation databases cannot be considered canonical, and one needs to weight the loss of each clique by the probability of occurrence of that clique, denoted by $p\left(C_i\right)$, which can be approximated by $p\left(C_i\right) = \#(C_i)/M$, where M is the total number of tracks of the database. In that case, the global loss is computed as

$$\text{loss} = \sum_{i=1}^{N} p\left(C_i\right)\ \text{FNR}_i. \tag{3.13}$$

In that case, the evaluation takes into account the number of tracks per clique explicitly, which means that the evaluation process becomes dataset dependent.

In the remainder of this chapter, we address the normalized case, defined by Equation 3.12.

3.2.2.2 Definition of the pruning

The pruning is the proportion, over all possible queries, of irrelevant samples from $\mathcal{D}$ that are pruned.

For any query q , one can write

$$\text{prune}(q) = p\left(\hat{s}(q,\,t) = 0\right)\ \forall t \in \mathcal{D}, \tag{3.14}$$

which corresponds to the proportion of tracks that are considered dissimilar and therefore are rejected. The prune only depends only on the decision threshold (see Section 3.2.1) and thus does not depend on the input query q. We make the assumption that

$$\text{prune}(q_i) \backsimeq \text{prune}(q_j)\,\forall q_i, q_j \in \mathcal{D}. \tag{3.15}$$

Indeed, inside a clique, because we made the assumption that the diversity is the same (Assumption $[A_2]$), the musical similarity is considered the same for all tracks of the clique. It follows that the remaining tracks of the cliques do not contribute to the prune. For different cliques, if we consider a well balanced database, the proportion of rejected tracks will be constant. The global prune value, for all queries, can therefore be obtained as follows:

$$\begin{aligned} \text{prune} &= \sum_{i=1}^{M} p\left(q_i\right)\, \text{prune}\left(q_i\right) \\ &= \frac{1}{M}\sum_{i=1}^{M} \text{prune}\left(q_i\right). \end{aligned} \tag{3.16}$$

The global prune can also be developed as follows. One can note that it is directly related to the TNR.

$$\begin{aligned} \text{prune} &= p\left(\hat{s}=0\right) \\ &= \underbrace{p\left(\hat{s}=0 \,|\, s=0\right)}_{\text{TNR}}\, \underbrace{p\left(s=0\right)}_{\simeq 1} + p\left(\hat{s}=0 \,|\, s=1\right)\, \underbrace{p\left(s=1\right)}_{\simeq 0} \\ &\simeq \text{TNR}. \end{aligned} \tag{3.17}$$

The last approximation is explained by the fact that in a retrieval system, the probability of having a match for a given query in a large database is close to 0. On the other hand, the probability of finding mismatches is close to 1.

3.2.2.3 Prune-Loss curves

The performance of a single similarity estimator can be represented in the PL space as a single point. Figure 3.4 shows the performance of four possible *fictional* systems in the PL space. One can observe different performances for the various classifiers. The optimal point in Figure 3.4 is shown in the lower right corner. This is the point that corresponds to a similarity estimator that would achieve an optimal pruning rate close to 1 and a loss of 0, which means it could identify all queries immediately. In that case, the first track identified as a match is always correct. At this time, there are no CSI systems able to reach that optimal performance , and it is likely that the optimal performance will never be reached. Existing systems correspond to

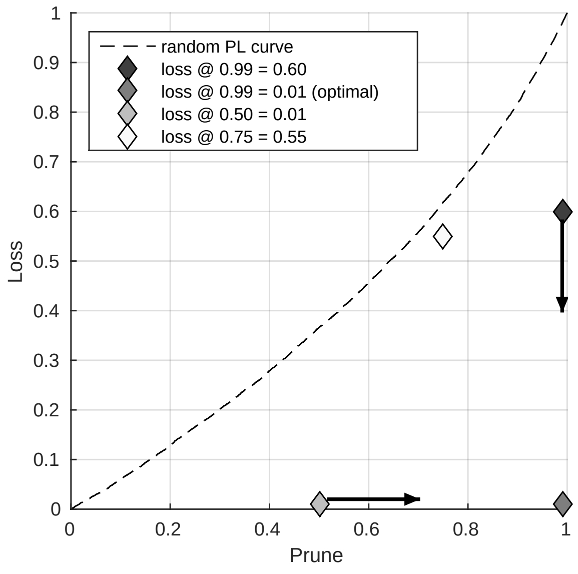


Figure 3.4: Single similarity estimators in the PL space.

estimators whose performance is a trade-off between the prune and loss rates. For example, the dark point on the right of the figure shows a system that achieves a loss of 0.6 at a pruning rate of 0.99. This means that $99\,\%$ of the music collection is thrown away, while $1\,\%$ is kept as the final search set for a given query. In other words, the probability of identifying a query while discarding $99\,\%$ of the database is $40\,\%$. The arrow pointing down shows the direction to get to the optimal point and thus progressively reduces the loss at a fixed prune threshold.

On the other hand, the light gray point shows a system with relatively low pruning and loss rates. Such a system does not reduce the size of the search set by much, but has a low loss rate, which ensures that most queries will still be identified. Here, the probability of identifying a track while reducing the search set by $50\,\%$ is $99\,\%$, as the loss rate at the same point corresponds to $1\,\%$. In a database of $100,000$ tracks, 1000 tracks will be lost if only the first $50,000$ tracks are kept. On the other hand, $99,000$ songs will still be identifiable if we drop half of the database. While the performance for such a point is not exceptional (the identified tracks will lie somewhere between position 1 and $50,000$), the computational power needed can be reduced by two, which can be interesting in some cases.

This shows that depending on the context, one might need more a system such as the darker point, or maybe a system such as the light gray point. As a comparison, the figure shows a white point that is close to a system with a random performance. Such system is not interesting as its performance are low compared to other possibilities.

Plotting pruning and loss values on a curve with a variation of the threshold of a similarity estimator allows us to visualize its effectiveness at all possible pruning rates as shown in

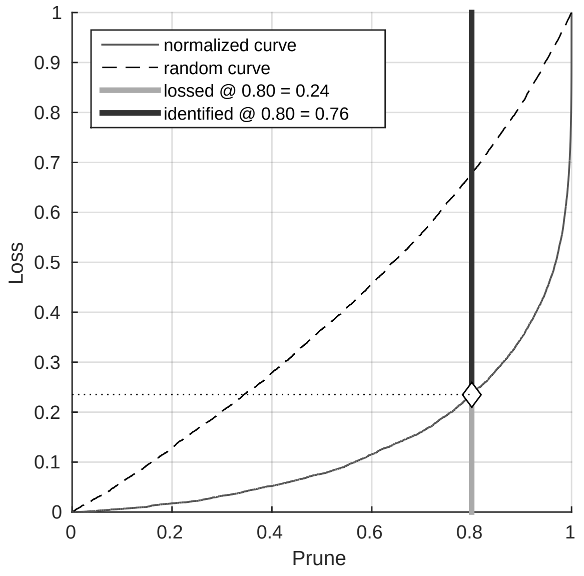


Figure 3.5: Example of a Prune-Loss curve with a CSI system implementing the 2D-FTM feature, as presented by Bertin-Mahieux [13] (see Chapter 2). One can observe the performance in terms of pruning and loss rates of a random system, as well as the performance of the 2D-FTM system. Note that the PL curve immediately allows to read the proportion of tracks unidentified at a given prune threshold (gray line), as well as the proportion of identified tracks (black line), which corresponds to $1 - loss(threshold)$.

Figure 3.5. For a complete SMR system, we want to achieve a high pruning rate and to minimize the loss rate. However, for a pruning method considered in the first step of the SMR system, achieving a high pruning rate is not the principal matter of concern since an expert is assumed to be able to finish the work, but a very low loss rate is necessary.

The PL curve of a random CSI system. To have a basis of comparison, we study the behavior of a random classifier Υ in the PL space. Υ does not take any information about its input into account to take its decision. Its behavior is parameterized by a constant $\alpha = p\left(\Upsilon\left(x,\, o\right) = F\right)$, which relates to the TNR. Therefore, $1 - \alpha = p\left(\Upsilon\left(x,\, o\right) = T\right)$, which relates to the TPR. As a consequence, its performance is given by (TPR, TNR) $= (1 - \alpha,\, \alpha)$. Equations 3.12 and 3.17 imply that $\mathrm{prune} = \alpha$ and $\mathrm{loss} = \varphi_{\mathcal{D}}\left(\alpha\right)$. It follows that determining the performance of a random system helps to determine the signature $\varphi_{\mathcal{D}}\left(\cdot\right)$ of the used dataset. Indeed, from Equation 3.12, one can observe that in the case of a random curve, the loss only depends on the size of the cliques. The random curve can therefore be seen as a characterization of the dataset used for evaluating a CSI system. The PL curve of a random system is expected to be linear if and only if $\mathcal{D}$ contains only one sample per clique, since in that case $\varphi_{\mathcal{D}}\left(\alpha\right) = \alpha$. Figure 3.6 shows several random PL curves for fictional datasets with cliques of fixed sizes.

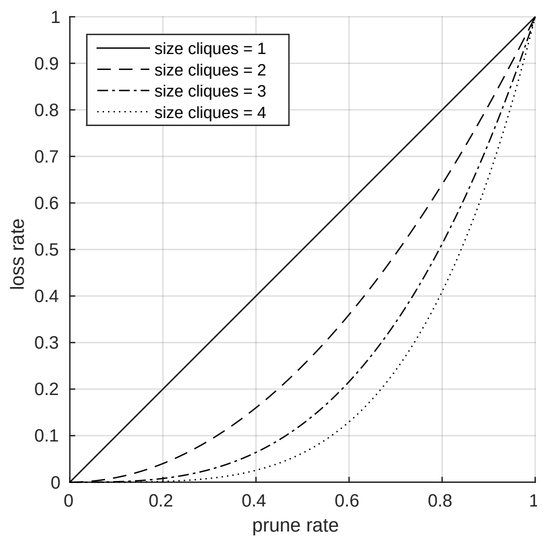


Figure 3.6: Random PL curves for fictional datasets with fixed cliques sizes of 1, 2, 3 and 4 respectively. One can observe that the curve is linear only when the cliques contain one element. As the size of the cliques increases, the random curve bends down.

One can observe that the curve is linear when the cliques contain one element. As the sizes of the cliques increases, the random curve bends down accordingly.

The non-linear behavior of the random curve can be understood intuitively. For a given query (*eg* with 4 versions), the loss will be 1 only if all 4 versions are not identified at a given threshold. However, if one version is identified at a given threshold, the loss stays at 0. If at a higher threshold, at least another version is identified, the loss for that query will stay set to 0, while the pruning rate will increase. Generalizing that to the whole dataset, one obtains a curve similar to the ones showed in Figure 3.6.

Figure 3.5 shows an example of a PL curve for a CSI system implementing the 2D-FTM feature described in Chapter 2. One can read on the curve the performance in terms of pruning and loss of the random system, as well as the 2D-FTM CSI system. In the context of pruning, the curve should be as close as possible to the horizontal axis. The dark (identified) and light (lossed) gray straight markers indicate respectively the proportion of tracks unidentified at the selected threshold, and the proportion of identified tracks. As this curve is obtained with Equation 3.12 which gives an equal weight to all cliques, we call this curve the *Normalized Prune Loss Curve* (NPLC).

3.2.3 Relationship between the ROC, PL and PR spaces

Equations 3.17 and 3.12 imply that $(\text{prune}, \text{loss}) = (\text{TNR}, \varphi_{\mathcal{D}}(\text{FNR}))$. $\varphi_{\mathcal{D}}(\cdot)$ is a strictly monotonic function, and can therefore be inverted. As a consequence, there is a one-to-one bijection between the ROC space and our PL space. It is quite remarkable that the signature $\varphi_{\mathcal{D}}(\cdot)$ of the evaluation database unequivocally defines the mapping between ROC and PL. The signature of an evaluation database simply corresponds to the list of the sizes of the cliques forming the dataset. As the performance represented in the PL space depends on the characteristics of the dataset used for the evaluation, one should always represent $\varphi_{\mathcal{D}}(\cdot)$ (*i.e.* the performance of a random classifier) on all PL plots[1]. Note that, since there is also a relationship between the ROC and PR spaces [28], by transitivity the bijection also applies to PL and PR spaces.

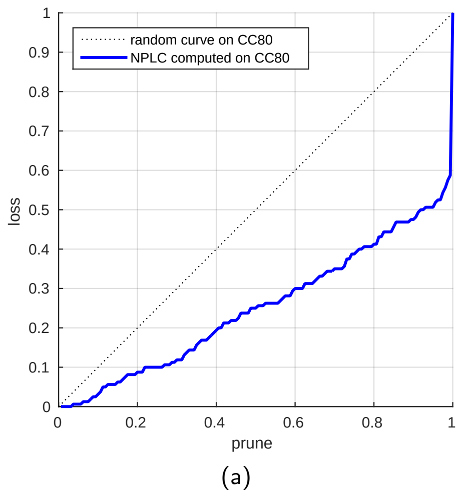
The bijection between ROC and PL gives the possibility to predict the performance of an algorithm with another dataset, by switching from PL to ROC space, and then back to the PL space using the signature $\varphi_{\mathcal{D}'}(\cdot)$ of a different database $\mathcal{D}'$. This is due to the fact that the ROC curve is independent of the structure of the dataset, as it represents the performance in terms of true positive and false positive *rates*. Moreover, it allows to apply in the PL space any known technique of the ROC space such as the combination of classifiers [4, 55, 59, 60].

Estimating the performance of bigger datasets

One straight application of the bijection between PL and ROC evaluation spaces is that it is possible to estimate the performance of a system, initially evaluated on a small database, on a much bigger database. As the bijection between PL and ROC only relies on the structure of the cliques contained in the dataset, it is easy to obtain a ROC curve from an original experimental PL curve. Then, the inverse operation can be done, by specifying a different database signature to predict an estimate of the performance of the system on a different database.

We computed the experimental NPLC on a small dataset containing 80 cliques of size 2, the Cover80 dataset. More details about the structure of this dataset will be given in the next chapter. From the NPLC, we computed a ROC curve, using equations 3.12 and 3.17. Then, from the ROC curve, we applied the same equations to compute a new PL curve, by specifying a different signature, corresponding to a subset of the Second Hand Song dataset (SHSD), containing approximately 13,000 songs. The latter will be detailed in the next chapter. It is quite remarkable that the estimated curve on the SHSD is quite close to the experimental curve computed on the same dataset. Figures 3.7 and 3.8 show the resulting curves for the

[1] Alternatively, one could also consider normalizing the performance with respect to the characteristics of the evaluation dataset. However, this approach should be avoided since it implies reporting the results as a ROC curve instead of a PL curve, with a loss of interpretability.



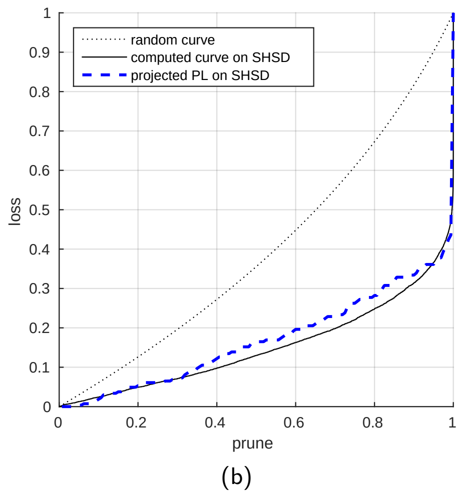


Figure 3.7: Predicting the performance of QMax, evaluated on the Cover80 dataset (figure (a)), on a bigger collection containing approximately 13,000 songs (figure (b)). The signature of the bigger dataset is simply a list of the sizes of each clique in the bigger collection. The predicted curve matches the experimental curve, actually computed on the bigger dataset.

QMax [92] and 2D-FTM [13] systems (see Chapter 5). In both cases, the estimated PL curve matches the experimental curve.

It should be noted that the computational time needed to compute a PL curve for QMax on a large audio collection is high, as it makes use of slow alignment algorithms. Using the bijection to estimate the curve without actually computing it constitutes an enormous gain of time as the execution of the algorithm is much faster on the Cover80 dataset. One should not forget however that this is an estimation of the performance, and although the shape of the curve gives a general information about the behavior of the method, it is not accurate enough to be taken as a result. One application could be to predict the performance of a system on a bigger database in order to determine is it looks promising or not, and if further research should be pursued with the considered system.

3.3 Conclusion

In this chapter, we have defined a relaxation of the CSI problem, so that it is considered effective when at least one matching track is found, with respect to a query. We showed that *Single Match Retrieval* problems are a particular case of content-based retrieval systems, and we provided a way of evaluating such problems, with respect to the notion of pruning and loss. As standard evaluation spaces such as ROC and PR are not appropriate for that kind of problems, we introduced our own evaluation space, the Prune Loss space, which allows to quickly identify

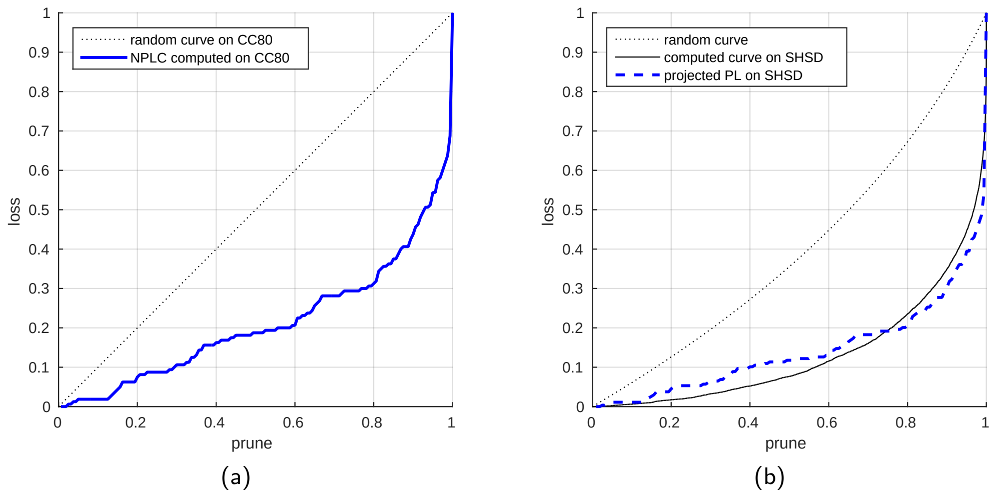


Figure 3.8: Predicting the performance of 2D-FTM, evaluated on the Cover80 dataset (figure (a)), on a bigger collection containing approximately 13,000 songs (figure (b)). The signature of the bigger dataset is simply a list of the sizes of each clique in the bigger collection. The predicted curve matches the experimental curve, actually computed on the bigger dataset.

the optimal threshold, given a performance curve. This allows to easily compare systems to select the best one for some specific needs.

In a PL curve, the region of interest is a small triangle in the bottom of the space, as shown in Figure 3.3c. As the goal is to ease the task of a further expert, we can tolerate a lower pruning rate as long as the loss rate is reasonable (depending on the final application). However, as the loss rate grows, there are more chances to drop every possible match for a query, rendering the system unusable. So, PL curves allow to tune a retrieval system to find the best trade-off between pruning and loss rates. Note that when the prior of the positive class is small, the PL space gives the most readable information as the PR space does not allow to read any interesting information. As shown in Figure 3.3b, the PR space only shows a zoom of the right hand side of the PL space which is not easily interpretable in a pruning context. Indeed, the zone we want to achieve in the PL space corresponds to the triangle in the bottom part of the space (Figure 3.3c) where the loss rate is minimum for a given pruning rate. This zone is almost invisible in the PR space. ROC space also does not give interesting information, especially when using small priors. Indeed, in the ROC space (Figure 3.3a), we usually want to go in the direction of the top left corner to maximize the TPR and minimize the FPR. However the zone of interest in the PL space correspond to a different area in the ROC space, which is less intuitive. So in the case of small priors, the more readable space is the PL one. Note that for SMR problems, positive class priors are often small since there are not many correspondences in a large database for a given query.

CHAPTER 4

Evaluating the Performance on an Unbalanced Database

The theoretical framework described in the previous chapter considers an evaluation where each clique is assigned an equal weight, as shown by Equation 3.12. It therefore normalizes the loss of each clique by its size, thus simulating an evaluation database with equal sized cliques, and rendering the evaluation independent of any specific database. The evaluation procedure thus considers that the database is canonical. This is necessary for comparing research works on CSI, as many authors dealing with that field evaluate their systems on different datasets, often not representative of the existing music. The Normalized Prune Loss (NPL) space thus ensures that evaluations on smaller datasets are comparable, and not biased by the structure of the database.

Nevertheless, assigning an equal weight to the cliques has one drawback: it does not reflect the reality of commercial databases, as it gives an equal weight of $1/\#(C_i)$ to each track of each clique C_i, where $\#(C_i)$ is the size of the clique. However, some traditional or popular songs are often more covered than other less contemporary tracks. In large commercial databases, such as the ones used by Spotify, or Apple Music, the databases usually contain *unbalenced* cliques, with a different number of tracks per clique. An evaluation based on the NPL space attributes the same weight to the cliques, thus considering that the dataset is canonical. However, a commercial database is unbalenced, with respect to a canonical dataset, and the evaluation should be normalized differently in order to avoid a bias in the performance curve.

As explained in Chapter 3, one straightforward solution is to assign different weights to each clique, with respect to the size of the cliques. This corresponds to assigning a unique weight to all the tracks of the database, rather than for the cliques. A clique containing 10 songs would have a weight of $10/N$, where N is the size of the database, instead of $1/N_{Cliques}$ in the NPL space, where $N_{Cliques}$ is the number of cliques in the database. Thus the probability of matching a song of such a clique is increased, compared to a smaller clique. For instance, consider a clique corresponding to the song *Georgia on my Mind*, by *Hoagy Carmichael*. This song has been covered at least 372 times according to the Second Hand Songs website [1]. In a large database, the probability that a query matches that clique would be naturally increased, compared to a song that has only been covered twice. One can thus interpret the number of covers of a song as a measure of popularity of the song, and therefore a measure of predictability of the probability of covering this song.

In this chapter, we consider an adaptation of the NPL space, such that it becomes compatible with such large databases. We modify the space, in particular the expression of the loss, so that it assigns a unique weight to each track, thus giving different weights to different cliques. We show that there is a relationship with evaluations based on the positions of the tracks in an ordered list of results, as mostly done in the literature. We name that modified space the *Ranked Prune Loss* space (RPL) and quantify the difference with the NPL space. We further show that the RPL space is compatible with a subset of standard metrics used in the literature, which makes it easily interpretable. Finally, we show that the bijection between the RPL and ROC spaces still exists, thus allowing to predict the performance of a system on a bigger database.

4.1 Evaluation based on ranks

Most evaluations of existing CSI systems are based on ranking. A set of queries is usually compared to some reference collection of songs and metrics are computed based on the position (the ranks) of the relevant tracks in the returned ordered similarity ranking. Each resulting track is thus mapped to an integer rank in the range $[1, \ldots, N] \cap \mathbb{Z}$ where N is the size of the collection, and $\mathbb{Z}$ is the set of integers. The metrics computed on the ranks therefore produce a performance score specific to a selected database. The performance therefore depends on the structure and the size of the used databases. It follows that, in order to perform a correct evaluation, a database should contains tracks as representative as possible of the existing music. Several databases have been used for diverse MIR tasks, including CSI. Most of them do not contain more than a few hundreds samples, which might be not representative enough of the existing music, and thus produce an evaluation biased towards the database.

[1]`https://secondhandsongs.com/work/2968`

Dataset	Songs / Samples	Audio Available
123Classical [6]	123	Yes
Cover80 [35]	160	Yes
YouTube Cover [94]	350	No
RWC [52]	465	Yes
Bimbot et al. [16]	500	No
CAL-500 [108]	502	No
Billboard [19]	1,000	No
GTZAN genre [109]	1,000	Yes
MusiCLEF [79]	1,355	Yes
SALAMI [98]	1,400	No
USPOP [10]	8,752	No
CAL-10K [104]	10,870	No
Magnatagatune [67]	25,863	Yes
Second Hand Song (SHS) [14]	18,196	No
Million Song Dataset (MSD)[14]	1,000,000	No

Table 4.1: List of databases available for MIR tasks. In particular, entries in bold are used for the evaluation of CSI systems.

4.1.1 Evaluation databases

Because MIR lacks large-scale databases to evaluate and compare algorithms, many researchers have used personal databases to evaluate their MIR system. In his thesis, Bertin-Mahieux [11] provides a survey of existing databases for several MIR tasks. Table 4.1 gives an overview of some databases, including specific ones appropriate for CSI (in bold). We focus mainly on the databases usable for CSI.

The 123Classical dataset [6] was originally proposed by Bello et al. [6] for measuring the structural similarity between audio recordings. It only contains 123 samples, which seems quite small for a representative evaluation of a CSI system. The Cover80 dataset [35] has been used in several CSI systems [3, 105] and has the advantage to provide free audio data for 160 songs, which is small. Several experiments from this thesis have been ran on that dataset (see Chapter 6). The Youtube cover dataset was introduced by Silva et al. [94] for the evaluation of their CSI system based on shapelets. For legal reasons, they could not provide audio data but a description of the database is available on their website[2]. The other databases of the table have been used for other MIR tasks, such as structural analysis (Bimbot, SALAMI), music tagging (CAL-500) and genre recognition (GTZAN). The MIREX framework [30, 31] allows to evaluate CSI systems on a standard database. Participants have to submit their code and human operators run the evaluations on a hidden dataset. Unfortunately, for CSI, MIREX

[2]`https://sites.google.com/site/ismir2015shapelets/`, accessed February 24, 2017.

deals with a small database containing at most a few hundreds or a few thousand songs. Two databases are used for CSI within MIREX: the first one contains 1000 tracks of pop music, in which there are 30 cover songs for which 11 versions exist. The second one is a classical database, the MAZURKA dataset [3], which contains 539 tracks

All these datasets have one common point: they lack a large amount of audio tracks, thus not allowing large-scale evaluations of MIR tasks. In addition, most of these datasets are not suited for the task of CSI. In 2011, a new dataset, the Million Song Dataset (MSD)[4] was released by Bertin-Mahieux et al. [14] to answer the need of large-scale evaluation database for MIR tasks. It does not provide audio data, but a set of pre-computed features for 1 million songs. The features are computed and provided by the EchoNest[5]. Many features are pre-computed for a wide range of MIR applications, including automatic genre detection, or automatic tagging of music. For cover songs tasks, it provides the Second Hand Song dataset (SHSD), a subset of tracks organized in cover groups called *cliques*. The subset contains $18,196$ tracks organized in $5,854$ cliques. This dataset is currently the largest one available for CSI, despite the fact that it does not provide audio. This is the dataset we chose for our experiments. Our evaluation setup for all our experiments is detailed in Chapter 5.

4.1.2 Evaluation metrics

Most metrics used in CSI systems are derived from the positions of the identified tracks in a ranked list of similarity results. For example, one can quantify the number of queries for which a different version has been found immediately (at the first position) using the so called Top-1 metric. Other popular scores such as the mean reciprocal rank, the mean rank or the mean average precision are also computed from the index position of the matches in a ranked list of results. These metrics have been extensively used in the CSI literature [3, 23, 42, 62, 63]. Manning et al. [70] gives a good overview of such metrics. In this section, we explain each of these metrics briefly, so that the reader understands their meaning. The next section covers the relationship with the RPL space.

4.1.2.1 Mean Reciprocal Rank

The Mean Reciprocal Rank (MRR) is a metric used in retrieval systems that quantifies the performance of a system that produces a list of responses to a query ordered by probability (or score) of correctness. The reciprocal rank for a given query is the multiplicative inverse of the

[3]`http://www.mazurka.org.uk/`, accessed February 28, 2017.
[4]`https://labrosa.ee.columbia.edu/millionsong/`, accessed February 28, 2017.
[5]`http://the.echonest.com/`, accessed February 28, 2017.

rank of the *first correct match*. The MRR is the average of the reciprocal ranks of results for all considered queries in a set Q; it is given by

$$\mathsf{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathsf{rank}_i}, \tag{4.1}$$

where rank_i corresponds to the rank of the first identified match for a query $q_i \in Q$. From Equation 4.1, one can observe that the closest to 1 rank_i is, the higher the MRR will be. It therefore gives more weight to the smallest ranks, which makes sense as the MRR corresponds to the harmonic mean of the ranks. One can interpret the MRR as the proportion of queries for which a correct match is found *close* to the top-1.

4.1.2.2 Mean Rank

For a given query, the position in the ordered ranking of the first matching track is considered. The Mean Rank (MR) of a CSI system is computed as the average of these positions for all queries in a dataset. If k_i is the position of the first matching track for the query $q_i \in Q$, where Q is the set of queries, then the mean rank is given by

$$\mathsf{MR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\mathsf{rank}_i. \tag{4.2}$$

This metric has been used in most research publications related to CSI. We demonstrate later that it is actually difficult to interpret.

4.1.2.3 Mean Average Precision

In a retrieval system, the Mean Average Precision (MAP) may be used to assess the performance if more than one track should be retrieved for a given query. In a ranked system, sets of retrieved tracks are given by the *top-k* retrieved tracks, that is, the first k retrieved tracks in a returned ordered list of results. For each set, precision and recall values can be computed and plotted on a precision-recall curve (see Section 3.2.1). MAP is a single measure of quality across recall levels. For a *single query*, the average precision (AP) is the average of the precision value obtained for the set of top-k documents existing after each relevant document is retrieved. This value is then averaged over all queries to build the MAP. If the set of relevant documents for a query $q_i \in Q$ is $d_1, \dots, d_m$ and R_{ik} is the set of ranked results from the top result until document d_k is found, then the MAP is defined by

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{m_i} \sum_{k=1}^{m_j} \text{precision}(R_{ik}). \tag{4.3}$$

When a track is not retrieved at all, the precision in the equation is set to 0. Note that the MAP is a generalized case of the MRR. If we consider only one match per query, MAP = MRR.

The MAP is not the preferred metric for SMR systems as it considers *all* matches to a given query. In contrast, the SMR definition considers that a query is identified if at least *one match* is found. In that case, the MRR seems more interesting, if one is interested in near top-1 performance. For comparison, we compute the MAP in all our experiments, as it has been used to evaluate most existing CSI systems.

4.1.2.4 Top-K identified tracks

The top-k value corresponds to the number of queries for which at least one corresponding match is found in the k first returned results. This gives an indication of the proportion of identified queries at a given threshold. This metric is commonly used to characterize the performance of a CSI system. We demonstrate later that the top-k is related to the prune rate, and thus can be interpreted on a PL curve.

4.2 Ranked Prune Loss space

It turns out that we can adapt the PL space so that it can be built on top of a ranking. We demonstrate that the prune rate can be related to a rank, and thus that standard metrics can be derived straight from the curve. This leads to a different evaluation space, named the *Ranked Prune Loss* (RPL) space, that produces RPL curves, optimized for a specific database.

4.2.1 Definition

In Chapter 3, we considered that the evaluation should account for changes in cliques sizes. We therefore normalized the loss by the number of cliques to assign the same weight to all cliques during the evaluation. This resulted in the loss expression given by Equation 3.12.

In practice, to compute a loss from a ranking, one has to count the number of queries for which a match has been found above the decision threshold. In contrast to the expression of the loss rate in the NPL space, the number of identified queries is not normalized with respect to the size of the cliques in which the queries belong. Indeed, one can refer to the definition of the metrics (Section 4.1.2) to understand that each query has the same weight. It follows that the loss computed from a ranking is different from a normalized loss and will produce a different curve. We call that curve the *Ranked Prune Loss Curve* (RPLC) as it is a PL curve computed from the ranks.

To be compatible with the notion of rank, we must weight each clique according to its cardinality. The weight thus corresponds to the probability of occurence of a clique within the evaluation dataset, and can be computed as $p(C_i) = \frac{\#(C_i)}{M}$, where C_i is a clique and M is the number of tracks in the database. This leads to an alternative definition of the loss, for datasets with cliques of unequal sizes, where each clique is given a weight defined by its probability of occurrence:

$$\begin{aligned} \text{loss}(\mathcal{D}) &= \sum_{i=1}^{N} p(C_i)\, p(\hat{s}=0|s=1)^{\#(C_i)} \qquad (4.4) \\ &= \sum_{i=1}^{N} p(C_i)\ \text{FNR}_i. \qquad (4.5) \end{aligned}$$

As this definition of the loss considers the probability of occurrence of each clique, an evaluation based on this expression will reflect the performance of a CSI system on a non-canonical database used for evaluation.

The prune rate does not change from the NPL space. Its expression does not change as it still corresponds to the proportion of tracks that are rejected. Note however that there is a direct equivalence between the prune rate and the corresponding threshold rank, as given by Equation 4.6:

$$\text{prune} = 1 - \frac{\text{rank}}{N}, \qquad (4.6)$$

where N corresponds to the number of tracks of the database.

4.2.2 RPL curves

The adapted formulation of the loss, defined by Equation 4.4, leads to an RPL curve that is remarkably close to the experimental curve, evaluated on a dataset, as drawn in Figure 4.1. The figure shows PL curves for two systems, respectively studied by Bertin-Mahieux et al. [13] and Serra et al. [92]. The curves are evaluated on a subset of the SHS dataset, containing $12,856$ songs. The theoretical curves (in blue), have been computed straight from Equation 4.4, by considering the sizes of the cliques in the dataset. The experimental curves (in red) have been experimentally computed, by running the evaluation procedure on the entire database. The

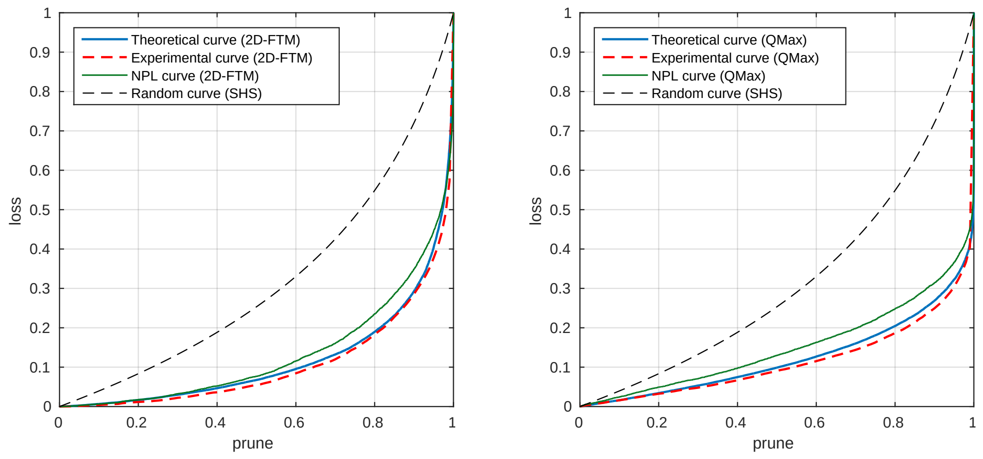


Figure 4.1: The loss expression (Equation 4.4) produces theoretical curves that remarkably match the experimental curves. The blue curves are the theoretical curves computed for the 2D-FTM (left) and QMax (right) systems, while the red curves are the experimental curves for the same systems. For comparison, the NPLC for both systems is also displayed (in green). One can observe that the RPLC is lower, and thus more optimal than the NPLC.

experimental curve is computed based on the proportion of tracks identified at each pruning threshold. As a baseline comparison, we also display the corresponding NPL curves (in green). One can observe that as expected, the RPL curves are more optimistic than the NPL curves. Keep in mind however that the RPL curves are related to the used dataset only.

4.2.3 Differences with the NPL space

We evaluated 10 systems on a subset of the SHS dataset (containing $12,856$ songs), and computed their NPL and RPL curves, in order to quantify the difference between them. Our evaluation setup and the individual performance of each system is detailed in the next chapter. Figure 4.2 shows both RPL and NPL curves for all systems and the area between both curves. We consider this area as a distance measure between both curves. One can observe that both RPL and NPL curves follow a similar general shape. However, the RPL curves produce better performance in all cases. Differences between both curves are quantified by the surface between them, with a mean area $\mu = 0.04$ and a standard deviation $\sigma = 0.01$.

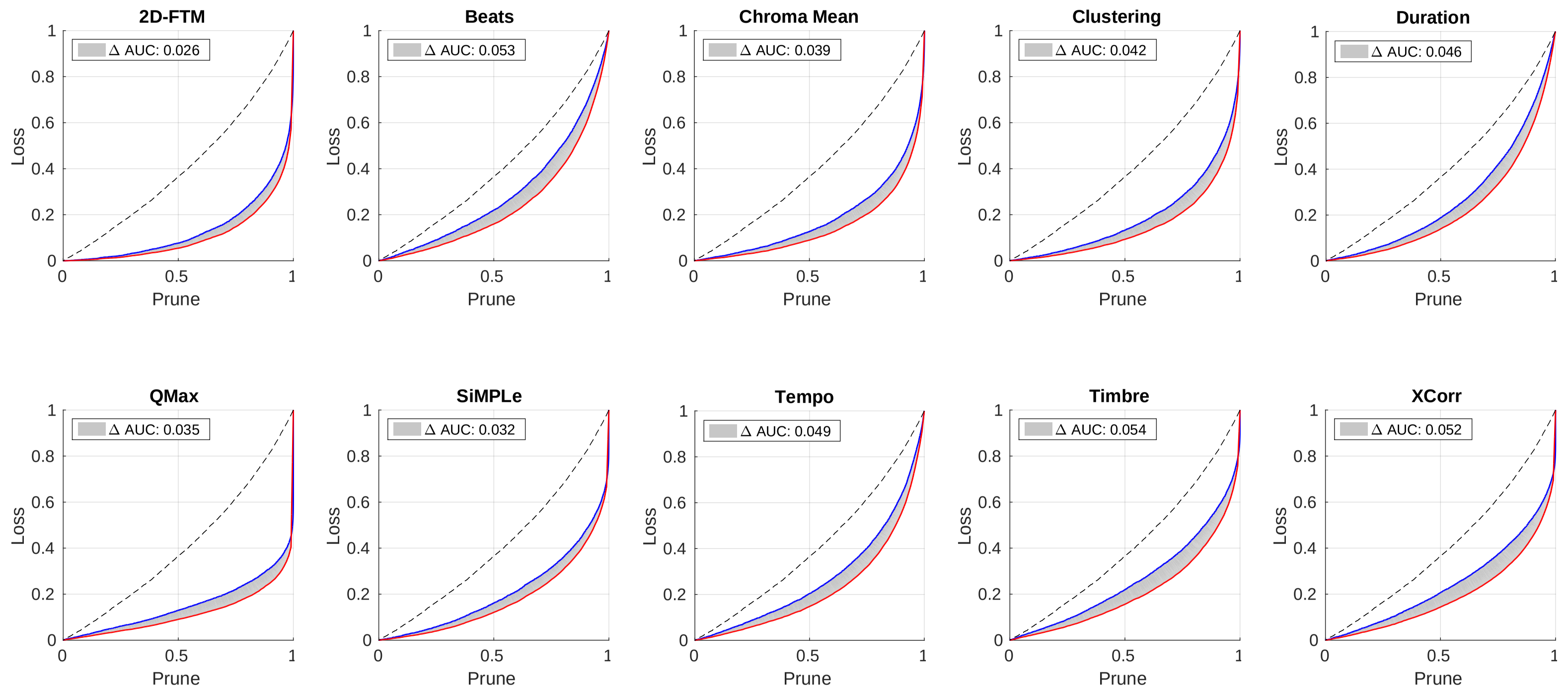


Figure 4.2: Differences between NPLC (in blue) and RPLC (in red) for 10 selected CSI systems. The curves have been computed on a collection of 12,856 songs. The shaded zones represent the area between both curves. The average difference in terms of surface is $\mu = 0.04$ and the standard deviation is given by $\sigma = 0.01$.

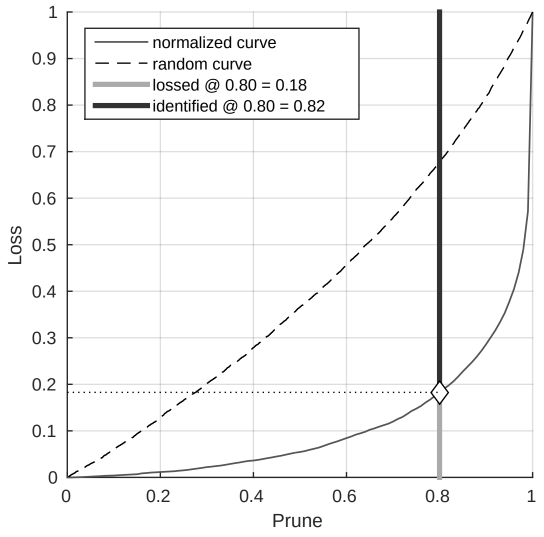


Figure 4.3: Ranked PL curve for the 2D-FTM system evaluated on the SHSD. The RPL space allows one to read the proportion of identified tracks at a given top-k threshold. Here, $prune = 0.80$, which corresponds to the $top - 2,571$ for a database of size $N = 12,856$. The prune segment on the right of the vertical marker corresponds to the $2,571$ tracks that will be returned. In dark gray, one can read the proportion of identified tracks (or lost in light gray), 0.82, which corresponds to $10,542$ tracks identified at that threshold.

4.3 Reading metrics on an RPLC

4.3.1 Relation with the Top-K

The PL space (NPL or RPL) displays the performance of a retrieval system in terms of a pruning rate and a loss rate. As the pruning rate corresponds to the proportion of tracks that are rejected from the final subset, it is related to the notion of ranks. If the considered dataset contains $N = 12,856$ tracks, a pruning rate of $80\,\%$ corresponds to returning the $top - 2,571$, which represents the $20\,\%$ remaining tracks returned. Thus, there is a direct equivalence between the pruning rate and the corresponding top-k subset, which is given by Equation 4.6.

The loss at a specific prune rate τ (the threshold) gives the proportion of tracks for which no covers are identified at all. Consequently, the proportion of tracks for which at least another version is *identified* (at the same threshold τ) is given by the top-k metric:

$$top_k = 1 - loss_\tau, \tag{4.7}$$

where $k = (1 - prune_\tau)\,N$. Figure 4.3 illustrates how to read the top-k value. In Figure 4.3, $82\,\%$ of the tracks are identified, which corresponds to $10,542$ tracks.

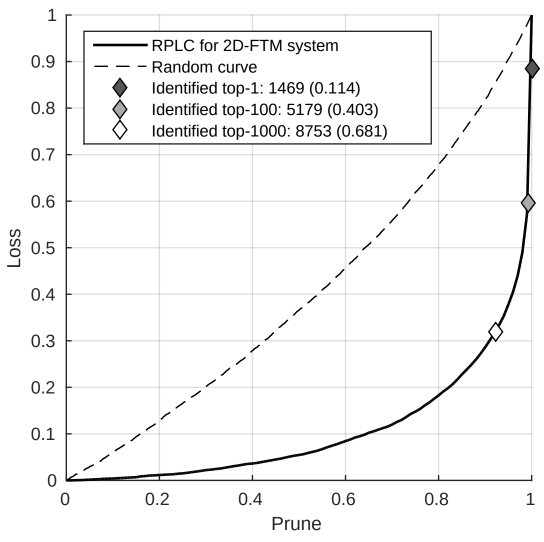


Figure 4.4: The Top-1 point is always represented on a RPL curve close to the right axis. The vertical space above the point gives the proportion of identified tracks at this threshold. The vertical space below the point gives the loss.

Special case for the top-1

For a given query, the subset corresponding to the top-1 contains a single track, which is considered the most similar to the query. In the RPL space, the top-1 metric will indicate the proportion of queries that find a match at the first position. The prune corresponding to the top-1 is given by $1 - \frac{1}{N}$. Following our previous example, if $N = 12,856$, we have $prune_{top-1} = 0.9999$. The top-1 point will therefore be almost at the end of the prune axis, lying somewhere on the right loss axis, as pictured in Figure 4.4. The figure also displays the top-100 and top-1000 points as a comparison. One can observe how close these points are to the right axis.

4.3.2 Relationship with Mean Rank

The RPL space allows one to directly estimate the Mean Rank (MR) from the curve. It turns out that the area under the RPLC approximates the MR.

Following Equation 4.6, the prune can be seen as the complementary of a rank, as pruning corresponds to rejecting a subset of the entire collection, as shown in Figure 4.5. The prune / rank axis can be scaled, whereas the loss axis does not change. For simplicity, we will refer to the Rank-Loss (RL) curve when the prune axis is expressed in terms of ranks. One can write

$$\text{Loss(rank)} = \text{Loss}(N(1 - \text{prune})). \tag{4.8}$$

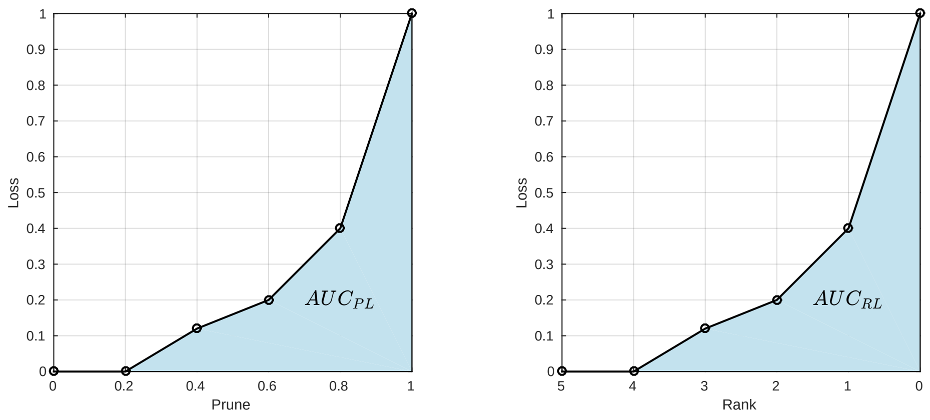


Figure 4.5: The prune rate can be interpreted as a rank, as pruning corresponds to keeping only a subset of the collection. In both cases, the area under the curve is the same. On the left, the one can observe the prune axis, and on the right, one can observe the rank axis for a dummy collection of 5 tracks.

From Equation 4.6, one can scale the area under the RL curve as follows:

$$AUC_{RL} = N.AUC_{PL}. \tag{4.9}$$

We can demonstrate that there is a relationship between the area under the curve and the MR metric (see Equation 4.2). Consider the example given in Figure 4.5, which represents the performance of a synthetic database containing 5 tracks. Each track of that collection is compared to the 4 remaining tracks. Therefore, the number of queries identified at rank 1 ($rank_1$ in Equation 4.2) is N_1. The corresponding proportion of tracks identified in the top-1 is N_1/N, as shown in Figure 4.6.

The same reasoning can be applied for rank 2. In that case, in Figure 4.6, $\frac{N_1+N_2}{N}$ gives the proportion of tracks identified in the top-2. This can be generalized to all points, with N_j tracks identified at $rank_j$. By definition, following Equation 4.2,

$$MR = \frac{1}{N}\sum_{j=1}^{N} rank_j = \frac{1}{N}\left(1.N_1 + 2.N_2 + \ldots + 5.N_5\right). \tag{4.10}$$

This sum can be represented graphically by the sum of the surfaces of all rectangles given by (R_i, R_i') in Figure 4.6 (in yellow and blue). Indeed, one can compute the surface of the R_i pair by

$$S(R_i, R_i') = \frac{N_i}{N}.i, \tag{4.11}$$

where i is the corresponding rank.

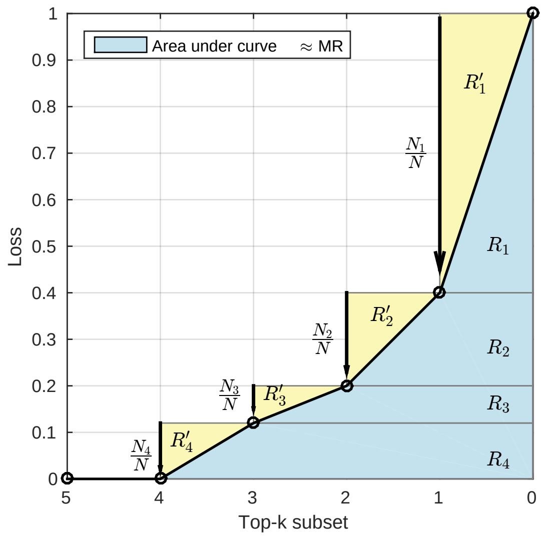


Figure 4.6: The area under the RPL curve corresponds to the Mean Rank metric.

Now, the blue area in Figure 4.6 corresponds to AUC_{RL}. Therefore, the MR is given by the sum of the AUC_{RL} and the area of R'_1, $R'_2, \dots, R'_N$ (in yellow in Figure 4.6). We have

$$S(R'_i) = \frac{N_i}{2N}. \tag{4.12}$$

The surface of the entire yellow area can be obtained by developing Equation 4.12:

$$S(R'_1, \dots, R'_N) = \frac{1}{2N}\sum_{i=1}^{N} N_i = \frac{1}{2}. \tag{4.13}$$

It follows that $MR = AUC_{RL} + 0.5$. As $AUC_{RL} = N.AUC_{PL}$, we have

$$MR = N.AUC_{PL} + 0.5. \tag{4.14}$$

So, back in the RPL space, the area under the curve, AUC_{PL} is given by

$$AUC_{PL} = \frac{MR - 0.5}{N}. \tag{4.15}$$

If the database contains many songs (N is big), one can approximate AUC_{PL} by $AUC_{PL} \approx \frac{MR}{N}$.

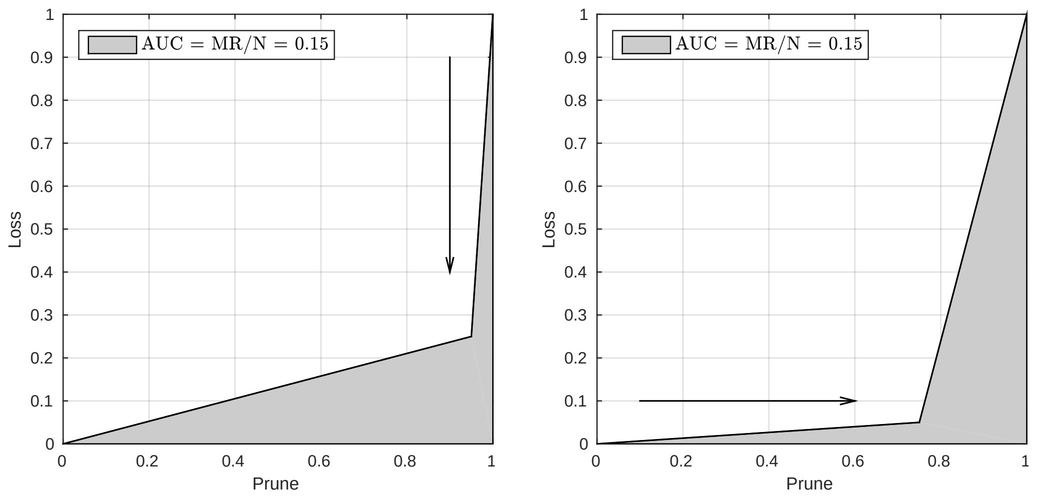


Figure 4.7: The MR corresponds to the area under the PL curve. However, due to the 2-dimensional space, the same area can correspond to different contexts. On the left, the direct identification rate (near top-1) is high, as the curve follows a steep descent following the right axis. On the right, many tracks can be pruned (up to 75%) with a minimal loss. In contrast the direct identification rate is low.

Interpretation of the MR

The MR metric is difficult to interpret as there are two degrees of freedom: the pruning rate and the loss rate. The MR can thus be directly related to the notions of prune and loss. The only think one can tell when considering the MR is that the lower it is, the deeper we lower the RPL curve. However, one cannot conclude anything about the zone that will be the most affected by the decrease of the area, as shown in Figure 4.7. The figure shows that the same area can correspond to two radically different situations. In the first case (left figure), the proportion of tracks identified near the top-1 is huge, while the pruning rate increases quickly. A CSI system producing such a curve would identify many tracks near top-1. On the other hand, the right figure identifies almost nothing near the top-1. In contrast, it shows a low loss rate up to 75% of pruning. A CSI system producing such a curve would be able to reduce the size of the collection while achieving a low loss rate, thus effectively returning a subset of 25% of the collection with the almost certainty that a match will be found in the remaining set.

4.3.3 Mean Reciprocal Rank and Mean Average Precision

Representing the MRR and the MAP on a RPL curve is not straightforward and there is no way of predicting their values from the curves. However, it is possible to draw some hypothesis about these metrics from the upper corner of the curve. As both metrics are related to the proportion of tracks identified in the top of the list of results, one can follow a simple rule of

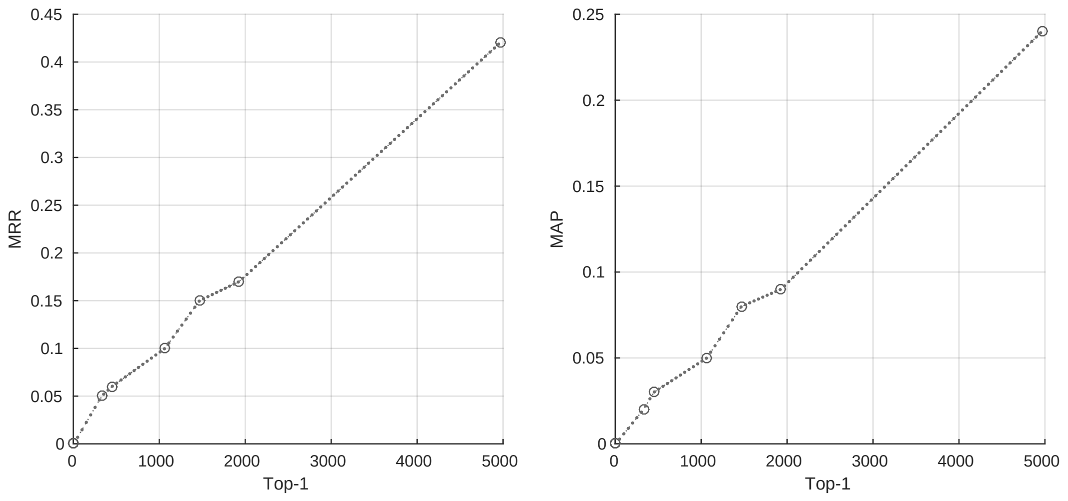


Figure 4.8: Interpretation of the MRR and MAP metrics. One can observe that both metrics are correlated with the top-1 metric, thus showing that an increase of the proportion of tracks identified in the top-1 produces an increased MRR and MAP.

thumb to predict such metrics. As the proportion of tracks identified at the top-1 increases, both the MRR and the MAP will increase. Therefore, *the steepest the RPL curve is in the upper right corner, the higher the MRR and MAP will be*. This simple rule gives an idea of the behavior of these two metrics straight from the curve. This can be observed in Figure 4.8, plotting experimental top-1 values against MRR and MAP values.

4.4 Predicting metrics on different databases

The expression of the loss in the RPL space allows a bijection with the ROC space, as in the NPL space. As shown by Equation 4.4, the loss can therefore be expressed in terms of FNR. The expression of the prune does not change, compared to the NPL space, and therefore, the prune can be expressed in terms of TNR (See Chapter 3). It follows that there is direct bijection between the RPL space and the ROC space, that only depends on the cardinalities of the cliques. It follows that similarly to the NPL space, the cardinality of the cliques define a *signature* of the dataset, which can be used to predict the performance of a system on a different database. Indeed, one can convert a RPL curve evaluated on a dataset D_1 to a ROC curve using the equations of the loss and the prune and the signature of D_1. Then, one can switch back to the RPL space, using the same equations with the signature of another dataset D_2.

As we now have a way of reading some metrics directly on the RPL curve, this technique can be used to estimate the values of the MR and the Top-K metric quite easily, as demonstrated

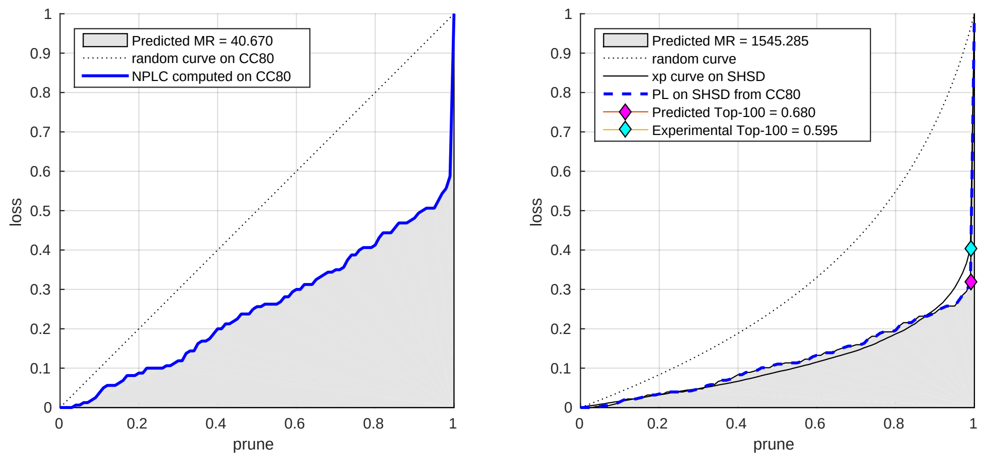


Figure 4.9: Using equations of the RPL space, one can predict the performance of a system on a different dataset. On the left, the QMax system is evaluated on the Cover80 dataset. On the right, we predicted the RPL curve on the SHS dataset. One can observe that the predicted curve is close to the experimental curve. The predicted MR is the area under the curve ($MR = 1,545$) and the predicted top-100 is 0.68 ($8,741$ as there are 12,856 tracks).

in Figure 4.9. The figure on the left is the RPL curve computed for the QMax system on the Cover80 dataset. The figure on the right is the predicted RPL curve evaluated on the SHS dataset. The prediction was computed using our theoretical framework. From the predicted curve, the estimated MR is $1,545.28$, compared to $1,466$ on the experimental curve, which is close. The estimated top-100 here is 0.68 which corresponds to $8,741.40$ (over $12,856$). The real value for the top-100 on the SHS is 0.60. On this particular example, estimated top-100 is a bit optimistic. Keep in mind however that this is a prediction that is computed without running any experiment. Considering that, the bijection can be a powerful tool to estimate the performance of CSI systems on bigger datasets without running any experiments.

4.5 Conclusion

In this chapter, we provided a modification of the NPL which allows to evaluate the performance of a CSI system on a non-canonical database. Instead of normalizing the loss with respect to the size of the cliques, the RPL spaces considers that cliques containing more tracks should be weighted accordingly. It therefore assigns a unique weight to all tracks of the database, rather than to all cliques. As the evaluated performance is dependent on the structure of the database, one should consider the RPL space only if the considered evaluation database is large enough and constitutes a representative sample of existing music.

We showed that the RPL space is compatible with standard metrics such as the mean rank and the top-k score. We also explained how to understand the behavior of the mean reciprocal rank and mean average precision directly from an RPL curve. Such an analysis of the RPL space allows one to interpret standard metrics used in the literature, which is not always straightforward. Indeed, metrics such as the MRR, MAP and MR are often presented together in the literature. However, it should be now clear that the MRR and the MAP quantify the performance differently, compared to the MR. While the MRR and MAP represent the performance in the upper-right side of a RPL curve, the MR is often correlated with the bottom part of the RPL space. The analysis of the performance of 10 systems, in the next chapter, will indeed indicate that a decrease of the MR is often correlated with a lowering of the PL curve at lower thresholds. The MR therefore is not valuable to represent the performance of the near top-1 zone of the PL curve.

Finally, we demonstrated in this chapter that the RPL space still allows to predict the performance of a system on another database. In particular, we are now able to estimate metrics directly from the curve, which allows to estimate the performance on any known database, without running any evaluation.

CHAPTER 5

Performance of Cover Song Identification Systems

The two previous chapters have introduced a complete evaluation framework for SMR problems. Taking advantage of that evaluation tool, we evaluate in this chapter 10 methods for CSI, including five state-of-the-art systems. As we run the evaluation on the SHSD, the choice of CSI methods to be evaluated is limited by the features provided in the dataset. Indeed, as pointed out in the previous chapter, no audio data is provided with the dataset, thus restricting CSI systems to the set of provided features. The dataset provides however chroma and timbre features for all songs, thus allowing to evaluate several existing systems.

In this chapter, we evaluate four methods based on chroma features, one method based on timbre features, and we include methods based on low-dimensional features such as the tempo, the duration, the number of beats and the average chroma features, to respectively account for rhythm and tonal information. As systems based on spectral features were proven to produce low performance [81], we chose to not evaluate them in this chapter.

We provide a comparison of the 10 selected systems on the SHSD, containing approximately $13,000$ songs. To our knowledge, this is the first evaluation of multiple CSI systems on a large evaluation database. For each selected system, we draw the performance on RPL and ROC curves as well as standard metrics such as the MR, the MRR and the MAP.

Results tend to demonstrate that chroma based system perform better than any other features. Furthermore, we show that systems based on alignment algorithms provide better results than systems based on summarized features. This tends to confirm the state-of-the-art analysis from Chapter 2.

This chapter is organized as follows. First, we describe each system selected for evaluation. We describe the way they process features and how a similarity score is computed. The next section continues with an explanation of our evaluation setup. We explain how we use the SHSD and how we split it in learning and evaluation sets. We further describe the algorithms chosen for supervised and unsupervised learning. In the next section, we continue with the performance review of each system, as well as a comparative analysis of the systems, with respects to several metrics. Finally, the last section concludes the chapter.

5.1 Systems description

For this study, we selected five methods from the CSI litterature based on chroma and timbre features. These systems were chosen based on their published performance, and, to a lesser extent, on their publication date. Four systems are based on chroma features, using different chromas post-processing algorithms and various distance and score computation methods. The selected systems based on chroma features were initially designed by Bertin-Mahieux et al. [13], Serra et al. [92], Silva et al. [95] and Ellis et al. [36]. We further selected a method based on timbre features rather than chroma, introduced by Tralie et al. [105] in 2015.

In addition to these state-of-the-art methods, we introduce our own systems based one low-dimensional features such as the tempo, the duration and the number of beats of the songs. Such systems make use of random forests to learn a similarity model and return probabilities of similarity. In addition, still based on machine learning, we experiment with a system based on the average chroma vectors of two songs. The last system is making use of a clustering algorithm to assign a fixed number of codewords to chroma features, using the so called bag-of-features [43] approach.

We implemented all systems described in this section in a C++ library that we used throughout our experiments. We based the implementation on the original papers, as presented by the authors. Except for a system based on timbre feature (Section 5.1.5) where we collaborated with the original author, all systems and features post-processing algorithms were coded from the original description.

5.1.1 2D Fourier transform magnitude (2D-FTM)

This method, provided by Bertin-Mahieux et al. [13] considers patches of 75 consecutive beat-synchronous chroma features, with a hop-size of 1. Figure 5.1 summarizes the process. For each patch of dimension 75×12 (chroma features are 12-dimensional, see Section 2.1.1), they compute the 2D Fourier transform magnitude coefficients. 2D-FFT patches are then

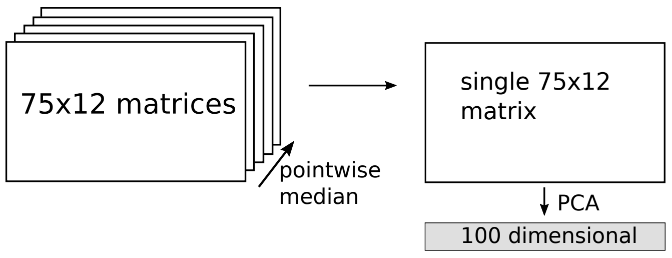


Figure 5.1: 75×12 beat-synchronous chroma patches are stacked and aggregated with point-wise median, then projected on a 100-dimensional PCA subspace.

stacked together and aggregated in a single 75×12 patch by computing a point-wise median. The resulting 900-dimensional patch is then projected on a 100-dimensional PCA subspace and the tracks are compared using the euclidean distance, which is a straightforward operation. In our own implementation, we observed that the cosine similarity produces better performance than euclidean distance. The performance reported here is computed with the cosine similarity.

5.1.2 Cross recurrence quantification (QMax)

In their work, Serra et al. [92] first extract chroma features from both songs to be compared (the query and a reference track) at regular and identical time points, and transpose one song to the key of the other by means of the Optimal Transposition Index (OTI) method [88]. Note that they consider the best two OTIs computed rather than using just one. Thus the whole comparison process is computed twice for a pair of songs, using both best OTIs, as suggested in [93]. Once the key transposition is done, they form representations of the songs by embedding consecutive chroma vectors in windows of fixed-length m with a hop-size τ. From both sequences of embedded chroma vectors, they build a cross-recurrence plot (CRP) as shown in Figure 5.2. To compute a similarity score, they build a cumulative matrix Q from the CRP. They consider constraints to take into account possible curvatures and time disruptions in the CRP while updating a similarity score, as shown in Figure 5.2. The final score is further normalized by the length of the candidate song, because document length is often correlated with relevance [93]. The QMax method is the most effective CSI system to date. Evaluated on the SHS dataset, it clearly outperforms other methods, as will be shown later. The drawback is that this representation requires a quadratic space in terms of memory, with respect to the length of the feature vectors used to represent the music. It follows that the method is at least quadratic in time complexity.

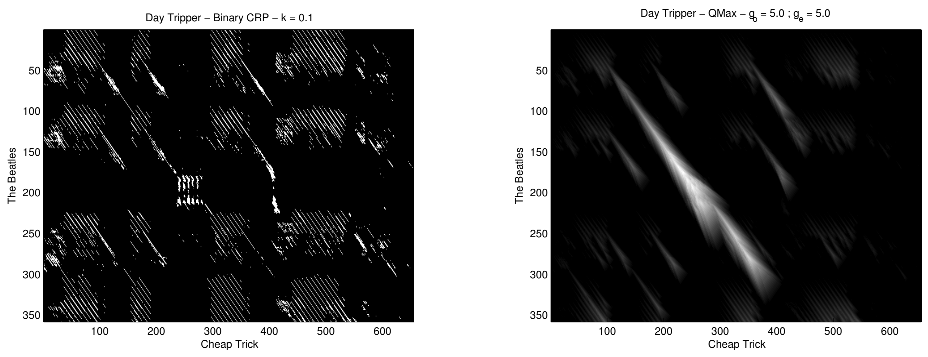


Figure 5.2: Left figure: CRP for the song *Day Tripper* as performed by The Beatles, taken as song X, versus a different song, taken as song Y. Song Y is a cover made by the group *Cheap Trick.* Right figure: Q matrix computed from the CRP with additional constraints to account for curvatures and time disruptions.

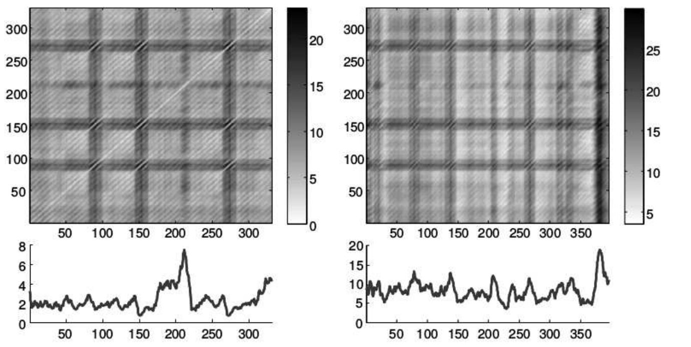


Figure 5.3: Similarity matrix for similar (left) and dissimilar (right) songs and their respective SiMPle. The figure is borrowed from Silva et al.'s paper [95].

5.1.3 Similarity matrix profile (SiMPle)

The SiMPle method, proposed by Silva et al. [95], exploits a new data structure, called the *matrix profile,* to speed up the process. It allows an efficient space representation and can take advantage of FFT based algorithms to be computed. The method works as follows. For a pair of tracks, sequences of chromas are extracted and embedded in windows, similarly to QMax. In our implementation, we use windows of 20 consecutive beat synchronous chromas, with a hop-size of 1. For each window of track A, we look for the closest window in track B, with respect to the L2 Euclidean distance. The distances between each window of A and the closest one in B are stored in a matrix profile, as depicted in Figure 5.3. The final similarity score is computed as the median of the matrix profile.

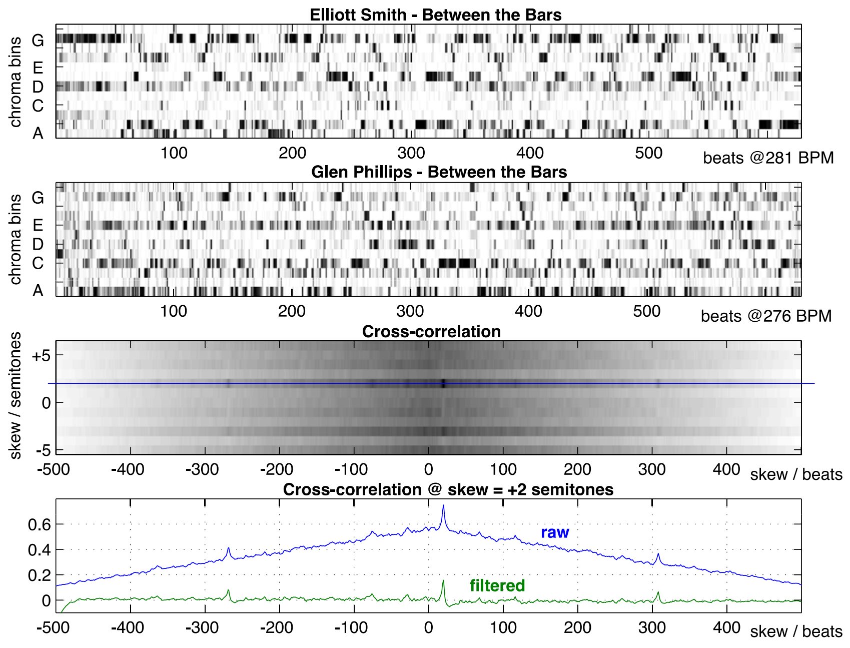


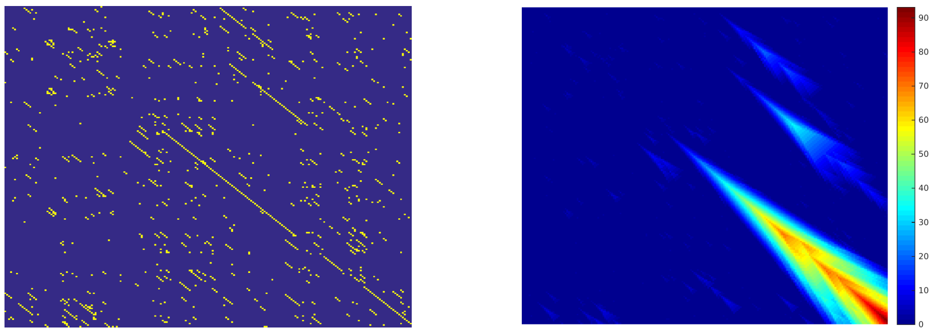
Figure 5.4: Cover song matching through 2D cross correlation. The first 2 panes are beat-synchronous chromas representation of two songs. The third pane is the 2D cross correlation for all chroma rotations. The best shift is +2 semitones as indicated by the blue line. So the last pane is the cross correlation at + 2 semitones plus the high pass filtering to emphasize peaks that results from the best alignment. This figure is borrowed from Ellis et al.'s paper [37].

5.1.4 Cross-correlation of chroma sequences (XCorr)

In that method, proposed by Ellis et al. [37], songs are represented by beat-synchronous chroma matrices. A beat tracker is first used to identify the beats time, and chroma features are extracted at each beat moment. This allows to have a tempo-independent representation of the music. Songs are compared by cross-correlating entire chroma-by-beat matrices (2-dimensional cross-correlation to identify the best tonal shift). Sharp peaks in the resulting signal indicate a good alignment between the tracks. The input chroma matrices are further high-pass filtered along time. The final score between two songs is computed as the reciprocal of the peak value of the cross-correlated signal. Figure 5.4 shows the steps involved in the cross-correlation score computation.

5.1.5 Timbral shape sequences (Timbre)

Tralie et al. [105] consider the use of timbre features rather than chroma features for cover song identification. They design features based on self-similarity matrices of MFCC coefficients and use the Smith-Waterman alignment algorithm to build a similarity score between two songs.



(a) Binarized cross similarity matrix. (b) Smith Waterman with local constraints (score = 93.1)

Figure 5.5: Binarized cross-similarity matrix based on SSMs of MFCC features and resulting application of the Smith-Waterman local alignment algorithm for two cover songs. The figures are borrowed from Tralie et al.'s paper [105].

Note that in contrast with other work considering MFCC features, they innovate by examining *relative* shape of the timbre coefficients. They demonstrate that using such features, cover song identification is still possible, even if the pitch is blurred and obscured. For this method, we collaborated with the authors to compute the exact same features as the ones they produced.

5.1.6 Quantization of chroma features (Clustering)

To take into account the harmonic information of the music, and to achieve fast retrieval, we introduced [80] an estimator based on the quantization of chroma features, rather than considering raw chroma features. Similar features have been used by Foster [42]. For each track, chroma features are mapped to specific codewords. A track is then represented by a histogram of the frequency of each codeword, known as *bag-of-features* representation [44]. Codewords are determined using an unsupervised K-Means clustering of $200,000$ beat-synchronous and unit-normalized chroma vectors. We evaluated the number of codewords in the range 25 to 100. Best performance was achieved with a clustering of 100 codewords. To account for key transpositions, we make use of the *Optimal Transposition Index* (OTI) [88] as it is a straightforward approach that has been used in other researches [2, 42, 87]. Figure 5.6 shows an example of song representation using quantized chroma features with 50 codewords. One can observe that both versions of the song "Little Liar" have a similar histogram, while a different song has a significantly different histogram.

The similarity between two bag-of-features representations is computed as the cosine similarity between both histograms. We evaluated the cosine similarity against Euclidean and Bhattacharyya distances, as well as a supervised learning based distance. However, best results

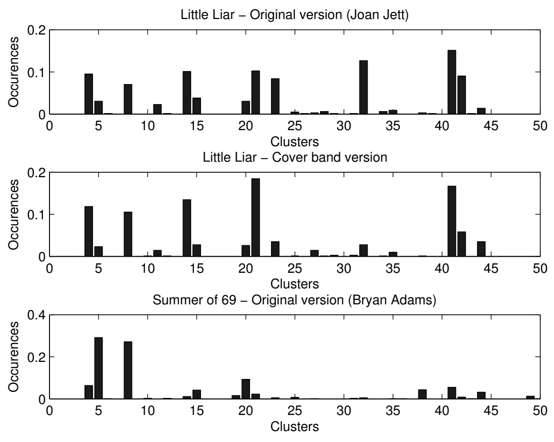


Figure 5.6: Quantization of chroma features into 50 codewords, thus producing a histogram of the number of occurrences of each codeword.

were achieved with the cosine similarity. Furthermore, the cosine similarity is fast to compute, especially when the vectors are normalized to unit norm, as it can be computed as a simple dot product.

5.1.7 Average chroma features

We introduce a method based only on the average chroma vectors of the songs to be compared. The average chroma vector of a song is computed as the bin-wise mean of all the chroma vectors across time. The resulting vector is a single 12-dimensional chroma. The average chroma vector can be related to the musical key of a song, and thus encode information relative to the key. One of its main usage if the transposition of a song to a common key, as suggested by Serra et al. [88].

Let us denote the average chroma vectors of two songs to be compared by V_1 and V_2. We begin by computing the two best OTI values, oti_1 and oti_2, to determine the two best alignments between both vectors. We apply the technique proposed by Serra [88]. We then rotate circularly V_1 by both oti_1 and oti_2 to be able to compute any distance between V_1 and V_2 with key invariance. If both tracks to be compared are similar, both shifts of V_1 by the preferred OTI values should render $V_1 \gg oti_i,\ i = 1, 2$ and V_2 similar, even if both songs are played in different keys. We further compute the average of both shifted V_1 vectors and V_2, as pictured in green in Figure 5.7. We compute angles θ_1 and θ_2, respectively between $(V_1 \gg oti_1, V_2)$ and $(V_1 \gg oti_2, V_2)$ as the cosine similarity. We finally compute the Bhattacharyya distances between the same pairs of vectors, which gives us β_1 and β_2. The Bhattacharyya distance is used to measure the similarity between two probability density functions. As our average

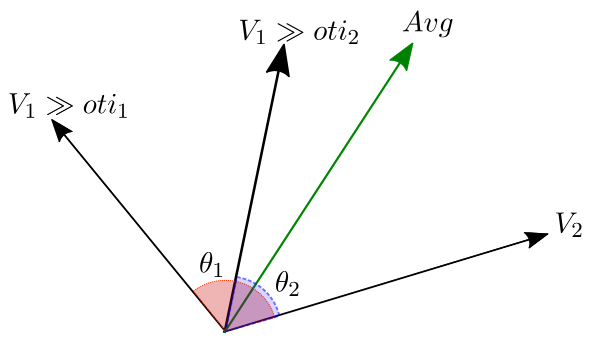


Figure 5.7: Graphical representation of the chroma mean features vector. V_1 and V_2 are the average chroma vectors of both tracks to be compared. oti_1 and oti_2 are the best computed transposition indices. $V_1 \gg oti_1$ is the rotation operator, to shift V_1 circularly by the oti_1 amount. Avg is the average vector of V_2 and both rotated V_1 vectors. Finally, θ_1 and θ_2 are respectively angles between $V_1 \gg oti_1$ and V_2 and $V_1 \gg oti_2$ and V_2.

vectors are normalized to unit-norm, we can use that distance to measure the similarity between average chroma vectors. The Bhattacharyya distance is computed as

$$D_B(p,q) = -\ln(BC(p,q)), \tag{5.1}$$

with $BC(p,q) = \sum_{x\in[1,12]} \sqrt{p(x)q(x)}$. We eventually end up with an 18-dimensional features vector:

$$\text{feature vector} = [oti_1,\ oti_2,\ \theta_1,\ \theta_2,\ Avg,\ \beta_1,\ \beta_2], \tag{5.2}$$

which we feed into a learning algorithm to build a similarity model. We specifically make use of the Extra-Trees [48] algorithm. The forest is learned using a learning set, as explained in Section 5.2.2. The model is learned with $1,000$ trees, with a maximum depth set to 15. The model is then used during evaluation to predict the probabilities for two tracks of being similar or dissimilar. The highest the probability is, the more similar the tracks are. Figure 5.8 shows the principle of a CSI system based on a learning algorithm.

5.1.8 Tempo, duration and beats

We implemented three additional systems based on machine learning, and relying on single dimensional features. We thus propose to compute a similarity of probability using the tempo, the duration and the number of beats of both songs to be considered. Although such musical features may not seem relevant for cover song identification, they can be used to eliminate outliers tracks, compared to the query. Indeed, songs with completely different tempi are not likely to be covers. The same yields for the duration and the number of beats.

Let us denote the feature considered for both input tracks as f_i, $i = 1, 2$ for track 1 and track 2.

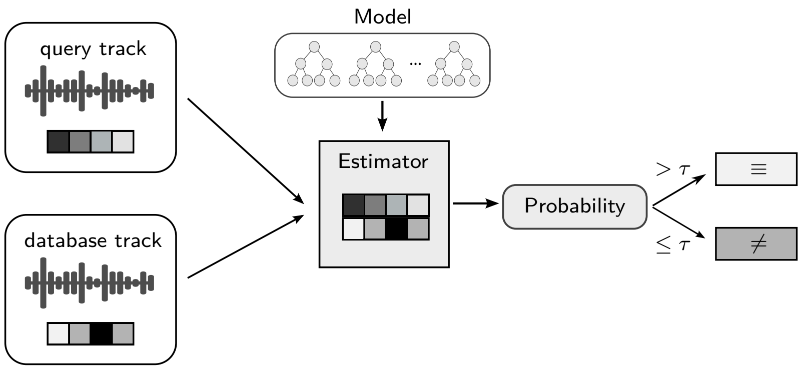


Figure 5.8: Principle of a similarity estimator based on a machine learning model. Both the query and a reference track are processed to build a feature vector composed with features extracted from both tracks (they grey-level squares). The feature vector is processed by a learned model (here with a random forest, or extra trees), which outputs a probability. The latter can be next thresholded to classify the tracks as similar ($\equiv$) or dissimilar ($\neq$).

f_i corresponds to one of the three proposed features, as the method is the same for all of them. We compute 6 non-linear modifications of both features, and feed them into a 6-dimensional feature vector:

$$\left[\min(f_1, f_2),\ \max(f_1, f_2),\ |f_1 - f_2|,\ \frac{|f_1 - f_2|}{f_1 + f_2},\ f_1 + f_2,\ \frac{\min(f_1, f_2)}{\max(f_1, f_2)}\right]. \tag{5.3}$$

As the input features are one-dimensional, such transformations are necessary for the learning algorithm to extract some meaningful information. As with the average chroma features, we use the Extra-Trees algorithm, which is capable of returning calibrated probability estimates. The higher the returned probability is, the most similar the tracks are.

5.2 Evaluation setup

5.2.1 Dataset

The Second Hand Song dataset (SHSD) is a subset of the Million Song Dataset [14] (MSD) organized in cliques, that is sets of tracks containing several versions of the same underlying musical piece. It contains $5,854$ cliques for a total of $18,196$ cover songs. To our knowledge, it is the biggest dataset available for cover songs retrieval. We split the dataset in a learning set (LS) containing $1,696$ cliques, and in a test set (TS), containing $4,128$ cliques, as it is usually done in most works based on this dataset [12, 13, 56].

The SHSD comes with a list of known duplicates[1]. That means that within one clique, two

[1]https://labrosa.ee.columbia.edu/millionsong/blog/11-3-15-921810-song-dataset-duplicates

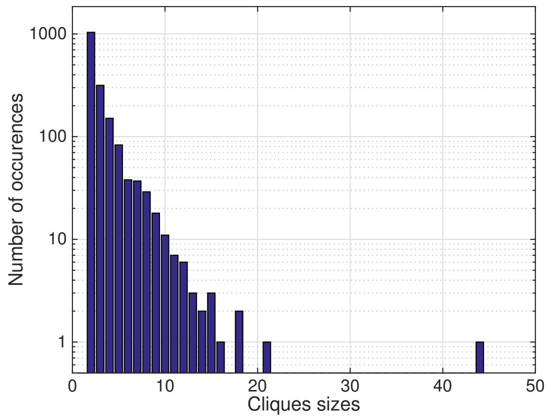


Figure 5.9: Histograms of the sizes of the cliques in the SHSD. Most cliques have a size 2 or 3. However some tracks contain much more versions. The mean clique size is 3.13 and the standard deviation is 2.36.

different track identifiers refer to exactly the same song. In that case, when evaluating a CSI system with the SHSD, the algorithm will most likely detect a true positive when comparing two duplicate songs (their content is supposed to be exactly the same). Such comparisons should not be made because it produces too optimistic results. Indeed, comparing a query with itself will return a near 100% similarity. We got rid of these duplicates to obtain a clean evaluation dataset. After executing the script, some cliques are left with one track only. We remove these cliques from the dataset, which reduces the total number of cliques. The original SHSD contains $5,854$ cliques, while the new pruned dataset contains $5,828$ cliques, losing 26 cliques in the process. The TS on which all evaluations are performed contains 12,856 tracks, instead of 12,960 tracks in the original set containing duplicates. Figure 5.9 displays the histogram of the sizes of the cliques in the SHSD without duplicates. The mean size of the cliques is 3.12 while its standard deviation is 2.36.

5.2.2 Learning algorithms

We introduce four methods based on supervised learning. These methods take into account single dimensional features such as the tempo, the duration, the number of beats and the average chroma vectors of the songs. For these methods, we learn a similarity model using random forests, specifically, Extra Trees [48]. We use our LS for learning (containing approximately 30% of the cliques of dataset, selected at random).

To avoid overfitting, we limit the depth of the trees in the forest, and the optimal depth is found by maximizing the area under the ROC curve. Note that to do that, we have to split

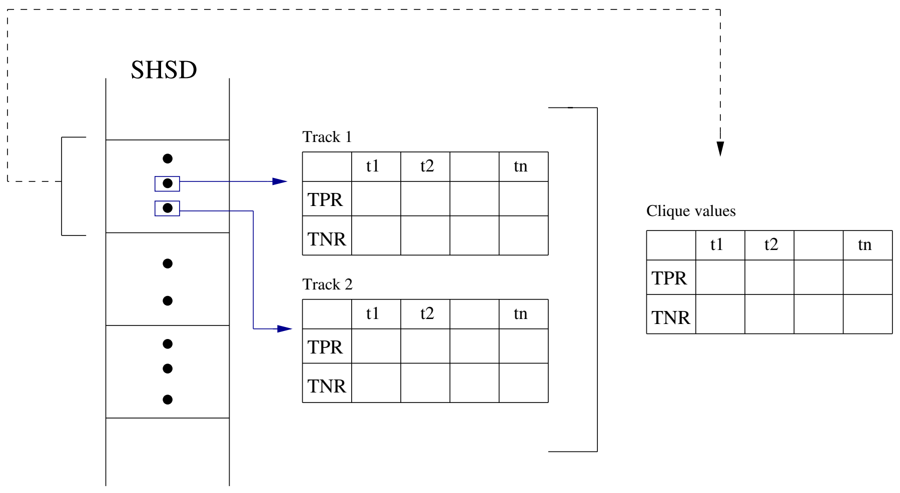


Figure 5.10: For each method, scores are computed at the track level, than at the clique level, and finally at the dataset level to match the evaluation framework defined in Chapter 3.

the LS to obtain a smaller Evaluation Set (ES) on which we tune the model parameters. All models are learned with 1000 trees with a maximal depth set to 15.

The implementation we chose is the one provided in the Scikit-Learn Python library[2]. This library provides a method for returning probabilities of belonging to each class, which we use as a similarity score between pairs of tracks. Features used for the learning process are computed for pairs of tracks and their corresponding label (similar, dissimilar). The features are system dependent and are described in the sections devoted to each system.

5.2.3 Evaluation procedure

In our evaluation setup, we use a single dataset of cliques for the queries, and the comparison tracks. As each clique contains at least two tracks, the evaluation procedure works as follows. Each track of the dataset is considered as a query, and compared to all other tracks in the TS, except the query itself ($12,855$ tracks). The comparison between the query and another tracks produces a similarity score, which might be a probability, depending on the studied system. Ordering the tracks from the highest score to the lowest one gives an *ordered ranking* for a given query. From the ordered ranking, one can compute the various performance curves (ROC, NPLC, RPLC) and metrics (MR, MRR, MAP, etc.).

Note that in order to match the SMR evaluation described in Chapter 3, one has to compute the performance at the track level, then normalizing by the number of tracks inside a clique in

[2]`http://scikit-learn.org/`, accessed March 2, 2017.

order to have a score at the clique level. Finally, the cliques scores can be normalized by the total number of cliques, thus producing a score at the dataset level, as depicted in Figure 5.10.

5.3 Results

Using the evaluation database and procedure described in the previous section, we display and discuss the performance of each system in the following. We first discuss the performance of each system, taken individually. Then, we provide a comparative study of the performance of the systems relative to the others. It should be noted that often, the performance we report for known systems is different from the results found in the literature. This is explained by the fact that we re-implemented all methods by ourselves, using the original paper as a reference, and sometimes some code when available. Furthermore, the only features we have at our disposal are the ones provided by the SHSD, which are known to produce inferior performance, compared to other chromas implementations [58].

5.3.1 Individual performance

5.3.1.1 Fourier transform magnitude (2D-FTM)

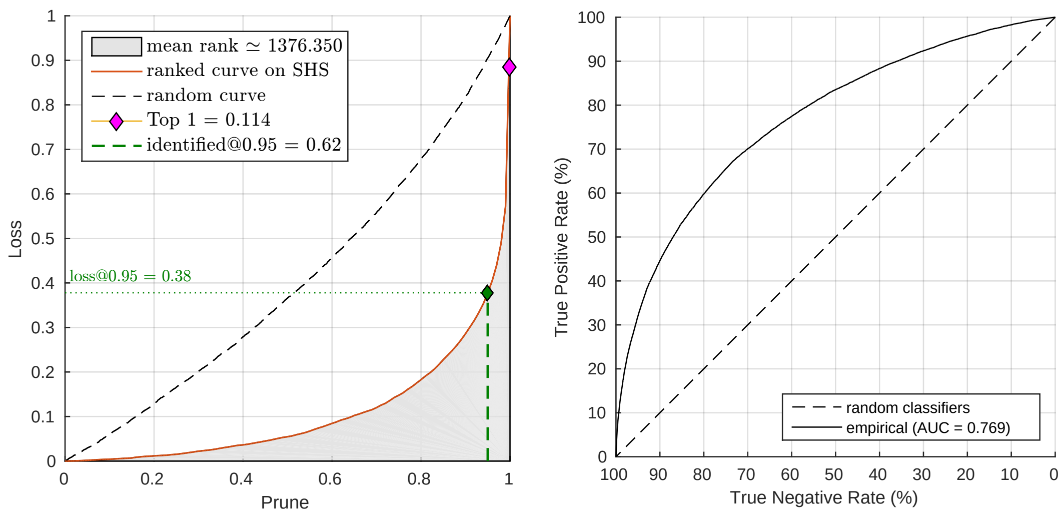


Figure 5.11: Performance curves for the 2D-FTM system.

MR	MRR	MAP	Top-1	Top-10	Top-100	ROC AUC
$1,359$	0.15	0.08	$1,469$	$2,869$	$5,179$	0.769

Table 5.1: Performance metrics on SHS Train Set (12,856 tracks) for the 2D-FTM system.

The 2D-FTM method is one of the fastest one, as it only needs to compute 100-dimensional euclidean distances, which is a fast operation. It is therefore scalable to a bigger database. Figure 5.11 shows the RPLC and the ROC curves for the system, evaluated on the TS described in Section 5.2.1. Table 5.1 quantifies the performance with respect to all metrics. From the table, one can observe that the ROC AUC (0.769) shows a suitable class separation [113]. From the RPLC, one can observe that the curve follows a vertical line in the upper right corner, thus showing a large proportion of queries identified near the top-1. $11.43\,\%$ of the queries are identified in the top-1, and $22.32\,\%$ and $40.28\,\%$ are respectively identified in the top-10 and top-100 returned tracks. In terms of MR, the performance is given by the area under the PL curve, which corresponds to 0.11 $(1,359$ tracks over $12,856)$. The MRR is 0.15, which is far away from the optimal MRR (1.0). However, as will be shown in the comparative analysis, the MRR can be considered as pretty good compared to other systems. The MAP in contrast, is pretty low with a value of 0.08. Keep in mind however that this is not the most relevant metric to our CSI problem, as we consider the problem of identifying at least a single match (See Chapter 3). Finally, the loss at a pruning rate of 0.95 (thus keeping the top-5%) is of 0.38, hence corresponding to an identification rate of 0.62.

5.3.1.2 Alignment of chroma sequences (QMax)

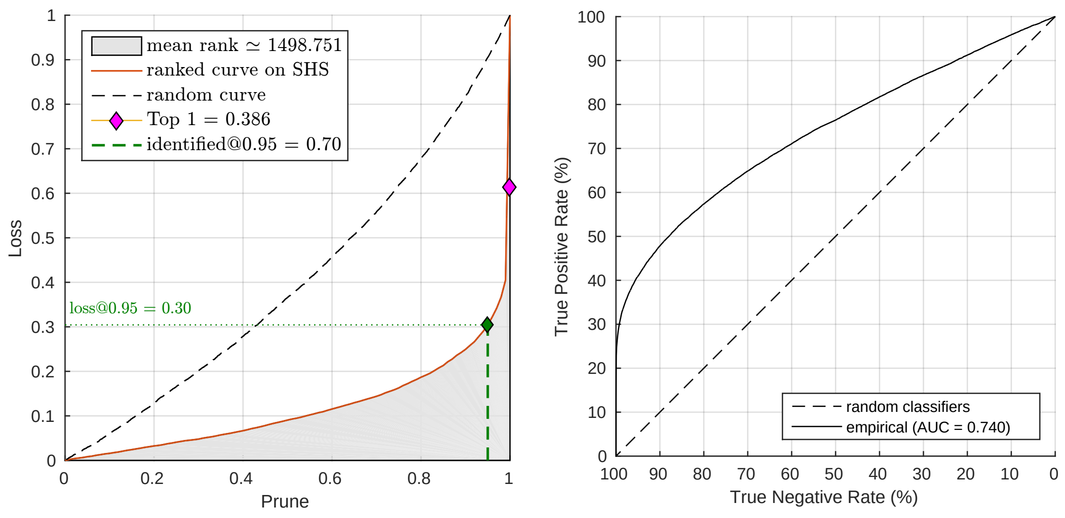


Figure 5.12: Performance curves of the QMax system.

MR	MRR	MAP	Top-1	Top-10	Top-100	ROC AUC
$1,466$	0.42	0.24	$4,965$	$6,188$	$7,505$	0.740

Table 5.2: Performance metrics on SHS Train Set $(12,856$ tracks) for the QMax system.

In contrast to the 2D-FTM, the QMax system is the slowest one. Figure 5.12 shows the performance curves for that system and Table 5.2 quantifies the performance. The QMax system is able to identify a lot of queries in the top-1, as can be observed on the PL curve. Indeed, the curve follows a vertical line in the upper right corner (respectively in the lower left corner on the ROC curve). $39\,\%$ of the queries $(4,965$ over $12,856)$ are identified directly in the Top-1, while $48\,\%$ are identified in the top-10. In terms of MRR, the system has the highest value with a MRR of 0.42. One can observe that this value is close to the $39\,\%$ identified in the top-1. The MAP is observed to be 0.24 which means that for approximately a quarter of the queries, all versions will be retrieved at the top of the returned list. The drawback of this method is its slow runtime performance. The loss at a pruning rate of 0.95 is if $30\,\%$, which means that $70\,\%$ of the queries are identified in the top-5% of the returned ranking.

5.3.1.3 Similarity Matrix Profile (Simple)

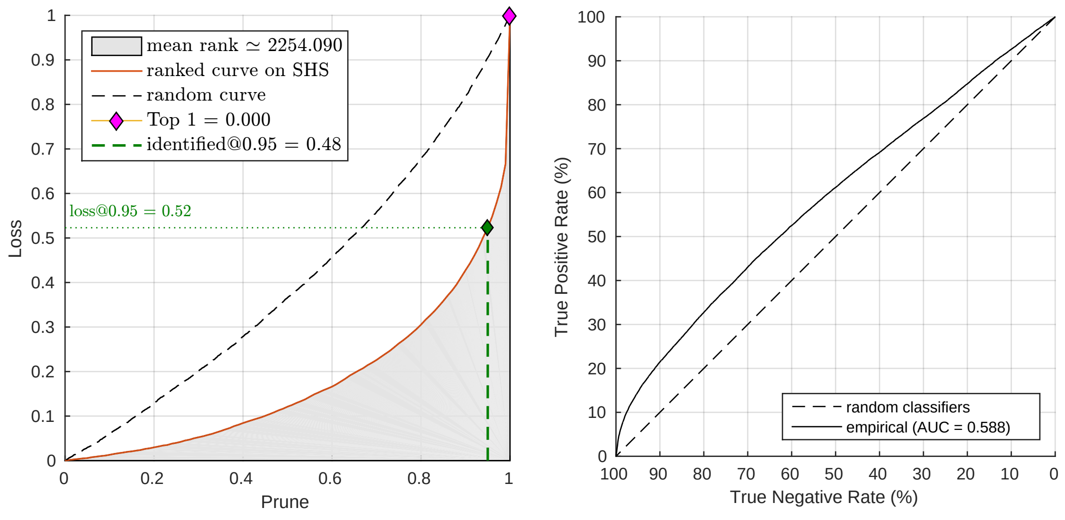


Figure 5.13: Performance curves for the SiMPle system.

MR	MRR	MAP	Top-1	Top-10	Top-100	ROC AUC
$2,240$	0.03	0.02	1	$2,493$	$4,074$	0.59

Table 5.3: Performance metrics for SiMPle.

The SiMPle system, introduced by Silva et al. [95] in 2016, attempts at considering structural invariance by using a new data structure called *Similarity Matrix Profile*. Figure 5.13 gives the performance curves in terms of RPLC and ROC. Table 5.3 quantifies the performance with respect to the metrics. From the RPLC, one can observe that the performance is rather poor, especially in terms of top-1 identification. Indeed, almost no queries are identified at the first position. In contrast, in terms of top-10, $19\,\%$ of the queries $(2,493)$ are identified, which is close to the 2D-FTM performance for the same metric. Due to the single query identified in top-1, the MRR is low, and so is the MAP. The MR is given by the area under the curve and corresponds to $2,243$ $(0.18$ when normalized). The loss rate at a pruning rate of 0.95 is of 0.52, which corresponds to $48\,\%$ of the queries identified in the top-5%. The ROC AUC is 0.59 which indicates that the separability between classes is not efficient [113]. The method is rather slow when implemented in a naive way. However, FFT based algorithms can be used for all-neighbor searches, which speeds up the identification process.

5.3.1.4 Cross-Correlation (XCorr)

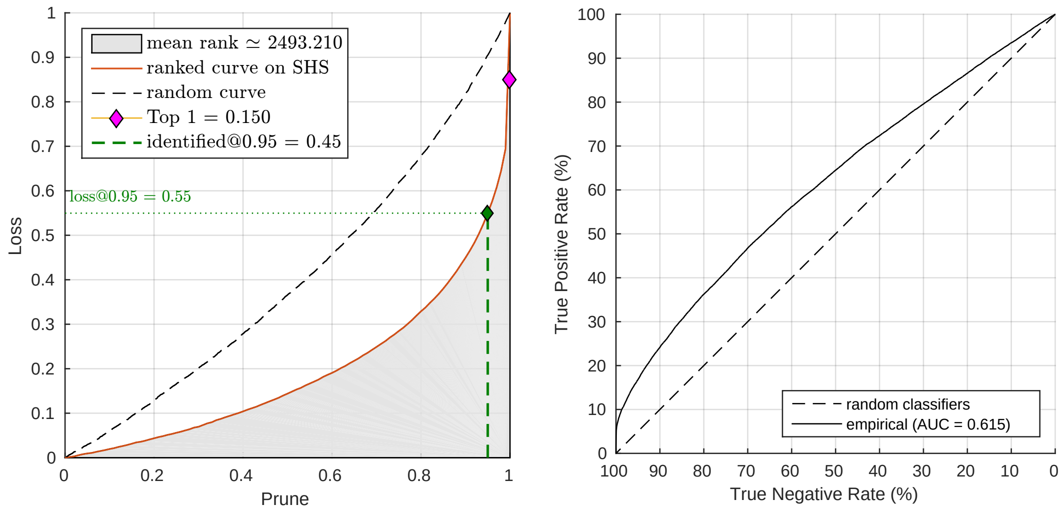


Figure 5.14: Performance curves for the XCorr system.

MR	MRR	MAP	Top-1	Top-10	Top-100	ROC AUC
$2,478$	0.17	0.09	$1,930$	$2,641$	$3,749$	0.61

Table 5.4: Performance metrics for the Cross-Correlation method.

The performance of the XCorr system is similar to the performance of SiMPle, as can be seen in Figure 5.14 and Table 5.4. The MR is $2,478$ which gives the average position of the first match for the queries $(0.19$ normalized with respect to $12,855$ queries). In contrast to SiMPle, the MRR is higher with a value of 0.171, and so is the MAP, with a value of 0.09. The number of queries identified in top-1 is $1,930$ which is a huge improvement over the previous method. Note however that in terms of top-100, the number of identified queries is $3,749$, which is lower than SiMPle $(4,074)$. The ROC AUC is close to SiMPle, with a value of 0.61, which does not indicate an ideal separability. The loss at a pruning rate of 0.95 is 0.55 which indicates that $45\,\%$ of the queries are identified in the top-5%.

5.3.1.5 Timbre (MFCC)

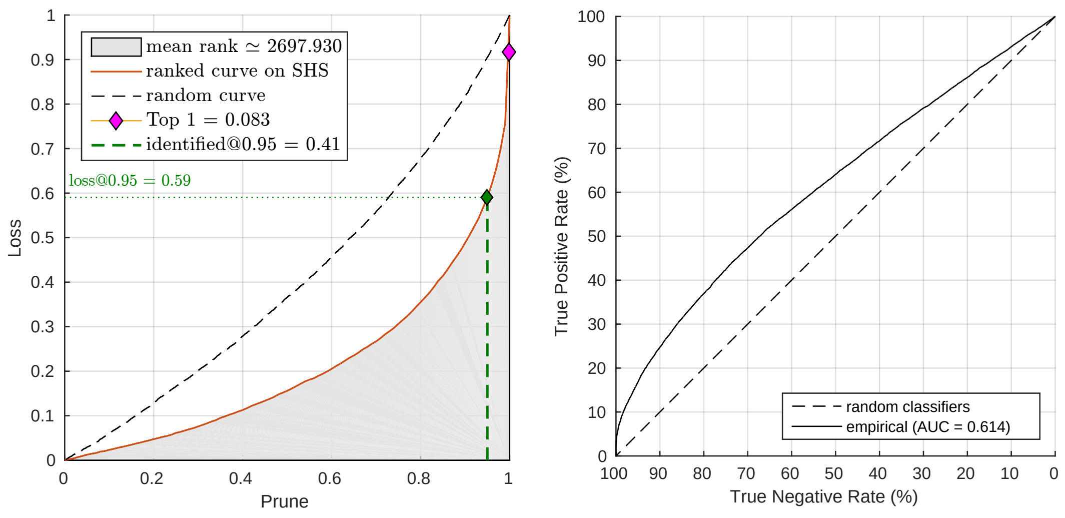


Figure 5.15: Performance curves for the MFCC system.

MR	MRR	MAP	Top-1	Top-10	Top-100	ROC AUC
$2,688$	0.10	0.05	$1,066$	$1,678$	$2,916$	0.61

Table 5.5: Performance metrics for the Timbre method.

This method was introduced by Tralie et al. [105] in 2015 and is based on alignment of timbral sequences (with the so called MFCC coefficients) rather than chroma features. Despite the fact that timbre should be dependent on the instrumentation and voicing, and therefore not produce adequate performance, Figure 5.15 shows that the system does not perform worse than others. This is explained by the fact that this method does not consider absolute timbre features, but the relative evolution of timbre over time, which is similar among cover versions. The MR here is $2,688$ (0.21) which is quite high, thus the area under the PL curve is quite wide. The MRR is given by 0.1 and the MAP by 0.05, which is quite low. Note that the top-1 metric is quite high with a value of $1,066$, which corresponds to 8 % of the test set. The top-10 and top-100 metrics, shown in Table 5.5 respectively correspond to 13 % and 23 % of the queries, and the loss rate at a pruning rate of 0.95 is 0.59. 41 % of the queries are thus identified when 95 % of the database is dropped.

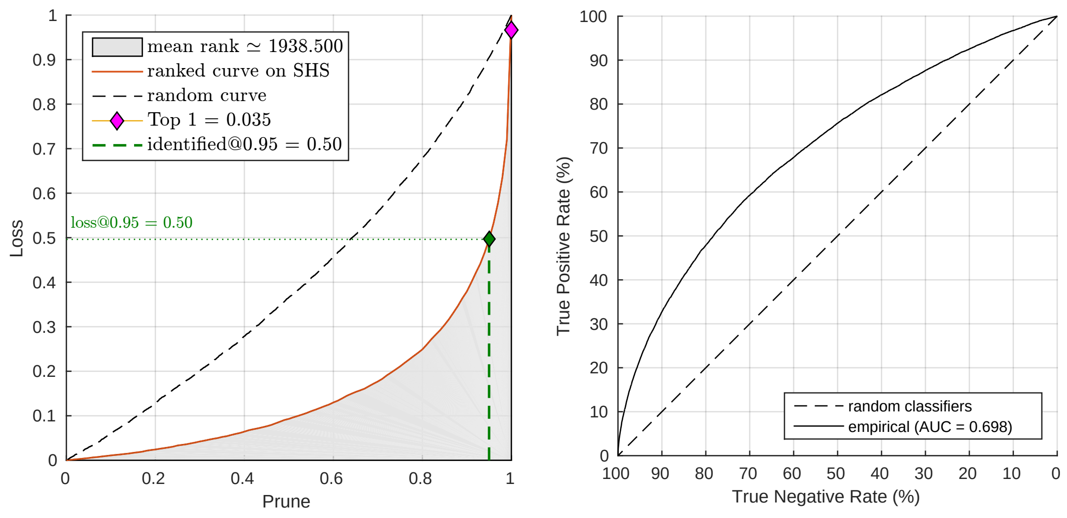


Figure 5.16: Performance curves for the Clustering system.

MR	MRR	MAP	Top-1	Top-10	Top-100	ROC AUC
$1,930$	0.06	0.03	449	$1,211$	$3,251$	0.59

Table 5.6: Performance metrics for the clustering system.

5.3.1.6 Clustering

The clustering method is fast in terms of runtime as it only computes 100-dimensional Euclidean distances. The performance is comparable to the results observed so far, as shown in Figure 5.16 and Table 5.6. The MR is $1,930$ (0.23), which is high compared to other systems. The identification rate in the top-1 is of 449, which corresponds $3\,\%$ of the test set. It follows that the MRR and MAP are also low, respectively set to 0.06 and 0.03. The proportion of tracks identified in the top-10 and top-100 are respectively given by $9\,\%$ $(1,211$ tracks) and $25\,\%$ $(3,251$ tracks). The loss rate at a pruning rate of 0.95 is high compared to other systems, with a value of 0.50. Therefore, only 50 % of the queries are identified in the top-5%.

5.3.1.7 Chroma mean

MR	MRR	MAP	Top-1	Top-10	Top-100	ROC AUC
$1,868$	0.05	0.02	329	$1,191$	$3,343$	0.696

Table 5.7: Performance metrics for the Chroma-Mean system.

Figure 5.17 displays the performance of the chroma-mean estimator as evaluated on the SHSD TS. The performance of the method is relatively weak and is given by Table 5.7.This confirmed

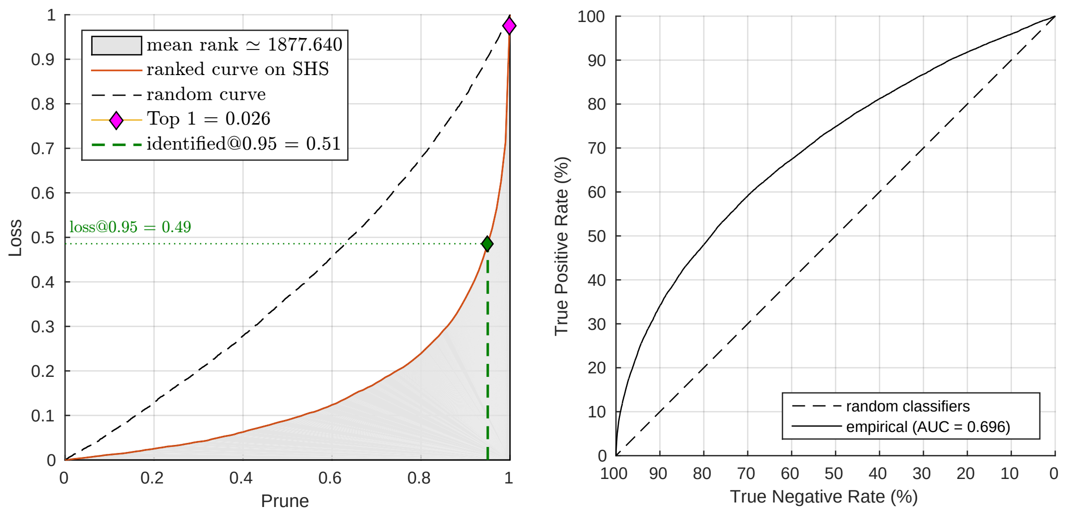


Figure 5.17: Performance curves for the chroma mean system.

by the ROC and RPLC, with a MR of 0.15, a MRR of 0.049 and a MAP of 0.02. The number of queries identified in top-1 is 329, which represents $2.5\,\%$ of all tested queries on the considered TS. In the top-10, the proportion of identified queries increases with $1,191$ identified queries, which corresponds to $9.2\,\%$ of the queries. In the top-100, the increase of identified queries is more significant, with $26\,\%$ of identified queries. The runtime performance depends on the implementation of the machine learning algorithm. Extracting the 18 requested features from a pair of tracks is straightforward and fast. Predicting a probability based on these features might be a bit slower as it has to run through the random forest.

5.3.1.8 Tempo

MR	MRR	MAP	Top-1	Top-10	Top-100	ROC AUC
$2,874$	0.00	0.00	3	79	920	0.56

Table 5.8: Performance metrics for the tempo.

Our method based on tempo, using machine learning, shows a poor performance. Considering that many tracks in any databases are likely to have a similar tempo, one should not be surprised by the low performance displayed in Figure 5.18 and Table 5.8. In Chapters 7 and 8, we will investigate how the method performs when combined with all the other systems.

The performance here is given by a MR of $2,874$ (0.22) and a ROC AUC of 0.59, showing a weak separability. One should note the low performance near the top of the returned list of results, with a MRR and MAP close to 0, and a top-1 metric of 3. The performance is also low

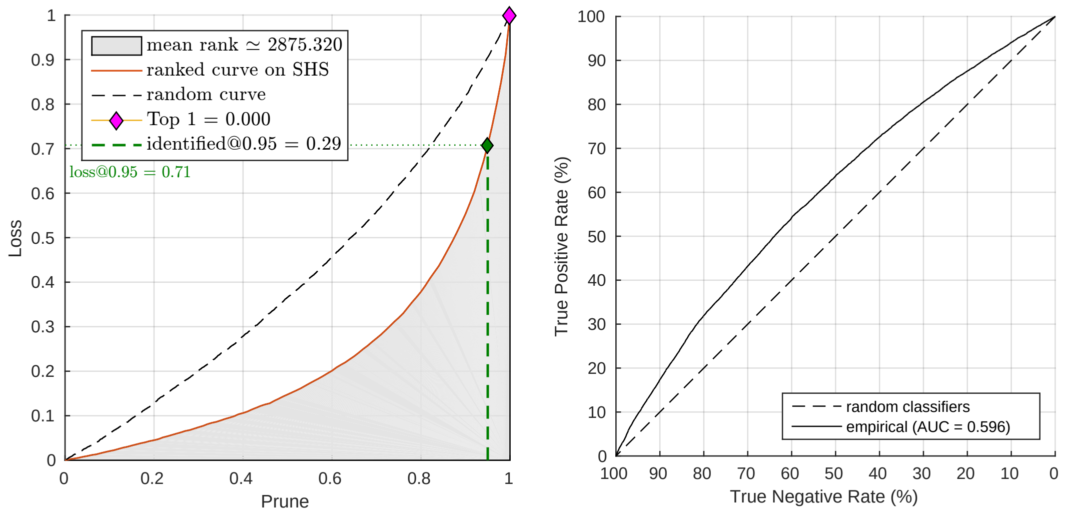


Figure 5.18: Performance curves for the tempo system.

for the top-10 and top-100, as observed in Table 5.8. Finally, the loss at a pruning of 0.95 is of 0.71. So only 29% of the queries are identified in the top-5%.

5.3.1.9 Beats

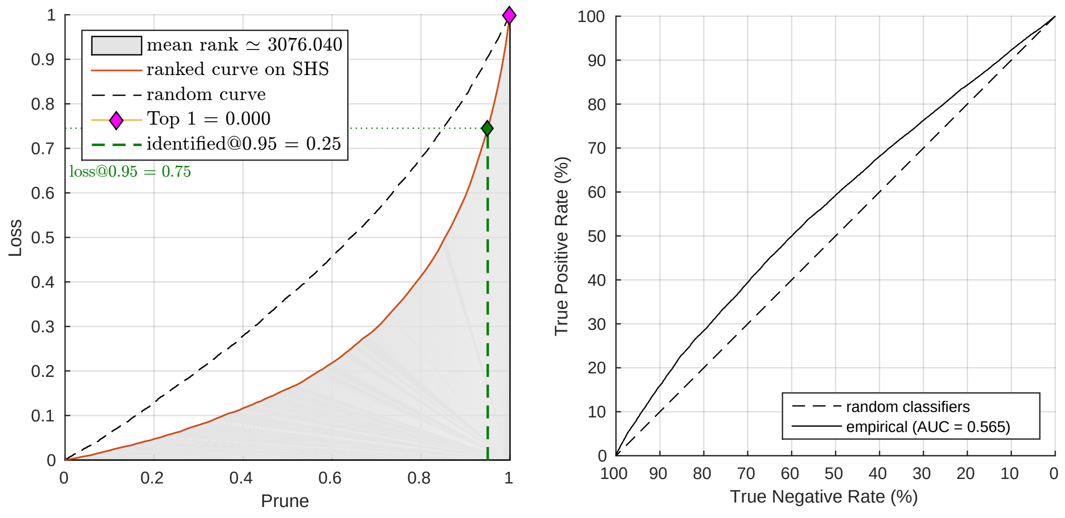


Figure 5.19: Performance curves for the beats system.

Similarly to the tempo, our method based on the number of beats, using machine learning, shows a poor performance. Performance is displayed in Figure 5.19 and Table 5.9. The performance here is given by a MR of $3,075$ (0.24) and a ROC AUC of 0.56, showing a weak separability. One should note the low performance near the top of the returned list of results,

MR	MRR	MAP	Top-1	Top-10	Top-100	ROC AUC
$3,075$	0.0	0.0	4	67	740	0.59

Table 5.9: Performance metrics for beats

with a MRR and MAP close to 0, and a top-1 metric of 4. The performance is also low for the top-10 and top-100, as observed in Table 5.9. Finally, the loss at a pruning of 0.95 is of 0.75.

5.3.1.10 Duration

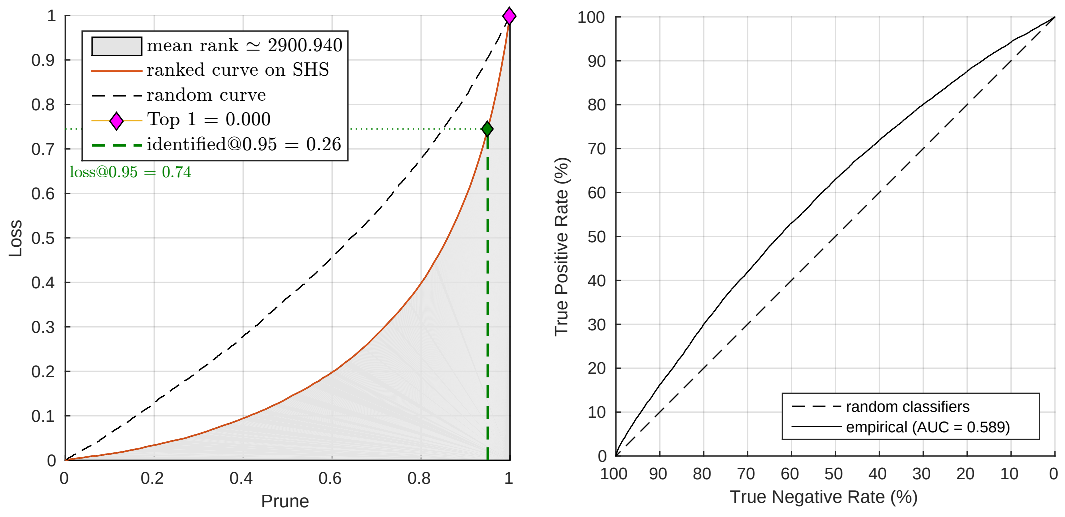


Figure 5.20: Performance curves for the duration system.

MR	MRR	MAP	Top-1	Top-10	Top-100	ROC AUC
$2,900$	0.0	0.0	6	79	638	0.59

Table 5.10: Performance metrics for duration

Similarly to the tempo and beats, our method based on the duration, using machine learning, shows a poor performance. Performance is displayed in Figure 5.20 and Table 5.10. The performance here is given by a MR of $2,900$ (0.23) and a ROC AUC of 0.59, showing a weak separability. One should note the low performance near the top of the returned list of results, with a MRR and MAP close to 0, and a top-1 metric of 6. The performance is also low for the top-10 and top-100, as observed in Table 5.10. Finally, the loss at a pruning of 0.95 is of 0.74. Note that similarly to the tempo, many tracks in a database might share a similar duration. Despite that, the performance is better than random, and thus brings some information that might be used in a combination process.

5.3.2 Comparative analysis

The 10 CSI systems presented in the previous sections show diverse performance, some of them reporting better performance near the top-1, and other weak performance. The first observation one can make is that none of these systems allows to identify more than 48% of the queries in the top-10, unless the pruning rate is set low. This means that none of them will allow one to have a fast user experience as many tracks remain to be analyzed by a subsequent expert. Some of the systems, however, perform better than others, with respect to the selected metrics. We provide in this section a comparative analysis of the performance of the studied systems, with respect to the MR, MRR, MAP and Top-K metrics.

5.3.2.1 Mean Rank

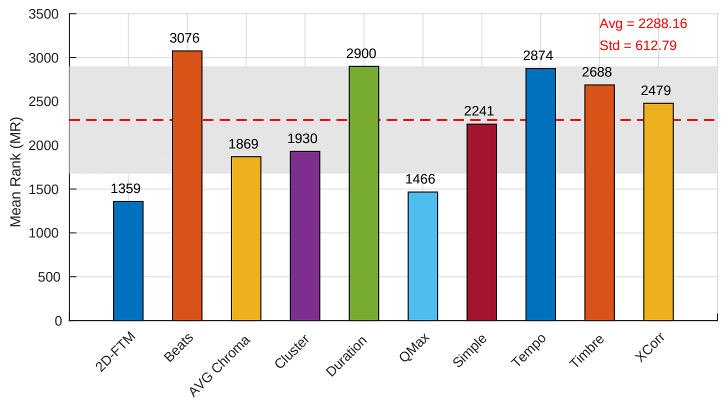


Figure 5.21: Performances of 10 CSI systems w.r.t. the Mean Rank (MR). The lower mean rank, the better it is. In this case, the 2D-FTM system outperforms the others, with the QMax method. The average MR is $\mu = 2,288.13$ with a standard deviation $\sigma = 612.79$.

Figure 5.21 shows the absolute MR value for the 10 selected methods on our SHSD TS containing $12,856$ tracks. The average MR, computed over all method is $\mu = 2288.13$, with a standard deviation $\sigma = 612.79$. We remind that the lower the MR is, the better the system performs. From Figure 5.21, one can derive that systems based on chroma features perform significantly better than other systems, except for the cross-correlation method (MR$=2,479$) and the Simple method (MR$=2,241$). Systems based on single features such as the beats, the duration and the tempo perform rather poorly. The same conclusion yields for the system based on timbre.

From Figure 5.21, one cannot conclude much more as the MR does not bring much information. Indeed, geometrically speaking, the MR is given by the area under the RPLC. Therefore, all we can tell from that score, is that the lower it is, the lower the curve lies. If the mean rank gets lower, either the curve gets closer to the x-axis (the prune axis), or the y-axis (the loss axis), thus reducing the area under the curve. Alternatively, the whole curve can lower towards the optimal point. The mean rank is therefore quite difficult to interpret.

The best system in terms of MR is the 2D-FTM, with a value of $1,359$ (0.1), followed by QMax (0.11) and AVG Chroma (0.14) and Clustering (0.15). One can observe that despite the low performance in other metrics for AVG Chroma and Clustering, they have a low MR, meaning that the area under the PL curve is decreased. Indeed, from the RPLC for AVG Chroma (Figure 5.17) and Clustering (Figure 5.16) one can observe that the curve fits the loss axis in the top right corner, thus indicating an increase of the performance near the top-1. The same applies for the QMax system with a steep descent from the top right corner, indicating an even better performance. This is not the case for tempo, duration and beats. However, SiMPle, XCorr and Timbre also fit the top-right corner. The difference in their MR is due to the fact they tend descend less steeper toward the (0,0) point.

Correlating Figure 5.21 with the RPLC of the considered systems, one cannot draw any conclusion. Some systems fits the top-right corner while having a low MR (Clustering, Chroma mean, QMax, 2D-FTM), while others also fit the top-right corner with a much higher MR (Timbre, XCorr, Simple). It follows that the MR metrics is not useful for evaluating cover song identification systems, because the MR as a surface can easily change with any change of direction in the curve.

5.3.2.2 Mean Reciprocal Rank

The MRR is a metric that emphasizes systems identifying more queries *near* the top-1. Geometrically speaking in the RPLC, this means that the curve fits better the top-right corner. The steeper the curve goes near the top-1, the more the MRR increases. Relatively to the discussion about the MR (Section 5.3.2.1), this metric is more suited for identifying these systems performing better in the top-1. Note that with the MRR, the highest it is, the best it is.

Figure 5.22 shows the performances of all evaluated systems, with respect to the MRR metric. The average performance is $\mu = 0.10$ with a standard deviation $\sigma = 0.13$. The best performing system here is the QMax system with a MRR of 0.42, which identifies $39\,\%$ of the queries in the top-1. It is followed by XCorr (0.17) and 2D-FTM (0.15) which respectively identify $15\,\%$ and $11\,\%$ of the queries in the top-1. The QMax system performs $141\,\%$ better with respect to the second best performing system (XCorr) in terms of MRR. This is definitely the best system in terms of *near top-1* identification. Without any surprise, the worse systems in terms of *near top-1* are the ones based on beats (0.003), duration (0.003) and tempo (0.004).

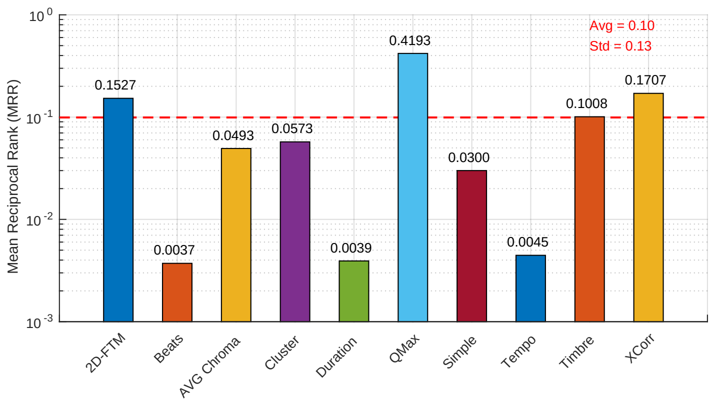


Figure 5.22: Performances of 10 CSI systems w.r.t. the Mean Reciprocal Rank (MRR). QMax outperforms the other systems, followed by XCorr and 2D-FTM. The average performance is $\mu = 0.10$ and the standard deviation is $\sigma = 0.13$.

SiMPle, AVG Chroma and Cluster have a relatively low MRR, but perform significantly better than the single-dimensional systems. One should observe that Timbre, on the other hand, has a decent MRR of 0.10. Indeed, it is able to identify $8\,\%$ of the queries in top-1 and $13\,\%$ in top-10, which is surprisingly good considering that it is only based on timbre features rather than more general chroma features.

5.3.2.3 Mean Average Precision

The mean average precision (MAP) is a generalization of the MRR. Instead of considering the rank of the first identified matching track, it considers the ranks of all identified versions for a given query (see Chapter 4). Thus, as the MAP increases, the number of queries for which most versions are found near the top-1 increases. It follows that the MAP behaves similarly to the MRR, as it inevitably considers the position of the first match. It increases if more versions are found in the top of the ranking. One can observe that the shape of Figure 5.23 is indeed correlated with the shape of Figure 5.22 (MRR).

The average MAP for all method is $\mu = 0.05$ with a standard deviation $\sigma = 0.07$. As with the MRR, the best system is QMax, with a MAP of 0.24, again followed by XCorr (0.09) and 2D-FTM (0.08). QMax performs $166\,\%$ better than the second best performing system (XCorr). This confirms the observation based on the MRR that QMax is the best for identifying queries near the top of the ranking.

Again, with no surprises, systems based on beats, duration, tempo have a low performance, all with MAP values around 0.002. The timbre method has a decent MAP of 0.05, meaning that for approximately $5\,\%$ of the queries, all versions are identified near the top of the ranking.

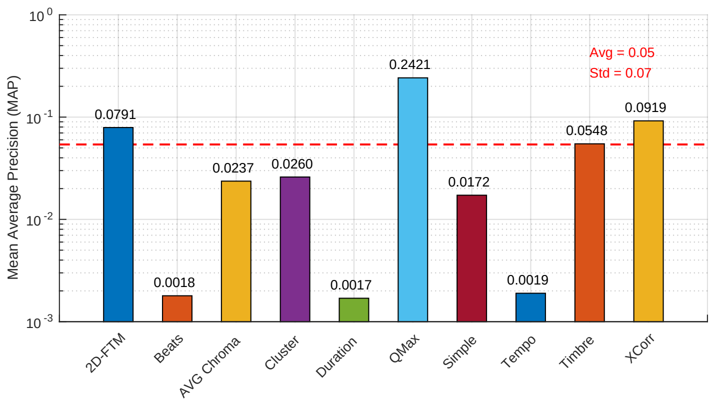


Figure 5.23: Performances of 10 CSI systems w.r.t. the Mean Average Precision (MAP). Performance is correlated to the performance in terms of MRR. Again, QMax outperforms the other systems, followed by XCorr and 2D-FTM. The average performance is $\mu = 0.05$ while the standard deviation is $\sigma = 0.07$.

The same observation yields for the AVG Chroma, Cluster and Simple, identifying all versions for approximately $2\,\%$ of the queries.

Note that the MAP here is given here as an indicator of how a CSI system is able to quickly identify all versions for a given query. However, following the SMR principle (Chapter 3), we are mainly interested about how the system is able to identify *at least one version* close to the top of the ranking. In that context, the MRR seems more appropriate than the MAP, if one is interested in the *near top 1* performance. However, the MAP can be used in addition to the MRR to analyze how the considered systems perform as identifying all versions for the queries (near the top of the ranking). Indeed, as can be observed on the individual RPL curves (Section 5.3.1), systems producing the highest MAP scores produce a curve that follows a steep descent in the top right corner of the PL space.

5.3.2.4 Identification at fixed prune rate

Comparing the identification rates at a fixed prune rate corresponds to considering the performance at a specific top-k threshold. Indeed, considering the identification at a prune rate of 0.95 is the same as considering the proportion of identified queries in the top-5% of the ranking. As our TS contains 12,856 tracks, the corresponding threshold is 643. The proportion of queries that are not lost at a prune of 0.95 thus is the proportion of queries identified in the top-643.

Figure 5.24 compares the identification at 0.95 for the ten selected systems. One can observe an average identification rate of $\mu = 0.38$ and a standard deviation $\sigma = 0.15$. As expected

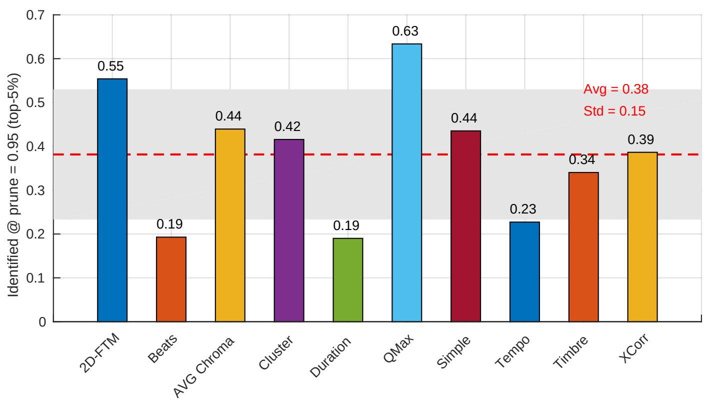


Figure 5.24: Performances of 10 CSI systems w.r.t. the percentage of identified tracks when the database is pruned by 95% (Identified@0.95). Best systems are QMax and 2D-FTM with a identification rate of respectively 0.62 and 0.55. The average identification rate is $\mu = 0.38$ with a standard deviation $\sigma = 0.15$.

from other metrics, QMax identifies the biggest proportion of queries, with $63\,\%$ of the TS identified in the top-5%. This means that $8,100$ over $12,856$ are identified. The second best performing system at that threshold is 2D-FTM, with $55\,\%$ of the queries identified $(7,071$ queries). AVG chroma, Cluster, Timbre and XCorr have all an identification rate between 0.39 and 0.44, which is a quite decent performance. Our methods based on beats, duration and tempo are able to identify approximately 20% of the queries in the top-5%.

If we now increase the threshold to a pruning rate of 0.99 (keeping 1% of the database, thus the top-128), the performance inevitably drops, as shown in Figure 5.25.

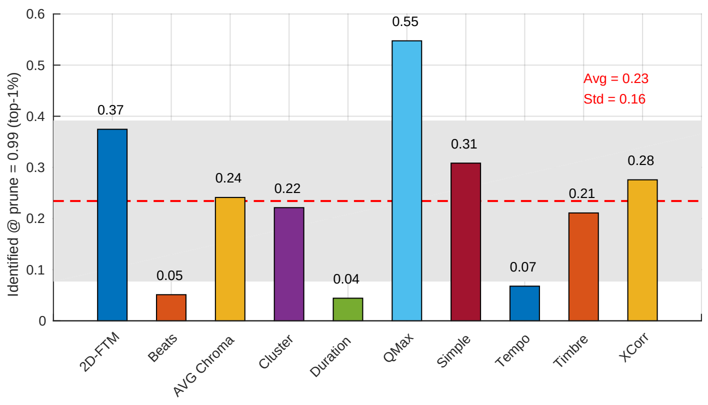


Figure 5.25: Performances of 10 CSI systems w.r.t. the percentage of identified tracks when the database is pruned by 99% (Identified@0.99). Clearly the QMax method outperforms all the others with an identification rate of 0.55 for the top 1%. The average identification rate at top-1% is $\mu = 0.29$ with a standard deviation $\sigma = 0.16$.

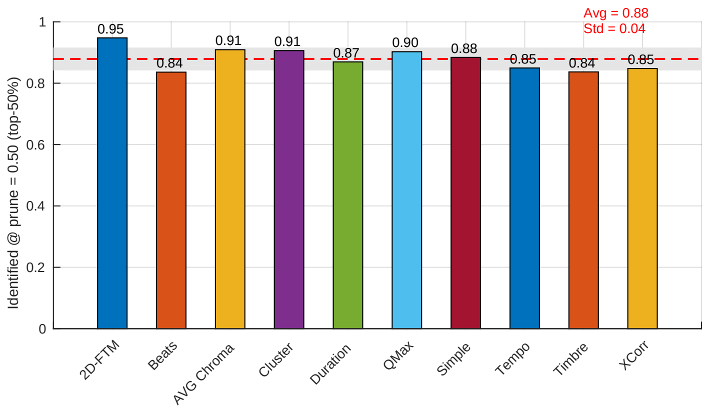


Figure 5.26: Performances of 10 CSI systems w.r.t. the percentage of identified tracks when the database is pruned by 50% (Identified@0.5). One can observe that the performance is quite similar for each method. The mean of the identification rates is $\mu = 0.67$ and the standard deviation is $\sigma = 0.09$.

One can observe that the general shape of that figure is similar to the shape of Figure 5.24, indicating the the performance scales when increasing the threshold. As the top-1% is close to the top of the ranking, the shape of Figure 5.25 is also similar to the shape of Figure 5.22,which compares the systems with respect to the MRR.

When only keeping 1% of the database, QMax is still the best, with an identification rate of 0.55, thus losing $12\,\%$ with respect to the performance at a prune rate of 0.95. 2D-FTM loses $32\,\%$ of its performance but keeps its second position in the ranking. AVG Chroma, Cluster, Timbre and Xcorr respectively lose $45\,\%$, $47\,\%$, $38\,\%$ and $28\,\%$. However, SiMPle still performs well in the top-1% with an identification rate of $31\,\%$, thus losing only $22\,\%$ of its performance compared to the top-5%.

One can now consider the performance at a pruning rate of 0.5, as depicted in Figure 5.26. In that case, one can observe that all methods perform similarly, with an average identification rate of $88\,\%$, with a low standard deviation $\sigma = 0.04$. This means that for all methods, one can drop $50\,\%$ of the database and still claim to have an identification rate close to 0.88. From that observation, one can conclude that most studied methods are optimized to identify more tracks near the top-1, when reducing the threshold. From Figure 5.26, it is clear that most systems are better at reducing the loss at lower pruning thresholds.

5.3.2.5 Ranking of the methods

Table 5.11 ranks all 10 CSI systems with respect to the aforementioned metrics. One can observe that for all metrics, the last systems of the ranking are systematically the tempo,

MR $\downarrow$		**MAP** $\uparrow$		**MRR** $\uparrow$		**Identified@0.95** $\uparrow$	
method	**score**	**method**	**score**	**method**	**score**	**method**	**score**
2D-FTM	$1,359$	QMax	0.24	QMax	0.41	QMax	0.63
QMax	$1,466$	XCorr	0.09	XCorr	0.17	2D-FTM	0.55
Avg chroma	$1,868$	2D-FTM	0.08	2D-FTM	0.15	Avg chroma	0.44
Cluster	$1,930$	Timbre	0.05	Timbre	0.10	Simple	0.44
Simple	$2,240$	Cluster	0.026	Cluster	0.06	Cluster	0.42
XCorr	$2,478$	Avg chroma	0.023	Avg chroma	0.05	XCorr	0.39
Timbre	$2,688$	Simple	0.02	Simple	0.03	Timbre	0.34
Tempo	$2,874$	Tempo	0.001	Tempo	0.004	Tempo	0.23
Duration	$2,900$	Duration	0.001	Duration	0.003	Duration	0.19
Beats	$3,075$	Beats	0.001	Beats	0.003	Beats	0.19

Table 5.11: Ranking of 10 CSI systems with respect to the MR, MRR, MAP and Identification@0.95 performance. The arrows near the title of the columns indicate whether the score is ordered in ascending or descending order.

followed by the duration and beats systems. It seems obvious that systems based on such simple features perform poorly, in comparison to systems based on chroma or timbre features. One can also observe that the QMax method is almost always at the top of the ranking. It further produces scores that are way above the next systems, as discussed in the previous sections.

From the ranking in Table 5.11, it is clear that the MAP is strongly correlated to the MRR as they both produce exactly the same ranking. Furthermore, the gap between QMax and XCorr is huge for both metrics. The ranking for the MR is different from MAP and MRR, as it does not consider the same area in the PL space. The MR column seems to actually be more correlated with the last column, corresponding to identification rate at a pruning of 0.95.

From the results in the table, one conclusion is straightforward: no system is able to identify more than $41\,\%$ of the queries near the top-1 (MRR column), in a database containing $12,856$ songs. Taken individually, none of these systems could be used in a commercial system, aiming at identifying as many tracks as possible near the top-1. However, such systems could be used as the first stage of a commercial system, for reducing the size of the set to be further processed later. Indeed, one can prune the database even more, thus setting the threshold lower, for example around 0.95, and achieve a satisfactory identification rate. The lower the threshold is set, the better the performance will be, in terms of identification rate. On the other hand, the size of the returned subset will increase significantly.

5.4 Conclusion

Results provided in this chapter constitute the first evaluation of multiple cover song identification systems on a large database, containing $12,856$ songs. To our knowledge, no such evaluation has been performed before. In addition to the selection of 5 state-of-the-art methods based on chroma and timbre features, we studied 5 systems of our own, based on chroma features as well as single dimensional features such as the tempo, the duration and number of beats of the songs. For some of these systems, we learned similarity estimation models using extra trees as a supervised learning algorithm. The learned models allow to predict probabilities of similarity or dissimilarity. Most systems based on the "simple" features (tempo, beats, duration, chroma mean) do not perform well for CSI. However, they do perform better than random. In the next chapters of this thesis, we will explain how to take advantage of such weak information.

Each system described in this chapter was evaluated on the same test set, thus leading to comparable results, using various performance indicators such as the mean rank, the mean reciprocal rank and the mean average precision. We also provide for the first time a large-scale evaluation of the best CSI system so far, based on the cross-recurrence principle and the QMax algorithm. As suggested by earlier publications, this system is still the best to perform so far. Due to its quadratic complexity, it is also the slowest one, and it took us several weeks of computing using multiple processors to evaluate it. We also included a quite recent system (published in 2016), SiMPle [95], that attempts at considering structural information to identify cover songs. Other systems known in the literature have been evaluated and compared to each others.

The analysis of the performance of the systems leads to one straightforward conclusion: no system at this time is usable on a large-scale basis by end users. Using these systems, it seems clear that the problem of CSI is far from being solved. It also leads to another conclusion: all systems allow to identify a large proportion of the tracks when setting the decision threshold lower. At a pruning rate of 50%, all systems are able to identify approximately 88% of the tracks, which is a good performance.

Minimizing the loss at a pruning rate of 50% is a correct result, yet still far enough from a commercial performance. It should be further reminded that, although the evaluation database contains several thousands of tracks, current commercial systems scale to millions of songs. Clearly, there is a need for improvement. In the remainder of this thesis, we present solutions to combine systems to take advantage of the information of each of them. This allows to improve the global performance, compared to individual solutions.

Prior to that, the next chapter provides an analysis of several systems when pushed to their limits. Specifically, we analyze how systems perform when the audio queries are highly acoustically degraded.

CHAPTER 6

Robustness to Audio Degradations

This chapter considers the evaluation of multiple state-of-the-art CSI systems in a noisy environment. Indeed, while many existing systems report a decent performance for CSI, they usually were evaluated in a controlled environment, with a single evaluation database designed for the task. Here, we consider a selection of four existing systems, already described in the previous chapter: XCorr (Section 5.3.1.4), 2D-FTM (Section 5.3.1.1), QMax (Section 5.3.1.2) and finally MFCC-SW (Section 5.3.1.5). These systems are based on chroma and timbre features. For each of these systems, we study the robustness of the features and the retrieval algorithms against acoustic degradations such as adding ambient noise at different levels, adding reverberation, simulating a live recording situation, applying harmonic distortion and convolving the query by the impulse responses of a smartphone microphone and speaker. Such experiments are interesting as they give us some information about how CSI systems perform in real conditions, for example at a live concert, with a smartphone in a crowded room. In order to apply audio degradations, it is necessary to access audio data to modify the signals using a suitable toolbox. Unfortunately, the Second Hand Song dataset that we have been using for our research does not provide audio content. Due to that limitation, we restricted our study on a smaller popular dataset, used in several research on CSI. The chosen dataset is the Cover80 (CC80) dataset and contains 80 songs for which two versions are available, thus proposing a total of 160 tracks. Because of the size of the CC80 dataset, we could not build suitable models for the CSI systems based on machine learning (described in the previous chapter). Indeed, data is not sufficient to split the dataset in representative learning and test sets to optimize the models. Note that we have analyzed the performance of several CSI systems on a larger scale in the previous chapter. The results show that the studied systems are quite robust against audio degradations.

6.1 Audio degradations

In this chapter, we study how audio degradations affect the performance of four CSI systems. We selected six modifications to apply to the audio queries: add ambient restaurant noise, apply harmonic distortion, live recording simulation, convolution with the impulse response (IR) of a large hall, and the IRs of a smartphone speaker and a smartphone microphone. We used the Audio Degradation Toolbox (ADT) by Mauch et al. [73] to modify the audio signals. The ADT provides Matlab scripts that emulate a wide range of degradation types. The toolbox provides 14 degradation units that can be chained to create more complex degradations.

6.1.1 Single degradations

We first apply non-parametric degradations. These audio modifications include: a live recording simulation, adding reverberation to the queries and convolving the queries by the IRs of a smartphone speaker and microphone. The live recording unit convolves the signal by the IR of a large room ('GreatHall', $RT_{30}@500Hz = 2s$, taken from [100]) and adds some light pink noise. The reverberation corresponds to the same convolution, without the added pink noise. The smartphone playback and recording simulations correspond to convolving the signal with the IR of respectively a smartphone speaker ('Google Nexus One') and the IR of the microphone of the same smartphone. The speaker has a high-pass characteristic and a cutoff at around 500Hz [73].

6.1.2 Parametric degradations

We add some ambient noise and distortion to the audio signals. The ambient noise corresponds to a recording of people in a pub. The recording is provided with the ADT [73]. We successively add the ambient noise at multiple SNR levels, from 30 dB to 5 dB to study how robust the systems are. We also successively add some harmonic distortion. To achieve this, the ADT applies a quadratic distortion to the signal. We iteratively applied the distortion with 2, 4, 6, 8 and up to 10 iterations. One iteration of quadratic distortion is applied as follows: $x = sin(x * \pi/2)$.

6.2 Experimental setup

6.2.1 Evaluation database

We evaluate our experiments on the Cover80[1] dataset [35, 37]. The dataset contains 80 songs for which two versions are available, thus proposing a total of 160 tracks. While this is definitely not a large scale dataset, it has the advantage of providing audio data, allowing us to extract features straight from the audio. Other bigger datasets such as the Million Song Dataset (MSD) or the Second Hand Song Dataset (SHS) are available, but they do not provide audio data. Rather than that, they provide pre-computed audio features that can be exploited in MIR algorithms. For this specific research, we need the audio data so that we can modify the signals with respect to each degradation. We created 4 copies of the dataset for the single degradations (convolutions) and applied the convolutions with the default parameters provided by the ADT. For the ambient noise degradation, we created 5 additional copies, with added noise at SNRs of respectively 30 dB, 20 dB, 15 dB, 10 dB and 5 dB. For the distortion, we also created 5 copies, applying the distortion as explained in Section 6.3.3 (parametric).

6.2.2 Features extraction

To use the four selected CSI systems, we need chroma features as well as MFCC features for the timbre. We extracted chromas from the audio using the Essentia library [2] with the HPCP [50] algorithm. Each chroma is elevated to a power of 2 to highlight the main bins, and then normalized to unit-norm. We first extracted beats location using a beat-tracker provided in the library, and computed 12-dimensional chroma features at each beat instant, with a sampling rate of 44.1kHz. For the computation of the self-similarity matrices based on MFCC features, we used some code that was kindly provided by the authors. The code makes use of the librosa[3] library to extract 20-dimensional beat-synchronous MFCC vectors.

6.3 Results

6.3.1 Evaluation methodology and metrics

For each modification of the database, we apply the same evaluation methodology. We consider all tracks of a noisy database (160 tracks) and we compare them to all tracks in the original

[1]`http://labrosa.ee.columbia.edu/projects/coversongs/covers80/`
[2]`http://essentia.upf.edu/`
[3]`https://github.com/librosa/librosa`

database. Note that both databases contain exactly the same tracks. Each track in the noisy database is taken as a query and compared to 159 songs in the original database (we do not compare the query to itself). Using the similarity scores, we build an ordered ranking of 159 candidates for each query (highest score is considered most similar). We then look in the ordered ranking where the second version of each query is located (in terms of absolute position). We report the results in terms of Mean Rank (MR), which corresponds to the mean position of the second version (lower is better), in terms of Mean Reciprocal Rank (MRR) which corresponds to the average of the reciprocal of the rank of the identified version (higher is better). We also reports the proportion of queries for which the second version was identified at the first position (TOP-1), or in the 10 first returned tracks (Top-10).

6.3.2 Single degradations

Figure 6.1 compares the performance of the four selected CSI systems with respect to single audio degradations. The first column (blue) always corresponds to the performance of the system with no degradation. As one can observe on the figure, the degradation that affects the most each system is the *smartphone playback*. In particular, the 2D-FTM system has a significant loss of performance, with a decrease of 80% in terms of MRR.

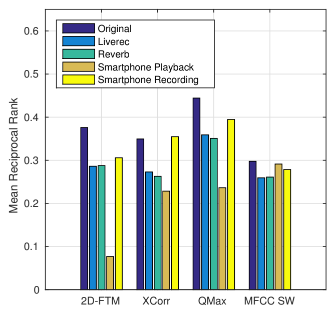
This can be explained by the fact that the smartphone speaker has a high-pass characteristic with a cutoff at around 500Hz. Therefore, the spectrograms upon which the chromas are built lose much information compared to no degradation at all. Note that the timbre based system (MFCC SW) is definitely robust against the smartphone playback degradation. For both live recording simulation and added reverberation, all systems are not degraded significantly and performs similarly for both degradations. The most stable feature with respect to all degradations is the MFCC SW, with a maximum decrease of 13% in terms of MRR for the live recording simulation.

6.3.3 Ambient noise and distortion

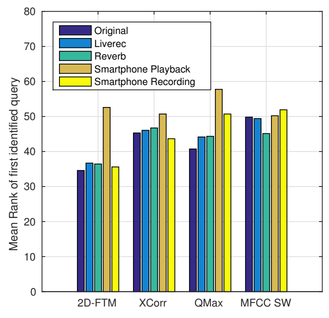
Figure 6.2 shows the evolution of the performance of the four selected CSI systems when the percentage of ambient noise is increased (the SNR gets lower). We plot the results in terms of percentage of Noise-to-Signal amplitude ratio (NSR) to be able to represent the original point, with no noise added at all. We compute the NSR as follows:

$$NSR = \frac{100}{10^{\frac{SNR}{20}}} \tag{6.1}$$

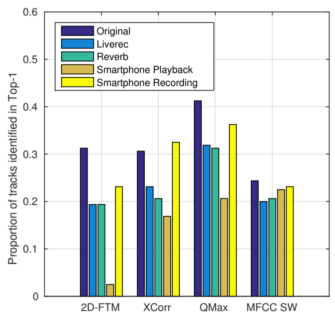
We add the ambient noise with a decreasing SNR (resp. increasing NSR) at values of 30 dB,



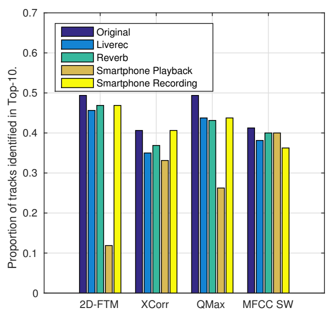
(a)



(b)



(c)



(d)

Figure 6.1: Evolution of the Mean Reciprocal Rank (a), the Mean Rank (b), the proportion of tracks identified in the Top-1 (c) and the proportion of tracks identified in the Top-10 (d) for single non parametric degradations.

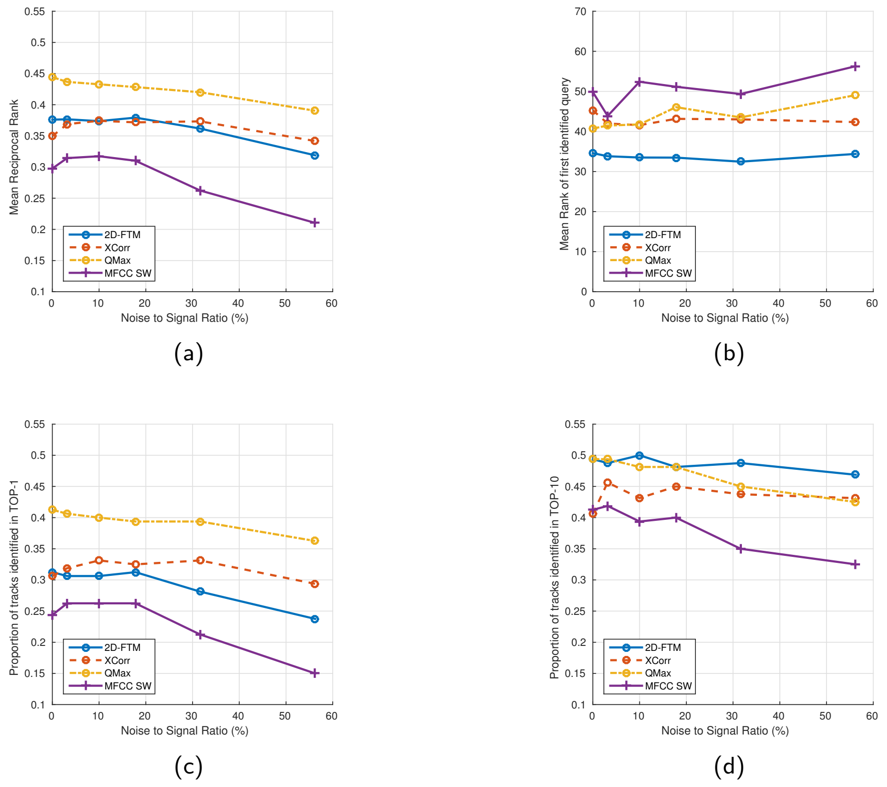
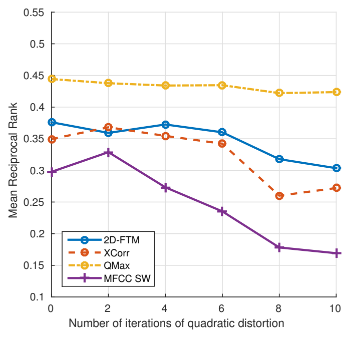


Figure 6.2: Evolution of the Mean Reciprocal Rank (a), the Mean Rank (b), the proportion of tracks identified in the Top-1 (c) and the proportion of tracks identified in the Top-10 (d) for an increasing ambient noise.

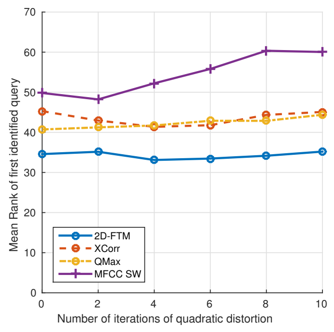
20 dB, 15 dB, 10 dB and 5 dB.

Adding an ambient noise to the original audio signal generates new frequencies in the spectrum. As the chroma features are computed based on that spectrum, we expect the performance to drop at some point. When adding up to 20% (SNR $\simeq 15$dB) of noise to the songs, all systems stay stable, with almost no loss in performance. Above 20%, the 2D-FTM and MFCC methods start to decrease the performance in terms of MRR, Top-1, and Top-10. In terms of MR, all methods stay stable at all noise levels. Note how the MRR and the Top-1 metrics render similar shapes. As both metrics take into account the position of the first match to the query, they seem to be strongly correlated.

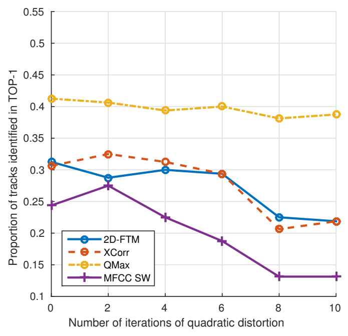
Figure 6.3 shows the evolution of the performance when we increase the number of iterations of quadratic distortion. The first observation one can make is that the QMax method is robust against any level of distortion, with respect to each metric. There is almost no loss of performance for the method. In terms of MRR and MR, 2D-FTM is also stable and does not decrease in performance. After two iterations, the MFCC method starts to drop in performance



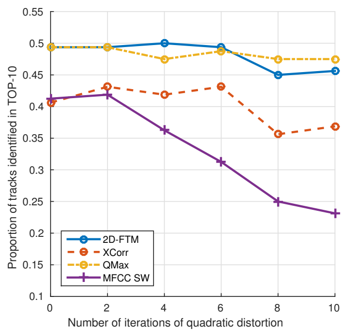
(a)



(b)



(c)



(d)

Figure 6.3: Evolution of the Mean Reciprocal Rank (a), the Mean Rank (b), the proportion of tracks identified in the Top-1 (c) and the proportion of tracks identified in the Top-10 (d) for an increasing number of iterations of quadratic distortion.

for each metric. This makes sense as the timbre is computed based on the harmonics of the signal. Applying quadratic distortion adds harmonics which can blur the timbre of the audio signal. The XCorr method drops in terms of MRR, Top-1 and Top-10 after 6 iterations, which makes it more robust than the MFCC method. Note that 6 iterations of distortion is clearly audible in the audio tracks, and the perceived music is strongly degraded compared to the clean song. In light of this, we can consider that all methods are pretty robust up to 4 iterations of quadratic distortion.

6.4 Conclusion

In this chapter, we evaluated multiple state-of-the-art cover song identification systems with respect to several audio degradations. We first selected three methods based on chroma features, thus considering the harmonic content of the songs as the main feature. These methods use different retrieval algorithms to find cover songs in a reference database. We also chose a fourth method based on timbre feature rather than chroma features. The latter makes use of a sequence alignment algorithm to find relevant cover songs. We selected the Cover80 dataset for our research, and used the Audio Degradation Toolbox to perform a series of degradations of the database. We selected six degradations, corresponding to potential real-life modifications of the sound. The degradations include a live recording simulation, adding reverberation, convolving with the impulse responses of a smartphone speaker and microphone, adding a restaurant ambient noise at multiple levels and finally adding multiple iterations of quadratic distortion.

Overall, the methods based on chroma features perform similarly against all degradations. The worst performance is achieved through a convolution with a smartphone speaker and is produced by the 2D-FTM method. Convolving the signal by the microphone of the smartphone, however, does not degrade the performance significantly. Same goes for the live recording simulation and added reverberation. The timbre based method is extremely stable with single degradations, with almost no loss in performance with respect to all metrics, which makes it a robust method, although it performs worse than chroma based methods in a clean situation.

When adding ambient noise to the songs, all systems are stable up to 20% of added noise. After that limit, the timbre method decreases significantly, while the chroma based methods stay stable. When adding quadratic distortion, all systems but the timbre one stay stable up to 6 iterations. The MFCC based system drops after two iterations. After 6 iterations, XCorr and 2D-FTM lose some performance, but not significantly (less than 10% in terms of all metrics).

Overall, the studied systems can be considered stable against the applied audio degradations. We voluntarily degraded the signals significantly to push the limits of the systems, and the performance stays pretty stable. Future work involves analyzing other cover song systems, and combining them together to study how the robustness against audio degradations performs.

CHAPTER 7

Combining Cover Song Identification Systems

Until now, we described the general task of cover song identification, as well as the relaxation known as single match retrieval. We investigated how such systems can be designed, and introduced an intuitive evaluation space that allows to visualize the performance. Having evaluated the performance of 10 selected systems, one can conclude that none of them performs sufficiently for a potential commercial system to immediately identify most of the query songs. Considered individually, these systems perform rather poorly, on a database containing 12,856 songs. The best system at this time is the QMax algorithm based on cross recurrence plots of chroma sequences, proposed by Serra et al. [92]. This system identifies approximately 5,000 tracks at the first position, which is less than half of the queries available. Such a performance is not sufficient for any real life application. It is therefore interesting to study techniques aiming at increasing the overall performance of a CSI system.

Several systems, studied in Chapter 5, are built upon chroma features. However, despite being based on the same features, the way they process them as well as the comparison algorithms they use are different. For example, the 2D-FTM and QMax systems are both based on chroma features. In the former case, chroma features are aggregated together across the entire track, thus summarizing the track. In the latter case, entire time chroma sequences are compared, thus considering the temporal evolution of the music.

Based on such differences, one can imagine that these two systems consider a different kind of information to compute the similarity. If we also consider other systems, based on timbre, tempo, duration, beats, structure, etc., it is likely that they also consider a different information

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1	1.00	0.01	-0.01	-0.04	0.01	0.18	0.02	0.00	0.03	0.36
c_2	0.01	1.00	0.01	0.04	-0.08	0.44	-0.00	0.10	0.21	0.00
c_3	-0.01	0.01	1.00	0.37	0.14	0.01	0.01	0.06	0.07	0.01
c_4	-0.04	0.04	0.37	1.00	0.15	0.02	0.01	0.08	0.06	-0.00
c_5	0.01	-0.08	0.14	0.15	1.00	-0.08	0.04	-0.02	-0.05	0.02
c_6	0.18	0.44	0.01	0.02	-0.08	1.00	-0.00	-0.00	0.13	0.16
c_7	0.02	-0.00	0.01	0.01	0.04	-0.00	1.00	0.01	0.05	-0.01
c_8	0.00	0.10	0.06	0.08	-0.02	-0.00	0.01	1.00	0.13	-0.09
c_9	0.03	0.21	0.07	0.06	-0.05	0.13	0.05	0.13	1.00	-0.08
c_{10}	0.36	0.00	0.01	-0.00	0.02	0.16	-0.01	-0.09	-0.08	1.00

Table 7.1: Pearson's correlation values between probabilities predicted by each similarity estimator. 100,000 similarity values per classifier are considered.

to output a similarity score. This suggests that these systems could be combined so that they all contribute to the global performance of a system. This can be confirmed by an analysis of the correlation matrix obtained from all systems, as displayed in Table 7.1. The values in the table show the Pearson's correlation between the scores predicted by the systems studied in Chapter 5. To compute such values, we sampled 100,000 pairs of tracks from the database, and we computed a similarity score with each classifier. We then computed the Pearson's correlation values between the lists of scores obtained with each classifier. From the values in Table 7.1, one can observe that the correlation values are quite low, thus pointing out that a potential combination might produce a system with increased performance.

Combining CSI systems together is not a trivial task. Indeed, it raises interesting questions: should the systems be combined at the feature level, or at the score level ? Most systems return scores on different scales. In that case, how to combine the resulting scores if they are not normalized to a common value ? Can we easily normalize these scores to a common scale ? Should we perform pairwise combinations between tracks, or is it better to combine entire lists of results (rankings) produced by each system ? And finally, do we need all initial systems in the combination to improve the performance, or are some systems redundant, with respect to other systems ? We attempt at answering these questions in this chapter and the following.

7.1 Combining classifiers

CSI systems can be considered as binary classifiers, taking two tracks as an input, and returning whether they are similar (positive class), or dissimilar (negative class). Thus, combining CSI systems can be considered as combining binary classifiers. Multiple techniques have been suggested in the literature.

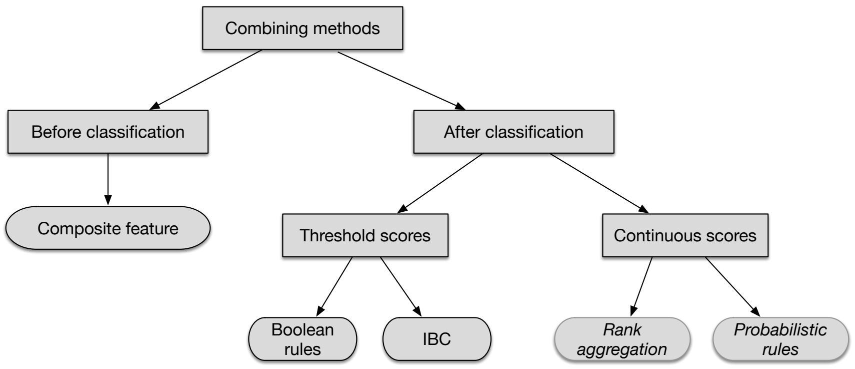


Figure 7.1: Existing combination methods for combining multiple classifiers.

Figure 7.1 shows existing methods to combine classifiers. As pointed out in Section 2.1.3.3, one way of doing is to apply combination *before classification*, thus at the feature level, or *after classification*, by considering scores returned by individual systems

Combining before classification

For combining before classification, one typical way of proceeding is to aggregate multiple features (considered by individual classifiers) together to build a composite audio feature that encapsulates multiple musical facets. However, such an approach can lead to a high dimensional feature, which can be difficult to process and classify, partly due to the curse of dimensionality. Several methods based on features aggregation have been proposed, as introduced in Chapter 2. We do not investigate such techniques in this thesis.

Combining after classification

To combine after classification, one has to consider the similarity scores returned by each individual classifier. In that case, each classifier takes its decision by considering a different features space, and computes a similarity score. Combining such scores consists in applying mathematical operations on the individual scores, to build a composite score that represents the global similarity between two tracks. We fundamentally have two ways of proceeding:

1. The first method consists in setting thresholds for the scores: if a score is above the threshold, both tracks are considered similar, otherwise they are considered dissimilar. The outputs of the considered classifiers are therefore binary (1 for similar, and 0 for dissimilar). Boolean combinations can then be applied to the binary outputs to take a final decision [80]. One advantage is that thresholded classifiers can be represented in a two-dimensional evaluation space, such as the ROC or PR spaces (see Section 3.3a). Combination techniques within the evaluation space, based on geometrical manipulation, can then be applied to retrieve a better classifier. One of such techniques is known as

Iterative Boolean Combination (IBC in Figure 7.1) and was provided by Kreich et al. [59]. This method applies all Boolean functions to combine the ROC curves corresponding to multiple classifiers. This creates a new curve in the ROC space that is above the initial curves, and thus outperforms any individual classifier. This method, however, requires the use of a specific evaluation space, the ROC space, which has been shown to not be optimal for the problem of CSI (see Section 3.2.1). It further requires an independent validation set to avoid over-fitting the data.

2. Thresholding the scores might lead to a potential loss of information, especially if the thresholds are not set accordingly. Furthermore, finding an optimal set of thresholds for a set of classifiers is not straightforward, and requires complex optimization algorithms. Consequently, other methods, based on *continuous scores*, can be used to keep the entire information. In this thesis, we investigate two ways of proceeding:

    (a) Given a query track that is compared to a reference database, the first method orders all comparison tracks from the highest score to the lowest one, thus creating an ordered ranking of the results. Each track is thus mapped to an integer position, between 1 for the most similar, and N for the least similar, with respect to the query. Considering different rankings obtained with several classifiers, we use *rank aggregation* techniques to combine such lists of results, by applying mathematical operations on the positions of the tracks in the rankings. Such techniques are interesting as they do not require any score normalization, because the tracks are mapped to integer positions. We investigate rank aggregation methods in the next section of this chapter.

    (b) Instead of mapping scores to integer positions in an ordered list of results, scores can be directly combined using simple probabilistic rules. However, as all systems do not return probabilities, techniques for mapping scores to interpretable posterior probabilities must be investigated. The resulting probabilities can next be combined using standard probabilistic rules. We investigates how such methods perform for CSI in Chapter 8.

7.2 Combining through Rank Aggregation

In a typical CSI scenario, a query issued by a user is sent to the the system to be compared to a database of reference tracks using some algorithm based one some feature. Each pairwise comparison between the query and each track of the database produces a similarity score. All comparisons between the query and all tracks thus produce a complete list of scores, called a *rating* [66]. Ordering the rating by decreasing order, thus from the most similar track to the least similar track, produces a *ranking* in which each reference track is assigned a fixed position in the range $[1, N] \cap \mathbb{Z}$, where N is the number of tracks in the database, and $\mathbb{Z}$ is the set of integers.

One can think of a ranking as a column of results, where the most similar tracks to the query are close to the top of the column (rank 1) and the least similar are close to the bottom of the column (rank N). We consider that a track A is ranked *above* track B if $rank(A) < rank(B)$. Therefore, a track that is ranked *high* (resp. *low*) in a ranking means that it is closer to rank 1 (resp. rank N).

In the first part of this thesis, we described 10 CSI systems that produce rankings by comparing queries to a reference database. As the systems are built upon different features and various matching algorithms, they do not produce the same rankings. As there is no clear correlation between the different features (see Table 7.1), the individual rankings should combined such that they all contribute to the performance of a combined CSI system.

In this chapter, we investigate how to combine such rankings in order to build a more robust list of results, with respect to a query. Combining ranking lists is the aim of *rank aggregation* techniques. The goal is *"to merge several ranked lists in order to build a single new superior ranked list"*, as claimed by Langville et al. [66]. The idea is not new, as it has been studied for decades in the context of political voting theory (social choice theory). Indeed, the study of rank aggregation goes back to 1770, when Jean-Charles de Borda suggested "election by order of merit", where each voter gives a preference order of the alternatives, with a score of 0 assigned to the least preferred, a score of 1 to the next least preferred, etc. [34]. As each voter produces such a list, one needs to aggregate them in order to determine the winner of the election. Later on, in 1785, the Marquis de Condorcet proposed that if there is some alternative, now known as the *Condorcet alternative*, that defeats every other in pairwise simple majority voting, then that alternative should be ranked first. We will elaborate more on the Condorcet alternative in the subsequent sections, as it proves to be effective when applied to CSI. More recently, the field of web search brought renewed interest in rank aggregation [33, 34, 86]. Indeed, the idea is that aggregating ranked lists produced by several search engines creates one parent list that takes advantage of the best properties of the child lists. This can easily be applied to CSI, as the ranking produced by each method can be seen as the result of a similarity search for cover songs, based on various music properties. Thus, if one list produced by a method based on *eg* timbre (Section 5.1.5) contains an outlier at the top of the produced ranking, then, after rank aggregation with other rankings, the effect of this outlier should be mitigated. Therefore, rank aggregation methods tend to smooth out the resulting ranking by removing outliers in the individual rankings. Note however that the ranking produced by

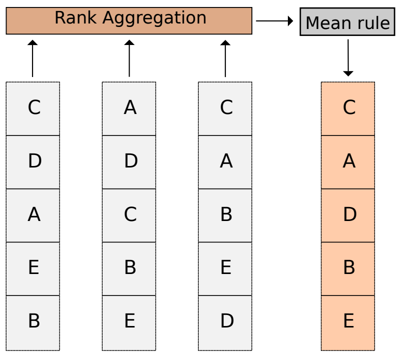


Figure 7.2: Rank aggregation of individual ranked lists of results.

aggregating input lists is only as good as the lists from which it is built. One cannot expect to create an accurate rank-aggregated list with a set of poor entry lists.

We begin with the description of three common rank aggregation methods, based on rather simple mathematical operations. Following that description, we elaborate our discussion on the Condorcet alternative, and show that while finding a Condorcet winner is considered an NP-hard problem, it can be approximated using a technique called *local-Kemenization*. We demonstrate that application to CSI produces interesting results. We continue next with hierarchical rank aggregation, which is a way of grouping methods together, before aggregating the resulting combination with other methods. This corresponds to attributing different ways to the methods. We show experimental results on the SHSD and demonstrates that hierarchical rank aggregation with applied local-kemenization produces an improved performance.

7.3 Rank aggregation methods

Rank aggregation can be performed using rather simple mathematical techniques, yet showing promising results when applied to CSI. Indeed, the typical way of aggregating lists is to consider the position of each track in each ranking, and apply mathematical operations such as the mean, the median, or the minimum of the ranks of each track in the initial rankings.

Rank aggregation shows a considerable advantage compared to other score based combination techniques. As it only considers the positions of the tracks in the input lists, it does not need to take into account the individual scores produced by the methods. Therefore, even if the scores obtained by individual methods are on different scales, the aggregation will not change, because the scores are automatically mapped to an integer rank in the interval $[1, N]$, where N is the number of tracks. Thus, there is no need for rescaling the values in a common range, or any other normalization technique. Rank aggregation is naturally calibrated and scale insensitive [86].

Figure 7.2 illustrates rank aggregation for three lists containing 5 items each. In the first list, C is at rank 1, in the second list, it is at rank 3 and finally, C is at rank 1 again in the third list. Using a mean aggregation rule, the final rank of C is 1.67, which places C at the rank 1. Other rules such as the *median* and the *minimum* rule can be applied and produce different rankings. We experimented with the three rules in this thesis and all of them show a significant improvement of the performance. Note that one advantage of such simple rules is that they are computationally cheap, as they consist in arithmetic operations on integer ranks. They further require none, or few parameters to set up.

When the input lists are large, ties can happen as several items can be ranked at the same position (using the mean rule for example). In that case, several policies can be adopted, as explained by Langville et al. [66]. One possibility is to pick items at random at a tie position and attributes a random position around that position. Other tie-breaking strategies involve giving priority to one of the input lists to select the final rank of an item.

7.3.1 Mean rule

The mean aggregation rule is computed as the mean of the ranks of the considered tracks in each input list. For a given track t being ranked in each list i at rank $r_i(t)$, the aggregated rank is computed as

$$r_{mean}(t) = \frac{1}{N} \sum_{i=1}^{N} r_i(t), \tag{7.1}$$

where $N = 10$ in our case, as we combine 10 systems. Table 7.2 displays the performance of the mean rule when applied to all 10 input systems (second row). For comparison, we also display the performance of the QMax system as a baseline.

From the results, one can observe that the performance is rather poor in terms of identification rate. Indeed, the mean rule decreases considerably the number of tracks identified in the top-1, 10 and 100. Consequently, the MRR and MAP metrics are also decreased, as they emphasize the performance near the top-1 (See Chapter 4). The only metric which brings a significant improvement is the MR, which means that the area under the RPL curve is decreased. Thus, the curve is lowered when applying the mean rule. As displayed in Figure 7.3, one can observe that the curve displays a better performance at lower pruning thresholds. Clearly the mean rule applied to all methods does not emphasize the upper right corner of the PL space, but rather the performance at lower thresholds, which makes it more useful for early pruning applications.

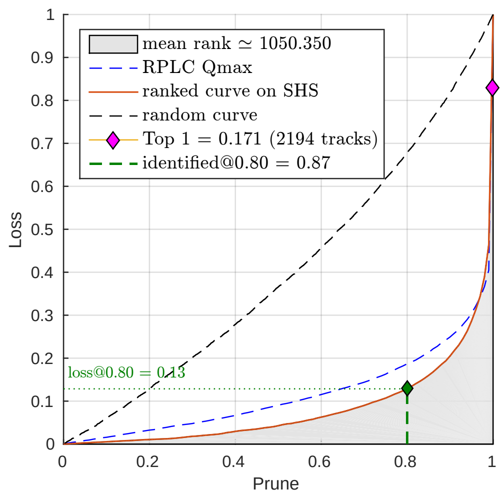


Figure 7.3: RPLC for 10 aggregated CSI systems using the mean rule.

	Top-1	Top-10	Top-100	MR	MRR	MAP
QMax (baseline)	$4,965$	$6,188$	$7,505$	$1,466$	0.42	0.24
Mean	$2,194$	$4,031$	$6,565$	$1,027$	0.220	0.119
Median	$2,669$	$4,355$	$6,593$	$1,077$	0.252	0.138
Minimum	930	$5,675$	$7,492$	$1,016$	0.177	0.093

Table 7.2: Performance of mean, median and minimum rules for single rank aggregation of 10 input rankings. Initial rankings contain 12,856 tracks.

7.3.2 Median rule

The median aggregation rule is computed as the median of the ranks of the considered tracks in each input list. For a given track t being ranked in each list i at rank $r_i(t)$, the aggregated rank is computed as

$$r_{median}(t) = \underset{i\in[1,10]}{\text{median}}(r_i(t)). \tag{7.2}$$

Performance in terms of standard metrics is displayed in Table 7.2 (third row). The first observation is that it behaves similarly to the mean rule compared to the baseline, with a decreased performance in terms of the upper right corner of the PL curve. Figure 7.4 displays the RPL curve for this rule applied to 10 CSI systems. Performance is better in general than the mean rule. However, at high pruning threshold, the performance is decreased.

7.3.3 Minimum rule

The minimum rule considers the minimum rank obtained for a track within the 10 input lists. Formally, for a given track t being ranked in each list i at rank $r_i(t)$, the minimum rule produces

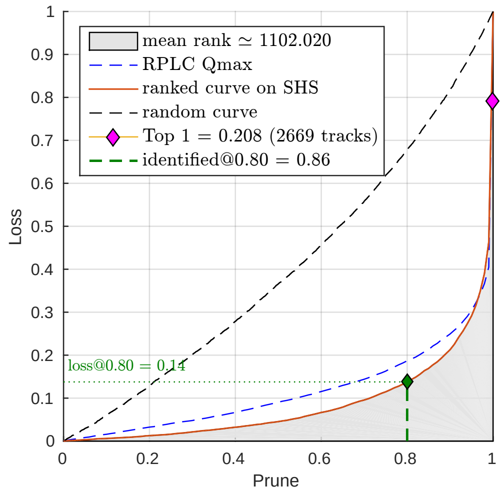


Figure 7.4: RPLC for 10 aggregated CSI systems using the median rule.

the minimum rank

$$r_{min}(t) = \min_{i \in [1,10]} (r_i(t)). \tag{7.3}$$

The rank $r_i(t)$ is an integer position in the range $[1, N]$, where the rank 1 corresponds to the first returned result (most similar track) and rank N is the last returned result.

This rule behaves a bit differently than both other rules, as can be shown in the last row of Table 7.2. One can indeed observe that the number of tracks identified at the first position (top-1 metric) is much lower than for the mean and median rules. However, in terms of top-10 and top-100, the performance is much closer to the baseline. This rule therefore brings a near top-1 performance close to the baseline, while lowering the curve at lower thresholds, as indicated by the MR metric, which is the smallest one achieved so far (an decreased MR corresponds to an increased performance). This is confirmed by the RPL curve, as displayed in Figure 7.5. Note that the MRR and MAP metrics are low because the performance in terms of *exact* top-1 is decreased, thus affecting the evaluation. We will see in the next section how to easily improve the rank aggregation to bring out tracks ranked low to the top of the list.

7.4 Optimal rank aggregation

While simple rank aggregation rules, such as mean, median and minimum, produce improved rankings, it is possible to add a successive step to the initial aggregation to further increase the performance. In 1785, the Marquis de Condorcet derived an important and desirable property of what an aggregated ranking should be. He suggested that *if there is some alternative that defeats every other alternative in pairwise simple majority voting, then that alternative should*

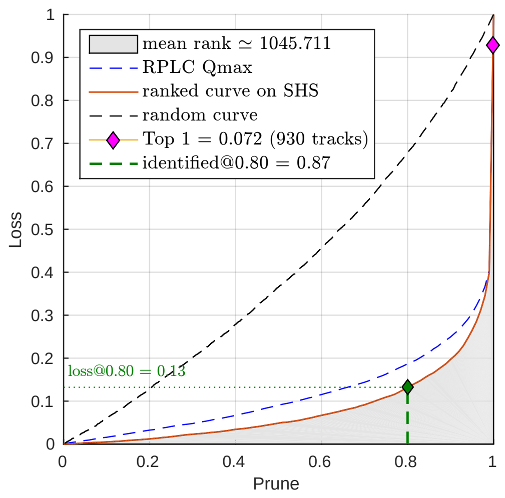


Figure 7.5: RPLC for 10 aggregated CSI systems using the minimum rule.

be ranked first. That first alternative is then known as the *Condorcet winner*. Later on, Truchon suggested a natural extension that is known as the *Extended Condorcet Criterion*. The latter mandates that if there is a partition (T, U) of the alternatives $\{1, \ldots, n\}$ such that for any $x \in T$ and any $y \in U$ the majority of the voters prefers x to y, then x should be ranked above y [106]. It turns out that, interestingly, none of the simple aggregation rules, discussed in the previous sections, ensures the election of the Condorcet winner [34].

In their work, Dwork et al. [34] describe an interesting method to demonstrate that the extended Condorcet Criterion can be achieved efficiently. They demonstrate in particular that an aggregation based on that principle has excellent "spam-fighting" properties. In this section, we explain the principles of the extended Condorcet principle and show that it can be algorithmically implemented efficiently. We further propose our own publicly available library to optimally aggregate rankings. In the next section, we apply the technique to the problem of CSI and demonstrate an improved performance.

7.4.1 Kemeny optimal aggregation

Kemeny optimal aggregations are interesting as they satisfy the extended Condorcet criterion. We first define the K-distance, necessary to understand the Kemeny optimal aggregations.

K-Distance.

Let π and σ be two partial rankings of $\{1, \ldots, n\}$ items. The K-Distance of π and σ, denoted $K(\pi, \sigma)$ corresponds to the number of pairs $i, j \in \{1, \ldots, n\}$ such that $\pi(i) < \pi(j)$ but $\sigma(i) > \sigma(j)$. If the pair (i, j) does not appear in the partial rankings, then that pair does not contribute to the K-distance. For two partial lists, $K(\pi, \sigma) = K(\sigma, \pi)$. If the rankings π and σ

are full permutations of the database, then K is a metric and is known as the $Kendall - tau$ distance. It corresponds to the number of transpositions a *bubble-sort* algorithm requires to turn π into σ.

In the case of CSI, we consider full permutations of the database, produced by each individual system. Therefore, the distance used to measure the distance between two rankings is the $Kendall - tau$ distance, denoted by K_τ, formally defined [66] by

$$K_\tau = \frac{n_c - n_d}{n(n-1)/2}, \tag{7.4}$$

where n_c is the number of concordant pairs and n_d is the number of discordant pairs. A pair of tracks (i, j) is concordant if i is ranked above j in both lists, and discordant otherwise. The denominator corresponds to the total number of pairs of n items in the lists.

SK, Kemeny optimal.

For a collection of partial (or full) lists $\tau_1, \ldots, \tau_k$ and a full list π, which could correspond to a first aggregation of the lists $\tau_1, \ldots, \tau_k$ using some of the rules described in Section 7.3, a permutation σ is a *Kemeny optimal* aggregation of lists $\tau_1, \ldots, \tau_k$ if it minimizes the sum of K_τ distances between π and each initial list $\tau_1, \ldots, \tau_k$ (also called the *aggregated Kendall measure*). Formally, we want to minimize [66]

$$SK(\pi, \tau_1, \ldots, \tau_k) = \sum_{i=1}^{k} K_\tau(\pi, \tau_i), \tag{7.5}$$

by modifying the full list π so that the aggregated measure decreases.

Although a Kemeny optimal aggregation satisfies the extended Condorcet criterion, finding such an aggregation is known to be NP-hard. It can be shown that this is the case from 4 complete lists to be aggregated [34]. It is therefore impossible to find such an optimal aggregation for our problem of combining 10 CSI systems. Furthermore, each of our systems consider a permutation of a database containing $12,856$ tracks, which can further slow down the aggregation. We turn therefore to an approximation of a Kemeny optimal ranking, called local Kemenization [34, 66], which we explain in the next section.

7.4.2 Local Kemeny optimal aggregation

In their work, Dwork et al. [34] introduce an efficient approach to rank aggregation that guarantees the satisfaction of the extended Condorcet criterion. They begin with any initial aggregation μ of $\tau_1, \ldots, \tau_k$, for example a mean rule, and then to compute what they call a "*local-Kemenization*" of μ, which produces an ordering σ that is consistent with μ while satisfying the requested criterion.

In the context of cover song identification, one can think of the rankings $\tau_1, \dots, \tau_k$ as the permutations of the search database, as returned by all CSI systems. In that case, $k \in [1, 10]$. For a given query, the $\tau_1, \dots, \tau_k$ correspond to lists of tracks from the search database, ordered from the most similar to the least similar. If a non-cover track is ranked high in most CSI systems, than no combination of these rankings will defeat that track. However, if that same song is ranked high in less than half rankings, the majority of the systems might prefer a "good" candidate song to the wrong one. In that case, all "wrong" tracks are the Condorcet losers and will occupy the bottom partition of any aggregated ranking that satisfies the extended Condorcet criterion. Similarly, assuming that "good" tracks are preferred by the majority of systems to the "wrong" ones, these will be the Condorcet winners, and will therefore be ranked higher.

Local Kemeny optimality

Based on the previous assumptions, the local Kemeny optimality can be defined as follows. A permutation π is a locally Kemeny optimal aggregation of lists $\tau_1, \dots, \tau_k$ if there is no permutation π' that can be obtained from π by performing a single transposition of an adjacent pair of elements and for which $SK(\pi', \tau_1, \dots, \tau_k) < SK(\pi, \tau_1, \dots, \tau_k)$. In other words, it is impossible to reduce the total distance to the τ's by flipping an adjacent pair.

Thus, the procedure for computing a local Kemeny optimal aggregation considers each pair of adjacent tracks in μ (the initial ranking) and verifies whether a swap will improve the aggregated Kendall measure. In practice, for two adjacent tracks (i, j) in μ, with i ranked above j, the procedure checks whether track j is ranked above track i in the majority of the τ_k. If it is the case, it swaps both tracks as it refines the aggregated list with a reduced aggregated K_τ distance. The procedure starts from the beginning of the list, and is repeated iteratively for all pairs of tracks, requiring $n - 1$ checks for an aggregated list of length n. Note that the consecutive swaps of the *Kemenization* process take into account earlier swaps.

We implemented our own version of the local Kemenization procedure and applied it to several initial rank aggregations to improve the performance of cover song identification. For implementation details, we refer the reader to our publicly available implementation of a rank aggregation C++ library[1]. We used that code for all experiments related to rank aggregation and local-Kemenization.

[1]`https://github.com/osmju/librag`

	Top-1	Top-10	Top-100	MR	MRR	MAP
QMax (baseline)	$4,965$	$6,188$	$7,505$	$1,466$	0.42	0.24
Mean	$2,194$	$4,031$	$6,565$	$1,027$	0.220	0.119
Mean + Kemeny	$2,683$	$4,215$	$6,590$	$1,026$	0.250	0.137
Median	$2,669$	$4,355$	$6,593$	$1,077$	0.252	0.138
Median + Kemeny	$2,947$	$4,437$	$6,596$	$1,076$	0.270	0.149
Minimum	930	$5,675$	$7,492$	$1,016$	0.177	0.093
Minimum + Kemeny	$3,956$	$6,132$	$7,591$	$1,011$	0.368	0.209

Table 7.3: Performance of mean, median and minimum rules for single rank aggregation of 10 input rankings, with applied local-Kemnization. Initial rankings contain 12,856 tracks.

7.5 Application to Cover Song Identification

7.5.1 Single aggregation

We applied local-Kemenization to the rules we experimented earlier (see Section 7.3). Using the same evaluation setup as for all the previous experiments, we first applied the mean, median and minimum rules to produce an initial ranking. This produces rankings that correspond to the performance displayed in Table 7.2. Taking the resulting ranking, we applied the local-Kemenization process, with respect to the 10 initial rankings as returned by each individual CSI system.

Table 7.3 displays the performance of each rule, with respect to the QMax baseline. For each rule, we recall the previous performance without applied Kemenization. From the results displayed in the table, one can observe that the local-Kemenization procedure is quite efficient to bring some tracks to the top of the final rankings. This is especially true for the Minimum+Kemeny rule, which allows to bring $3,035$ more tracks to the first position, thus approaching the baseline Top-1 performance. The improvement is less significant for the other metrics. Observe however how this rule achieves the lowest MR. Figure 7.6 displays the corresponding RPL curve, which indeed produces a lowered curve, compared to QMax, while being close to the QMax curve at higher thresholds.

Figure 7.7 displays the performance of the mean and median rules with applied Kemenization. Both curves have similar shapes and performance. Observe how the mean and median curves cross the QMax baseline curve around threshold 0.95. The overall curve is lowered, while degrading the performance in the upper right corner.

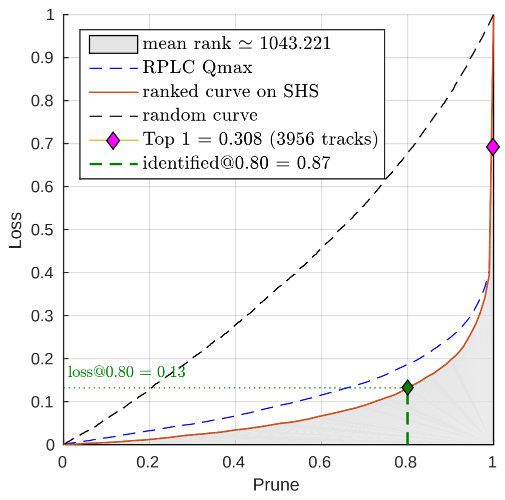


Figure 7.6: Local-Kemenization applied to the minimum rule, with 10 CSI systems taken as inputs. The performance is closer to the baseline, in terms of near top-1 performance (upper right corner). The performance at lower threshold is improved.

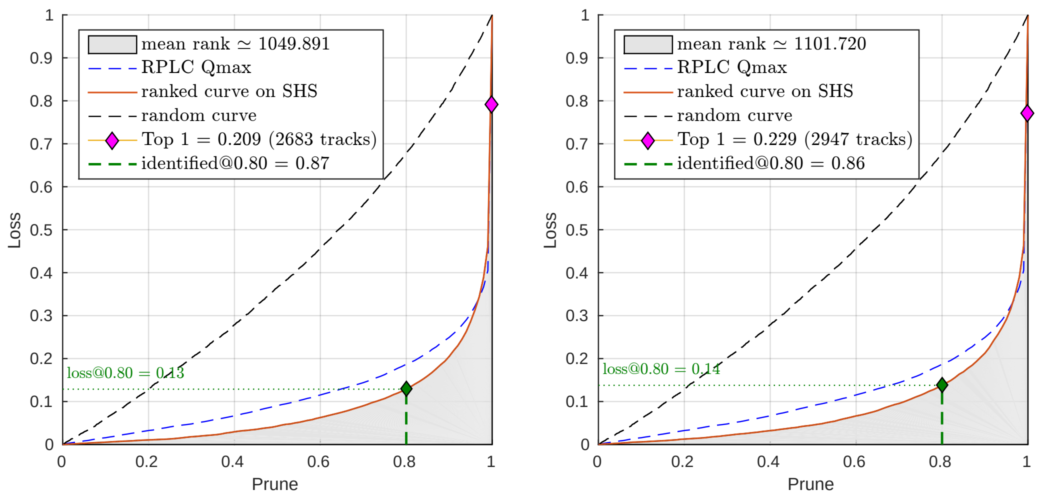


Figure 7.7: Local-Kemenization applied to the mean (left) and median (right) rules, with 10 CSI systems taken as inputs.

7.5.2 Adding hierarchization

Until now, we applied rank-aggregation to all 10 rankings returned by our CSI systems, considering the same importance for all systems. Aggregating all systems on the same level corresponds to giving the same weight to each method. However, based on the performance of each individual system (See Chapter 5), one can clearly observe that some methods, such as the one based on the tempo, duration or beats, perform worse than other more complex methods (2D-FTM, QMax, etc.).

In this section, we experiment with hierarchical rank aggregation [82]. We first consider CSI systems based on global and low-dimensional features. These are systems based on the beats, the duration, the tempo and the average chroma vector. Each of these systems are described in Chapter 5.

Hierarchical aggregation consists in first combining these four systems using one of the aforementioned aggregation rules. This produces a first aggregated ranking which is the combination of multiple global features together. The resulting combined system produces an increased performance, compared to the individual systems [82].

Once these four initial systems are aggregated, the resulting ranking is further combined with the remaining CSI systems, using the same, or another rule. As this is the final aggregation, we also apply local-Kemenization to bring more tracks to the top of the ranking. We experimented with several rules, for both combinations. Table 7.4 shows the performance in terms of each metric for the various combinations. We recall results from Table 7.3, and we add hierarchization (in italic in the table). In this experiment, we use the same rule for the four initial systems, and for the aggregation with the remaining systems. Thus, we have for example

$$\text{Hierarchical Mean} = \text{Kemenize}(\text{mean}(\text{mean}(C_1, \ldots, C_4), C_5, \ldots, C_{10})). \tag{7.6}$$

From the performance in the table, one can observe that rank aggregation rules, with local-Kemenization and hierarchization do not necessarily improve the near top-1 performance. However, it considerably lowers the RPL curve, as indicated by the MR metric, thus offering better identification rates at lower prune thresholds. The best trade-off in terms of performance was achieved using first the mean rule to aggregate the systems based on global features, then the minimum rule with applied local-Kemenization for the final aggregation:

$$\text{Combined ranking} = \text{Kemenize}(\text{min}(\text{mean}(C_1, \ldots, C_4), C_5, \ldots, C_{10})), \tag{7.7}$$

as pictured in Figure 7.8. Performance is displayed in the last row of Table 7.4. The corresponding RPL curve is shown in Figure 7.9 and displays a lowered curve, with respect

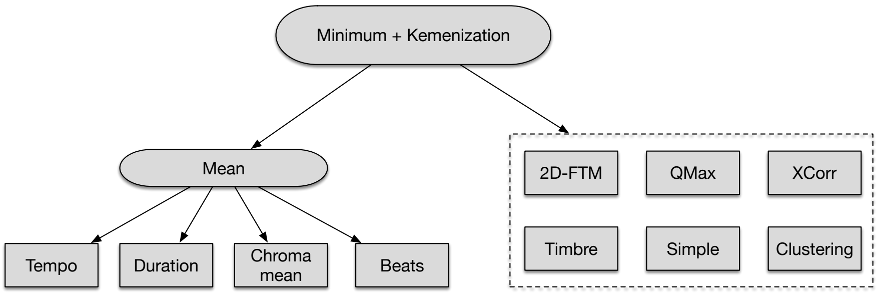


Figure 7.8: Hierarchical rank aggregation using a two-level combination. First, systems based on global features are aggregated using the mean rule, then the resulting ranking is combined with the remaining systems using a minimum rule, with applied local-Kemenization.

	Top-1	Top-10	Top-100	MR	MRR	MAP
QMax (baseline)	$4,965$	6,188	7,505	1,466	**0.42**	**0.24**
Mean + Kemeny	2,683	4,215	6,590	1,026	0.250	0.137
Hierarchical Mean	*3,588*	*4,915*	*6,922*	*979*	*0.32*	*0.174*
Median + Kemeny	2,947	4,437	6,596	1,076	0.270	0.149
Hierarchical Median	*3,625*	*5,087*	*7,056*	*1,049*	*0.32*	*0.18*
Minimum + Kemeny	3,956	6,132	7,591	1,011	0.368	0.209
Hierarchical Minimum	*3,853*	***6,383***	***7,796***	***971***	*0.37*	*0.21*
Kemeny-min(mean, others)	**4,187**	**6,433**	**7,790**	**983**	**0.39**	**0.22**

Table 7.4: Performance of mean, median and minimum rules for hierarchical rank aggregation of 10 input rankings, with applied local-Kemnization. The last row (in bold) first aggregates global systems with the mean rule, and combines the resulting ranking with the remaining systems using the minimum rule with local Kemenization. Initial rankings contain 12,856 tracks.

to the QMax baseline. In terms of Top-K metrics, this combination is close to the baseline, even better for the Top-10 metric. The MR is decreased by $33\,\%$, compared to QMax, thus considerably lowering the curve at lower thresholds. In terms of MRR and MAP, the performance is slightly decreased, but still close to the baseline.

7.6 Conclusion

In this chapter, we showed that rank aggregation is an interesting technique to combine rankings produced by different CSI systems that returns scores on different scales. Rank aggregation automatically maps the score to an integer rank and thus does not require any score normalization to combine the results.

Applied to CSI, we are able to produce an improved performance, especially at lower pruning rates, thus rendering the combined system interesting for early pruning applications. While the

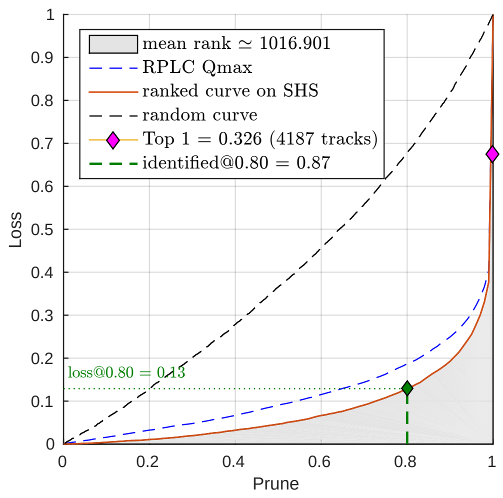


Figure 7.9: Performance of a hierarchical rank aggregation of 10 CSI systems. Four global CSI systems are first aggregated using the mean rule, then the resulting ranking is combined with the remaining systems using the minimum rule with applied local-Kemenization.

various rank aggregations we studied in this chapter tend to lower the RPL curve in general, they also decrease the performance at higher pruning thresholds, compared to our baseline algorithm, QMax. For all studied rules, we observe that the Mean Rank (MR) metric is decreased, thus lowering the RPL curve. We further realized that all aggregation rules produce a similar performance at pruning rates sitting in the range $[0.1; 0.9]$. In average, the loss rate at a pruning rate of 0.6 lies around 0.075, which corresponds to an identification of $93\,\%$ in the top-60%. Similarly, the aggregation rules produce a loss at a pruning rate of 0.8 lying in the range $[0.13; 0.14]$, which is lower than QMax.

While the performance stays stable at low pruning rates, applying local-Kemenization allows to brings many tracks to the top of the final ranking. This can be seen in particular for the *minimum* aggregation rule, which identifies 930 tracks in the top-1 without Kemenization, and up to $3,956$ tracks after local-Kemenization. The minimum rule with Kemenization produces the best performance when aggregating all 10 inputs systems on the same level.

In order to bring even more tracks to the combined system, we proposed to aggregate methods on different levels, thus applying weaker weights to the systems based on global features. The best performance was achieved by first aggregating four CSI systems based on global features using the *mean* aggregation rule. We then re-combined the resulting ranking with the remaining input systems, using the *minimum* rule with applied local-Kemenization. This produced an increased performance for the top-K metrics ($K \in [1, 100]$), as well as for the MRR and MAP metrics, as they are related to the Top-1. In particular, we achieved a low MR of 983, which considerably lowers the RPL curve.

Overall, rank aggregation techniques can be used to lower the RPL curve for CSI systems, while maintaining a stable performance near the top-1. Considering this, such combination rules should not be used to increase the direct identification rate, but rather for applications requiring early pruning strategies.

CHAPTER 8

Improving Combination through Probabilistic Rules

In this chapter, we consider methods for directly combining the similarity scores returned by all classifiers, in order to build a new combined similarity score, that takes into account all individual scores. The latter can then be used to produce a new ranking of the tracks of the database, with respect to the query. To do so, we investigate how to make use of combination rules based on the probability theory. In order to do so, one has to consider a CSI system as a binary classifier that maps a track to two possible classes: similar (class ω_+), or dissimilar (class ω_-), with respect to the query. Based on an observation vector, which can include any feature extracted from the music, the classifier returns a score that corresponds to the estimated probability to belong to one class. Let us denote the feature vector for a track t by $\mathbf{x} = o(t)$, where $o(t)$ is a function that extract relevant observations from t. The classifier returns a score

$$y = p(\omega_+ \,|\, \mathbf{x}), \tag{8.1}$$

which corresponds to the *estimated* probability to belong to the similar class (ω_+) given the observation $\mathbf{x}$. Note that in practice, few classifiers return probabilities. In our case of CSI, only the systems based on the ExtraTrees [48] learning algorithm (beats, tempo, duration and average chroma vectors, see Chapter 5) return probabilities. The other systems return a score, that can be a distance, or an accumulated similarity score, which is not a probability. However, it is in general possible to establish a relationship between the predicted scores and probabilities, as will be shown in this chapter.

Estimating the probability $p(\omega_+ \,|\, \mathbf{x})$ is difficult in practice. Indeed, using Bayes' theorem, one can write

$$p(\omega_+ \,|\, \mathbf{x}) = \frac{\pi_+ \, \rho_+(\mathbf{x})}{\pi_- \, \rho_-(\mathbf{x}) + \pi_+ \, \rho_+(\mathbf{x})}, \tag{8.2}$$

where ρ_+ and ρ_- denotes the probability density functions associated with the similar and dissimilar classes, respectively, in the *features space*. π_+ and π_- respectively correspond to the prior probabilities of the similar and dissimilar classes. We thus have $(\pi_+, \, \pi_-) = (p[\omega_+], \, p[\omega_-])$. From Equation 8.2, one can observe that a CSI system should estimate two probability density functions ρ_+ and ρ_-, which is a difficult problem, as the features space can be high dimensional, depending on the observation vector. An alternative would be to estimate a probability based on the one-dimensional score y predicted by a CSI system:

$$p(\omega_+ \,|\, y) = \frac{\pi_+ \, \rho_+(y)}{\pi_- \, \rho_-(y) + \pi_+ \, \rho_+(y)}, \tag{8.3}$$

which is easier to do because the probability density functions must be estimated in a one-dimensional space. In such a case, satisfactory methods exist, as will be explained in this chapter.

The process of converting a score returned by a classifier to a posterior probability is called *calibration*, and will be described in the next section of this chapter. Following calibration, any probabilistic combination rule can be used to build a combined similarity probability. In particular, we demonstrate in this chapter that a product combination rule can be easily derived, using the probability theory. We further introduce two other rules, the sum rule and the median rule, that respectively consider the averaging and the median of the initial probabilities. Following the description of these rules, we experiment with our 10 CSI systems, and demonstrate that a significant increase of the identification rate can be achieved by using the product rule.

We begin this chapter with a review of existing calibration methods, as well as the particular one we chose for our CSI application. The following section continues with a description of the combination rules, and an analysis of their behavior. The next section details the evaluation of each method on our SHS evaluation set, and a comparison of the performance, with respect to previous results. In particular, we show that probabilistic combination leads to the best performance achieved with our 10 initial CSI systems. Finally, the last section concludes this chapter.

8.1 Estimating probability estimates

8.1.1 Principles of calibration

In many classification applications, it is important to be able to interpret the scores returned by a classifier as posterior probabilities. Probabilities are indeed essentials in domains such as medical decision making, weather forecasting, fraud detection and risk analysis [38]. They are often desirable as they are more meaningful and easier to interpret than arbitrary scores. Gebel et al. [47] consider that probabilities should be indeed preferred to scores because:

1. a classifier is part of a decision process where decisions are associated with costs. If the costs differ between classes, it is desirable that the scores reflect the assessment uncertainty;
2. probability estimates simplify the comparison and combination of results coming from different classifiers. Indeed, it can be shown that under an assumption of independence, the probabilistic product combination rule produces an optimal performance [32].

Although several classification algorithms, such as bagged decision trees or neural networks, are accurate at estimating probabilities, most of them produce poor estimations of probabilities [77, 78]. In our problem of cover song identification, the similarity estimators can be seen as binary classifiers: they map a track to a *similar class* ω_+ or to a *dissimilar class* ω_-, with respect to a query q. Note that a classifier does not necessarily make use of a learning algorithm. As shown in the previous chapters, most of our estimators are based on heuristic matching algorithms that do not consider a learning phase. The CSI similarity estimators return an unnormalized score that is interpreted as the confidence an estimator has in its prediction: the higher the score is, the higher the probability that the given track is similar to a query. As such scores cannot be used with probabilistic combination rules, it is necessary to convert the output score to accurate posterior probabilities. Such a process is called *calibration*, and consists in mapping scores from classifiers to posterior probabilities $p(\omega_k \,|\, \mathbf{x})$, where ω_k is the class ($k \in \{+, -\}$), and $\mathbf{x}$ is the observation. The calibrated probability claims to cover the confidence that a track belongs to the particular class k. According to Elkan et al. [115], a classifier is well calibrated if the empirical class probability $p(\omega_k \,|\, c(\mathbf{x}) = y)$ converges to the returned score value $c(\mathbf{x}) = y$ as the number of classified examples goes to infinity, where c is the classifier (or CSI system in our case) taking the observation $\mathbf{x}$ as an input and returning the similarity score y. Calibration is thus important if we want the scores to be directly interpretable as the chance of membership in the class. As all the CSI systems studied in this thesis are based on different features and algorithms, we need calibration to map the scores to accurate posterior probabilities so that we can combine them using probabilistic rules.

Formally, given a classifier c that scores instances $c : X \to \mathbb{R}$ from the feature space X to real scores in $\mathbb{R}$, calibration creates a function $\mathcal{M}$ such that $\mathcal{M}(c(\mathbf{x})) \to [0,\, 1]$ is an estimate of the posterior probability that instance $\mathbf{x}$ belongs to the positive class ω_+. Several methods were suggested in the past for finding the mapping $\mathcal{M}$, such as methods based on parametric estimation [84], binning [114] or nonparametric Isotonic Regression [115].

The parametric approach, proposed by Platt [84], was initially provided as a mapping for Support Vector Machine (SVM) scores. It is motivated by the fact that the relationship between SVM scores and the empirical probabilities appears to be sigmoidal for many datasets [115]. The method consists in finding parameters A and B for a sigmoid function of the form $p\left(\omega_k \,|\, \mathbf{x}\right) = \frac{1}{1+e^{Ac(\mathbf{x})+B}}$ mapping the scores $c(\mathbf{x})$ into probability estimates such that the negative log-likelihood of the data is minimized. Note that this method was optimized for scores produced by SVMs only. The same method was applied to Naive Bayes classifiers by Bennet [7]. However, the sigmoidal shape does not appear to fit Naive Bayes scores as well as it fits SVM scores [115]. This method is therefore quite specific to a subset of methods. A similar approach based on a sigmoid mapping function was successfully applied to outliers detection by Gao et al. [46].

If the shape of the mapping function is unknown, one can resort to a non-parametric method such as binning, studied by Zadrozny et al. [114]. Binning sorts samples according to their score and the sorted set is divided into b subsets of equal size called bins. For each bin, lower and upper boundary scores are computed, and each example $\mathbf{x}$ is placed in a bin according to its score $c(\mathbf{x})$. The probability that $\mathbf{x}$ belongs to class k is estimated as the fraction of training examples in the bin that belong to k. However, binning requires the number of bins to be chosen by cross-validation, which is problematic when the dataset is small or highly unbalanced. The size of the bins and the position of the boundaries is also chosen arbitrarily, which might lead to poor probability estimates.

The Isotonic Regression method was also introduced by Zadrozny et al. [115] as an intermediary approach between sigmoid fitting and binning. It relies on the assumption that the mapping function that transforms scores to posterior probabilities is monotonically increasing (isotonic), with respect to the classifier score. Given a training set (y_i, g_i),where $y_i = c(\mathbf{x}_i)$ is the output of a classifier c for example $\mathbf{x}_i$ and $g_i \in \{0,\, 1\}$ is the true label, the method finds the isotonic function m such that

$$m = \arg\min_z \sum (g_i - z(y_i))^2, \tag{8.4}$$

where the arg min is taken over all isotonic functions [38]. The most studied algorithm for the isotonic regression problem is the *Pool Adjacent Violators* (PAV) algorithm. Fawcett et al. [38] gives a clear explanation of the algorithm. Given a set of training examples ordered by the scores assigned by the classifier, PAV first assigns a probability of one to each positive instance and a probability of zero to each negative instance and puts each instance in its own group. It then looks at each iteration at adjacent violators, which are adjacent groups whose probabilities locally increase rather than decrease. For such groups, it polls them and replaces their probability with the average of the group's values. It continues until the entire sequence is monotonically decreasing. The results produce a sequence of instances where each has a score and an associated probability.

Another approach, suggested by Bennett [9, 8], considers calibration via the Bayes theorem. At first scores s_+ for the positive class are split according to their true class (ω_+ or ω_-) in two groups so that probabilities $p(s_+ \,|\, \omega_k)$ for the score given the true class can be derived. The idea is to model the distribution of unnormalized scores rather than posterior probabilities [47]. The second step is to estimate the class priors π_k, $k = \{+, -\}$ and determine the score-conditional probabilities by application of the Bayes' theorem:

$$p(\omega_k \,|\, s_+) = \frac{\pi_k \, \rho(s_+ \,|\, \omega_k)}{\pi_- \, \rho(s_- \,|\, \omega_-) + \pi_+ \, \rho(s_+ \,|\, \omega_+)}. \tag{8.5}$$

Class priors can be easily estimated by class frequencies in the data set and the class-conditional densities are modeled as Asymmetric Laplace density, for which parameters are estimated for each class separately using Maximum Likelihood [8].

8.1.2 Our approach

We made use of an approach similar to the one performed by Bennett [9], based on the Bayesian probability theory. For each of our CSI systems, we need to obtain a posterior probability $p(\omega_+ \,|\, S = y)$, where S is a random variable taking the score value y, which is returned by the system. Figure 8.1 shows a Gaussian probability density function (PDF) for a hypothetical CSI system returning a similarity score in the range $[0,\, 100]$. The probability for a range of scores can be obtained as the area under the curve between two boundaries, as shown in the figure for boundaries 20 to 40. Therefore, one can write

$$p(\omega_+ \,|\, S = y) = \lim_{R \to 0} p(\omega_+ \,|\, S \in [y - R,\, y + R]) \tag{8.6}$$

Using Bayes' theorem, this expression can be developed as

$$p(\omega_+ \,|\, S = y) = \lim_{R \to 0} \frac{\pi_+ \, p(y \in [y - R,\, y + R] \,|\, \omega_+)}{p(y \in [y - R,\, y + R])}, \tag{8.7}$$

where the probability $p(y \in [y - R,\, y + R] \,|\, \omega_+)$ is given by the PDF, $\rho_+(y)$, displayed in Figure 8.1. If one needs the probability of a single score, thus with $R \to 0$, the area under the curve resumes to a simple rectangle, with a base equal to $2R$ and a height given by $\rho_+(x)$.

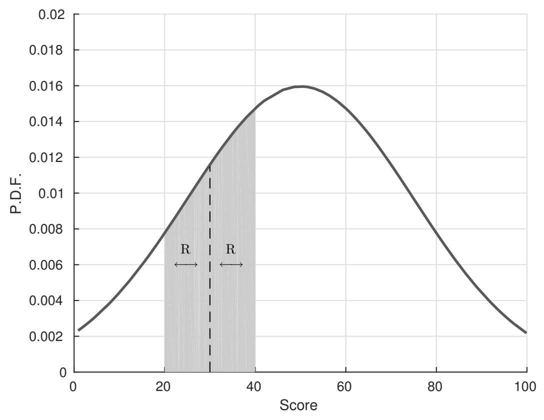


Figure 8.1: Calibrating scores to probabilities. When R tends to 0, the area between the PDF resumes to a rectangle.

The probability corresponding to a single score $S = y$ is therefore given by

$$p(\omega_+ \,|\, S = y) = \lim_{R \to 0} \frac{\pi_+ \, 2R \, \rho_+(y)}{\pi_- \, 2R \, \rho_-(y) + \pi_+ \, 2R \, \rho_+(y)}, \tag{8.8}$$

which can be simplified to

$$p(\omega_+ \,|\, S = y) = \frac{\pi_+ \, \rho_+(y)}{\pi_- \, \rho_-(y) + \pi_+ \, \rho_+(y)}, \tag{8.9}$$

in which π_+ and π_- correspond to the prior probabilities and ρ_+ and ρ_- correspond to the PDF of the similar and dissimilar scores, respectively. The last equation shows that one has to estimate the probability density functions corresponding to the distributions of the similar and dissimilar scores. To do that, we make use of a kernel density estimation method based on diffusion, as studied by Botev et al. [17], and extract the prior probabilities from the dataset, by computing the proportion of similar and dissimilar true labels in the dataset. Botev et al. [17] provides an adaptive kernel density estimation method based on the smoothing properties of linear diffusion. Their approach leads to a simple and intuitive kernel estimator with reduced asymptotic bias and mean square error. In addition, they provide a toolbox, which is straightforward to use to estimate probability density functions. Using that estimation method, we are able to compute a probability corresponding to a score returned by a CSI system. Figure 8.2 shows the obtained mapping functions for the scores returned by the 2D-FTM and QMax systems. Such mappings are obtained from the calibration method to convert similarity scores to posterior probabilities. One can observe that the 2D-FTM mapping function is quite linear, while the mapping to probabilities for the QMax method follows a sigmoidal shape. To assess the performance of the mapping function, we display in Figure 8.3 calibration plots for both mappings. The calibration plot represents the proportion of true positive samples

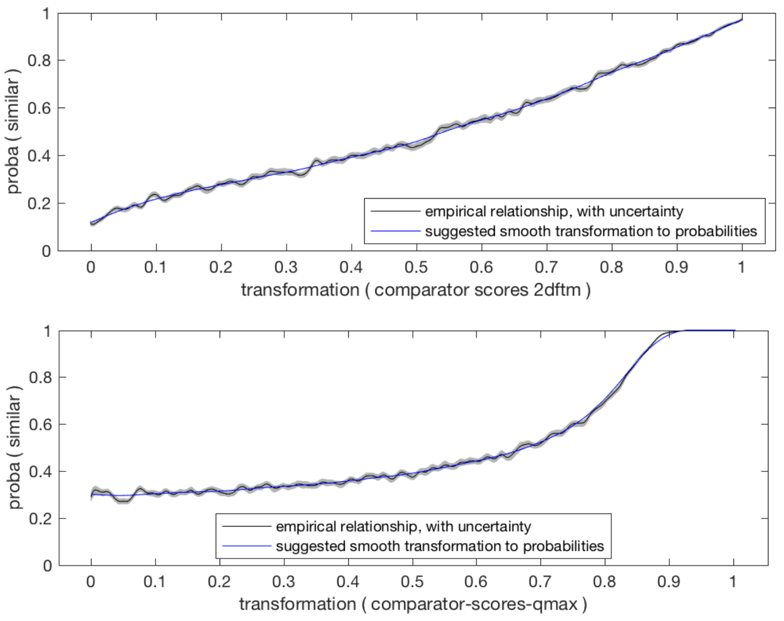


Figure 8.2: Mapping for the 2D-FTM system (top) and the QMax system (bottom). The figures represent the mapping between the scores returned by the systems and the corresponding posterior probabilities.

corresponding to the calibrated probability. For instance a well calibrated (binary) classifier should classify the samples such that among the samples to which it gave a calibrated probability value close to 0.8, approximately 80% of the samples actually belong to the positive class. The ideal calibration curve is thus represented by a diagonal straight line, as shown in Figure 8.3. One can observe that the resulting mappings for the 2D-FTM and QMax methods follow the diagonal ideal curve, which tends to confirm that the applied calibration method is effective.

8.2 Combining rules

Having now a way to map scores to posterior probabilities, we have the possibility to combine probabilities using several standard fusion rules, as provided in the literature. These rules are part of a theoretical framework, proposed by Kittler et al. [64], that we summarize below.

The problem of CSI can be considered as a pattern recognition problem where a pattern Z, which corresponds to a musical track of the database, should be assigned to one of k classes ω_k. In the case of CSI, we have two classes: either the track is similar to the query, or the track is not similar to the query, so $\omega_k = \{\omega_+, \omega_-\}$. Let us assume that we have N classifiers, each representing the pattern Z by a distinct measurement vector, or features vector. In our case, $N = 10$, and the measurement vectors correspond to the chroma features, the tempo,

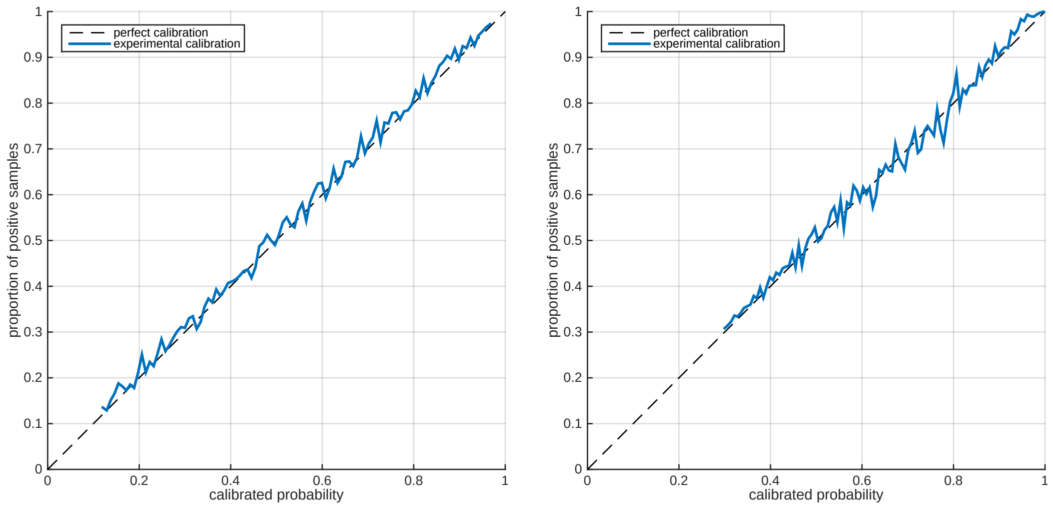


Figure 8.3: Calibration plots for the 2D-FTM (left) and QMax (right) systems. Calibration plots displays the proportion of positive samples for all calibrated probabilities.

the MFCC features, etc. We denote the feature vector used by the ith classifier by $\mathbf{x_i}$. In the feature space, both our classes ω_k are modeled by the probability density function $p\left(\mathbf{x}_i|\omega_k\right)$, with an a priori probability of occurrence $p\left(\omega_k\right)$.

According to the Baye's theorem, given the measurement $\mathbf{x_i}$, $i = 1, \ldots, N$, the pattern Z should be assigned to the class ω_k if the posterior probability for that class is maximum. So Z is mapped to ω_k if

$$p(\omega_k \,|\, \mathbf{x_1}, \ldots, \mathbf{x_N}) = \max_k p(\omega_k \,|\, \mathbf{x_1}, \ldots, \mathbf{x_N}). \tag{8.10}$$

So according to this rule, to take advantage of all the available information to reach a decision, the probabilities should be computed by considering all measurements simultaneously. One can observe that this is not a practicable proposition, as it implies estimating probabilities in a high dimensional space, by considering all features spaces. It would further require a large computational power to derive joint probability density functions in a high dimensional space.

In this section, we show that it is possible to simplify the above rule and express it in terms of the decisions taken by the individual classifiers, exclusively exploiting the information brought by the measurement vector $\mathbf{x_i}$. Indeed, in each of the N feature spaces, a classifier can be constructed to approximate the true class probability $p(\omega_k \,|\, \mathbf{x_i})$, $i = [1, \ldots N]$ in that feature space:

$$c_i^k(\mathbf{x_i}) = p\left(\omega_k \,|\, \mathbf{x_i}\right). \tag{8.11}$$

An accurate combination rule makes use of these classifiers $c_i^k(\mathbf{x_i})$, which correspond to our individual CSI systems, to approximate $p\left(\omega_k|\mathbf{x_1}, \ldots, \mathbf{x_N}\right)$ as best as possible. We focus in particular in this section on three rules. The first one is a product rule, which can be directly derived theoretically, under an assumption of independence, using the Bayes' theorem. The second one is a sum rule, which corresponds to averaging the probability estimates returned by each classifier. Finally, the last rule we investigated is based on percentiles, specifically the median rule.

8.2.1 Product rule

The probabilistic product combination rule assumes that the representations used by each classifier are conditionally independent [64]. In other words, in the measurement space and under that independence assumption, we can write

$$p(\mathbf{x_1}, \ldots \mathbf{x_N} \,|\, \omega_+) = \prod_{t=1}^{N} p(\mathbf{x_t} \,|\, \omega_+), \tag{8.12}$$

$$p(\mathbf{x_1}, \ldots \mathbf{x_N} \,|\, \omega_-) = \prod_{t=1}^{N} p(\mathbf{x_t} \,|\, \omega_-). \tag{8.13}$$

Note that in practice, this is a rather strong assumption that cannot always be justified. For example, in the case of CSI, knowing the number of beats and the tempo of a song could allow one to determine the duration of a song, which shows some dependency between these features spaces. Note however that such dependency was not clearly pointed out in the correlation matrix displayed in Chapter 7. Furthermore, under such independence hypothesis, it is possible to derive an optimal combination rule that should be experimented. Finally, in practice,even if the independence assumption is often not respected in combination problems, it can be shown that the product rule does produce improved performance [64, 102, 103]. According to Kittler [64], for many applications, the product rule will provide an adequate and workable approximation of the reality which may be more complex.

Let us denote the a priori probabilities of both classes $(p\,(\omega_+)\,,\, p\,(\omega_-)) = (\pi_+,\, \pi_-)$. Using the Bayes' rule, one can write

$$p\,(\omega_+ \,|\, \mathbf{x_1}, \ldots, \mathbf{x_N}) = \frac{p\,(\mathbf{x_1}, \ldots, \mathbf{x_N} \,|\, \omega_+)\; \pi_+}{p\,(\mathbf{x_1}, \ldots, \mathbf{x_N})}. \tag{8.14}$$

The denominator of this equation can be expressed as an unconditional joint probability by

$$p\,(\mathbf{x_1}, \ldots, \mathbf{x_N}) = \sum_{j=1}^{k} p\,(\mathbf{x_1}, \ldots, \mathbf{x_N} \,|\, \omega_j)\; p\,(\omega_j)\,,\; k = 2. \tag{8.15}$$

Using the independence assumption (Equation 8.12) and Equation 8.15 in Equation 8.14, we obtain

$$p\left(\omega_+ \,|\, \mathbf{x_1}, \ldots, \mathbf{x_N}\right) = \frac{\pi_+ \prod_{i=1}^{N} p\left(\mathbf{x_i} \,|\, \omega_+\right)}{\sum_{j=1}^{k} p\left(\omega_j\right) \prod_{i=1}^{N} p\left(\mathbf{x_i} \,|\, \omega_j\right)}. \tag{8.16}$$

In terms of posterior probabilities, as computed by each classifier, we obtain

$$p\left(\omega_+ \,|\, \mathbf{x_1}, \ldots, \mathbf{x_N}\right) = \frac{\pi_+ \prod_{i=1}^{N} \frac{p(\omega_+ \,|\, \mathbf{x_i})}{\pi_+} p\left(\mathbf{x_i}\right)}{\pi_- \prod_{i=1}^{N} \frac{p(\omega_- \,|\, \mathbf{x_i})}{\pi_-} p\left(\mathbf{x_i}\right) \,+\, \pi_+ \prod_{i=1}^{N} \frac{p(\omega_+ \,|\, \mathbf{x_i})}{\pi_+} p\left(\mathbf{x_i}\right)}, \tag{8.17}$$

which corresponds to the product rule. Note that posterior probabilities in the products in Equation 8.17 are estimated by each individual classifier $c_i^+(\mathbf{x_i}) = p\left(\omega_+ \,|\, \mathbf{x_i}\right)$. Thus, taking the output of each CSI system and applying the product rule produces the combined probability of similarity.

Behavior

We applied the product rule to all probability estimates returned by each CSI system. Prior to combining, we first mapped the scores returned by each method to a probability using the technique described in Section 8.1.2. For each track of the database and each method, we computed the combined probability that a track is similar to a query or not, for all queries. Varying the probability threshold, one can obtain an RPL curve for the resulting combination. Figure 8.4 shows the behavior of the product rule for a subset of classifiers at randomly sampled thresholds. The resulting combined system is shown in black at different thresholds, while the initial systems are shown in light gray. One can observe that the product rule produces a significant improvement, with respect to all other methods. The general behavior seems to correspond to a global lowering of the curve, although the improvement at low thresholds, up to 0.2, is not significant. In the range $[0.4,\, 1.0]$ however, the combination produces an impressive improvement.

8.2.2 Sum rule

Several research works propose to use a *sum rule*, which corresponds to averaging the probability estimates returned by all classifiers. However, although the product rule can be derived from conditional probabilities through the Bayes' rule, one cannot easily derive a sum rule using the probability theory. Several authors [32, 103] justify the sum rule for classifiers that are used on identical representations. In that case, all classifiers estimate the same class posterior probability, as they are correlated together. The sum rule is then applied to suppress the errors in the probability estimates, by averaging the outputs of the classifiers.

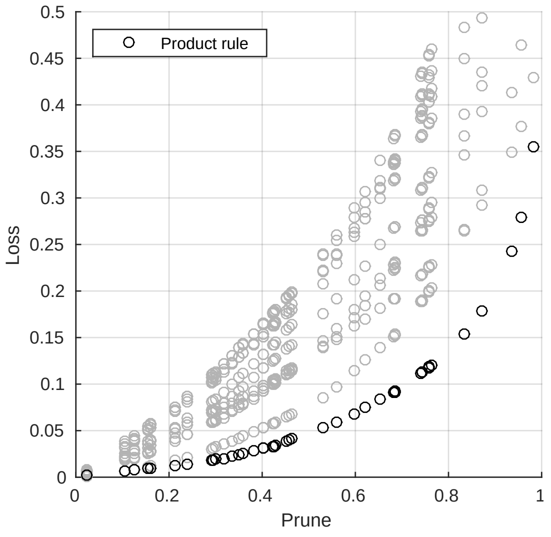


Figure 8.4: Behavior of the product rule (in black) when applied to 10 calibrated CSI systems (in light gray).

In their theoretical combining framework, Kittler et al. [64] claim that in some applications, it is reasonable to assume that the posterior probabilities computed by all classifiers will not deviate dramatically from the prior probabilities. As an example, they consider cases where the available observational information is highly ambiguous due to high levels of noise. In that case, the posterior probability can be expressed as

$$p\,(\omega_k\,|\,\mathbf{x_i}) \,=\, p\,(\omega_k)\,\left(1\,+\,\delta_{ki}\right), \tag{8.18}$$

with $\delta_{ki} \ll 1$ corresponds to the small deviation. In that case, averaging the outputs will smooth out the error and produce a better probability estimate. Note however that this is quite a severe assumption.

Let us denote the posterior probability returned by each classifier i by $y(\mathbf{x_i})$. Then the sum rule is given by the following equation:

$$y(\mathbf{x_1}, \ldots, \mathbf{x_N}) \,=\, \frac{1}{N} \sum_{k=1}^{N} y(\mathbf{x_k}). \tag{8.19}$$

Behavior

Similarly to the product rule, we applied the sum rule to our 10 calibrated CSI systems. Figure 8.5 displays the performance of the combined system in black, versus original CSI systems in gray. The thresholds are exactly the same than for the product. One can observe that the sum rule behaves very closely to the product rule. The difference is barely readable

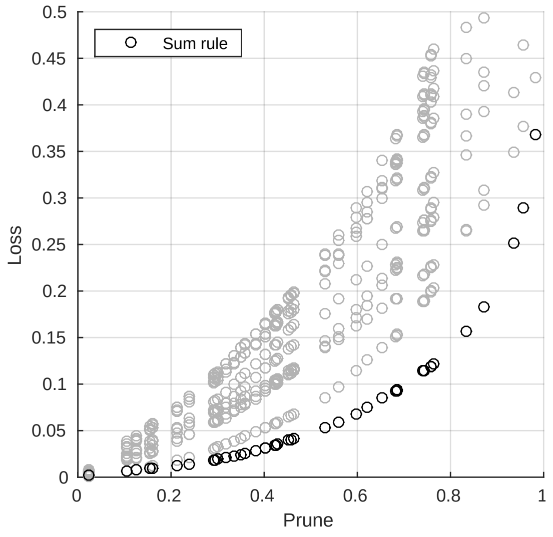


Figure 8.5: Behavior of the sum rule when applied to 10 calibrated CSI systems.

on such plots. Approaching the top-1 threshold (on the right hand side of the figure), we can observe that the product rule performs slightly better than the sum rule, which will probably lead to a decreased performance near the top of the returned list of results. However, overall both curves are similar.

8.2.3 Median rule

The sum rule corresponds to computing the average of the individual posterior probabilities returned by the classifiers. Thus the rule assigns a pattern to the class for which the average posterior probability is maximum. If any classifier returns a probability for a class that corresponds to an outlier, it will affect the average which might in turn lead to a wrong decision. A well known alternative to the average is the median, which is considered a robust estimate of the mean. Therefore, it could be more interesting to base the combined decision on the median rather than on the mean rule.

The median rule has not been much used in the literature. However, Kittler et al. [64] discuss it in their framework, as well as other rules such as the minimum or maximum probability rules, that we do not consider in this thesis.

Behavior

Figure 8.6 shows the behavior of the median rule for some classifiers selected at fixed thresholds. One can observe that, in opposition to the product and sum rules, the median rule is clearly the one that produces the smallest improvement. For some points even, the resulting classifier is worse than the individual classifiers (one can observe the top-right black point in the figure).

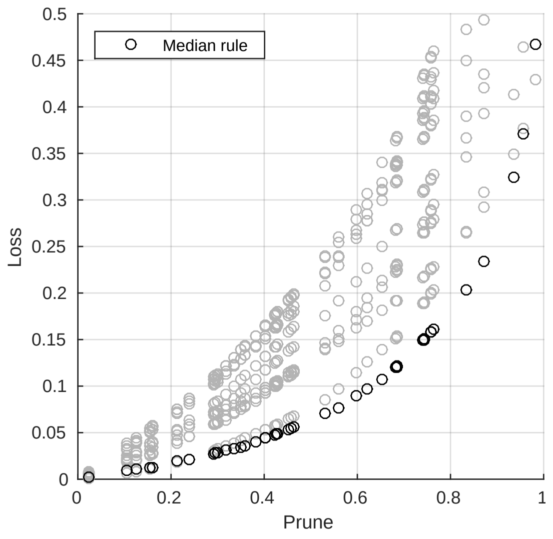


Figure 8.6: Behavior of the median rule when applied to 10 calibrated CSI systems.

8.3 Performance of combined CSI systems

Having applied each combination rule to our ten CSI systems, we evaluate in this section the resulting performance on our evaluation set, described in Chapter 4. We begin by applying the combination to all systems and display the performance in terms of RPLC and metrics. We next consider a subset of systems and demonstrate that by removing some classifiers from the pool of systems, we further improve the performance.

8.3.1 Using all systems

The combination rules described in the previous sections were applied to our subset of the SHS dataset, containing $12,856$ tracks, as explained in Chapter 4. To apply the combination practically, one has to take as an input the probabilities of similarity returned by each CSI system, for all pairs of tracks. It is then a simple matter of applying the requested rule to evaluate its performance. Figure 8.7 displays the RPL curves corresponding to the product and sum rules. Figure 8.8 shows the RPL curve obtained with the median rule. The figures also display the performance of the QMax algorithm (in dashed blue), which can be considered as state-of-the-art.

An analysis of the curves at first sight indicates that all of the combination rules definitely lower the curve, with respect to QMax, showing an important increase of the identification rate at low thresholds (up to a pruning of 0.8 approximately). One can observe that the identification on the right side of the curves is not so significant, compared to QMax. Indeed, QMax already

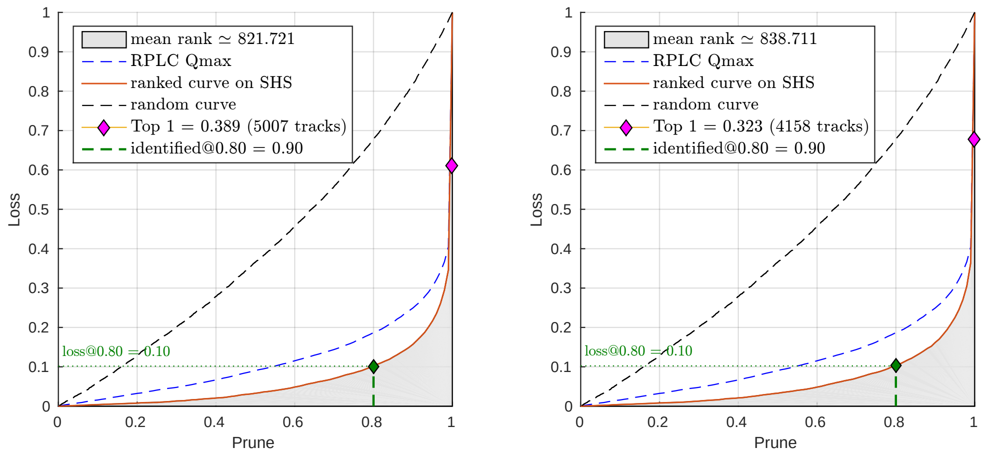


Figure 8.7: Performance of the probabilistic rules when applied to 10 CSI systems. RPLC curves are computed on the SHS dataset. The top-left figure is the *product* rule and the top-right figure corresponds to the *sum* rule. The blue dashed curve corresponds to the performance of the QMax system, which is considered state-of-the-art.

Method	Top-1 ↑	Top-10 ↑	Top-100 ↑	MR ↓	MRR ↑	MAP ↑
QMax (Baseline)	4,965	6,188	7,505	1,466	0.42	0.24
Rank Aggregation	3,956	6,132	7,591	1,011	0.37	0.21
Product	**5,007**	**6,525**	**8,197**	**787**	**0.43**	**0.25**
Improvement	+ 0.8 %	+ 5.4 %	+ 9.2 %	+ 46 %	+ 2.3 %	+ 4.2 %
Sum	4,158	5,939	7,925	806	0.37	0.21
Median	2,598	4,189	6,473	1058	0.24	0.14

Table 8.1: Performance of each combination rule when considering all 10 CSI systems. The product rule produces the best performance, followed by the sum rule. The improvement is quantified between the product rule and the baseline (QMax).

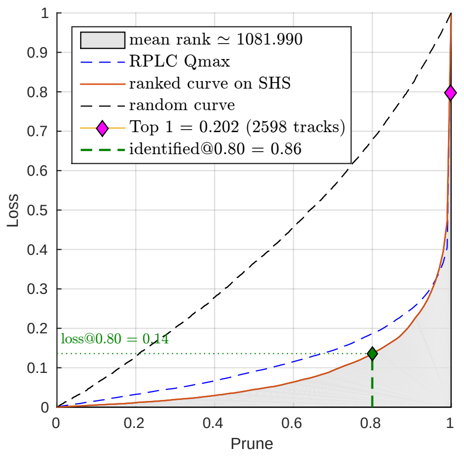


Figure 8.8: Performance of the median probabilistic rule.

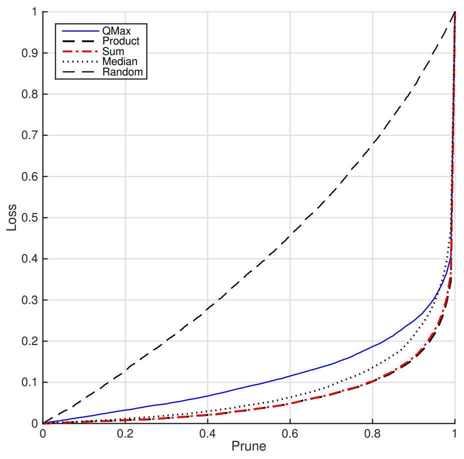


Figure 8.9: Comparison between three probabilistic combination rules and QMax. Observe how the product and sum rules behave similarly. One can also observe that the median rule starts to degrade the performance at higher thresholds.

has a high identification rate near the top-1, and it looks like the probabilistic rules do not improve much the performance in that area. Table 8.1 gives the performance with respect to the usual metrics. From the table, one can observe that the most efficient rule is clearly the product rule, followed by the sum rule and the median rule. For the product rule, numerical results confirm that the improvement near the top-1 is not so significant, with an increase of identification in the top-1 of only $0.8\,\%$. Note that this improvement increases in the top-10 and top-100, respectively with improvement of $5.4\,\%$ and $9.2\,\%$ in terms of identification rate. The MR metrics sees a quite significant improvement of $46\,\%$. This means that the area under the RPL curve is clearly lowered globally, compared to QMax. As the MRR and MAP metrics emphasizes the performance near the top-1, the improvement is not so significant. Observe however that the MAP is increased by $4.2\,\%$, which is interesting, knowing that the MAP considers all versions with respect to a query. This means that applying the product rule pushes some tracks to the top of the final list of results.

The sum rule behaves similarly to the product rule, with a less significant improvement with respect to the metrics, however. Figure 8.9 shows the RPL curves of the three rules on the same plot. One can observe that the sum and product produce indeed a similar curve. The median rule however, is clearly the worst one, with a performance worse than QMax at some thresholds points. In particular, the MRR and MAP metrics are the lowest ones. The MR is the highest one, also much smaller than the baseline, this indicating that the curve is indeed lowered. Up to a pruning rate of 0.4, the median behaves similarly to the product and sum rules. Above that threshold, the performance is decreased, which does not make it an interesting rule.

C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
2D-FTM	Chr-Mean	Beats	Duration	Timbre	Clustering	Tempo	Xcorr	Qmax	SiMPle

Table 8.2: Mapping between systems names.

8.3.2 Using a subset of systems

Instead of considering all 10 CSI systems for a probabilistic combination, we experimentally tried to find out if there is a subset of the systems that might improve the overall performance. To achieve that, we produced 1024 boolean strings corresponding to all possible subsets of 10 systems (2^{10} possible combinations) and studied whether some combinations perform better than others. To avoid over-fitting, we chose a subset of CSI systems that is optimal with respect to all combination rules and all evaluation metrics. We achieved that by ordering the subsets with respect to all metrics and rules. Doing that allows to compute a rank for each subset, with respect to the rules and metrics. The final ranking of the subsets is performed by computing an average rank for each set.

Table 8.3 displays the Top-20 subsets that produce the best performance. One can observe that systems C_1, C_5 and C_9, respectively corresponding to the 2D-FTM, Timbre and QMax systems, are always selected. This is not surprising for 2D-FTM and QMax, however, it clearly demonstrates that the CSI system based on timbre, proposed by Tralie et al. [105] contributes significantly to the overall performance. One can also observe that systems C_3 and C_6, corresponding to the beats and clustering systems are seldom included in the top-20 preferred subsets. Table 8.2 shows the correspondence between the numbers and classifier's names in these tables. Table 8.4 shows the corresponding metrics values for the Top-20 best subsets. Note that this table shows that considering some systems in the combination actually degrades the performance. This means that in a combination scheme, some classifiers might decrease the overall performance, while they should be bringing more information. Note however that this loss of performance is highlighted here with respect to the selected metrics only.

Table 8.5 shows the performance of each combination rule applied to the subset highlighted in Table 8.4. In particular, we drop systems C_2 and C_3, which seems to produce the best improvement. Performance are similar for all subsets in the Top-20. We select as the final system the combination that performs best in terms of Top-1, highlighted in bold in Table 8.4, as this metric represents the performance in terms of direct identification. Note that this selected subset corresponds to the performance on our specific database. The general tendency shows that we should only drop C_2 and C_3, which corresponds to the first line of Table 8.4. We voluntarily selected the best system with respect to the Top-1, which performs much better, which respect to the QMax baseline. The product rule is again the most efficient, with an

selected classifiers										ranks (/1023)									
										sum, w.r.t			product, w.r.t			median, w.r.t			mean
C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	MR	MRR	MAP	MR	MRR	MAP	MR	MRR	MAP	rank
V	V		V	V		V	V	V	V	4	31	29	4	39	33	33	49	32	28.22
V	V		V	V		V	V	V		10	44	36	15	44	35	80	88	86	48.67
V			V	V	V	V	V	V	V	52	27	31	68	45	47	53	77	67	51.89
V	V		V	V		V		V	V	3	38	47	2	35	40	31	147	133	52.89
V	V		V	V			V	V		7	88	77	9	111	94	39	29	27	53.44
V	V		V	V			V	V	V	2	68	62	3	120	99	46	60	34	54.89
V	V	V		V		V	V	V	V	76	29	33	92	47	50	128	52	33	60.00
V	V			V		V	V	V	V	45	46	46	48	87	85	108	47	31	60.33
V	V		V	V		V		V		6	67	72	7	34	34	41	133	150	60.44
V	V	V	V	V			V	V	V	16	58	60	19	90	82	145	66	44	64.44
V	V			V		V	V	V		77	59	58	93	70	79	117	31	28	68.00
V			V	V			V	V	V	176	16	12	184	25	25	156	22	16	70.22
V	V	V	V	V		V	V	V	V	26	36	35	27	53	45	158	150	114	71.56
V	V		V	V				V	V	1	92	93	1	117	101	4	117	137	73.67
V				V				V	V	205	9	19	216	48	43	106	11	12	74.33
V			V	V				V	V	168	11	15	180	14	14	155	72	48	75.22
V	V			V		V		V	V	41	66	83	45	81	84	28	118	135	75.67
V			V	V		V		V	V	182	4	4	189	1	1	161	79	63	76.00
V	V	V	V	V		V		V	V	18	64	65	21	42	42	144	158	153	78.56
V			V	V	V		V	V	V	40	65	67	52	124	122	93	82	76	80.11

Table 8.3: Ranking of the 20 best subsets of CSI systems that produce the best performance with respect to each combination rule and each evaluation metric. Some subsets dropping CSI systems are ranked better than combinations considering all systems.

selected classifiers										Scores											
										sum, w.r.t				product, w.r.t				median, w.r.t			
C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	MRR	MAP	MR	TOP-1	MRR	MAP	MR	TOP-1	MRR	MAP	MR	TOP-1
V	V		V	V		V	V	V	V	0.40	0.23	786.81	4453.00	0.45	0.26	773.71	5288.00	0.27	0.15	1032.85	2842.00
V	V		V	V		V	V	V		0.39	0.23	798.55	4396.00	0.45	0.26	783.76	5284.00	0.26	0.14	1069.52	2706.00
V			V	V	V	V	V	V	V	0.40	0.23	821.15	4498.00	0.45	0.26	808.27	5306.00	0.26	0.14	1051.55	2748.00
V	V		V	V		V		V	V	0.39	0.22	784.20	4387.00	0.45	0.26	771.89	5291.00	0.24	0.13	1030.40	2577.00
V	V		V	V			V	V		0.38	0.22	793.61	4297.00	0.45	0.26	779.90	5184.00	0.27	0.15	1037.55	2779.00
V	V		V	V			V	V	V	0.39	0.22	784.16	4337.00	0.45	0.26	771.96	5180.00	0.26	0.15	1041.75	2846.00
V	V	V		V		V	V	V	V	0.40	0.23	829.82	4468.00	0.45	0.26	813.46	5278.00	0.27	0.15	1105.35	2844.00
V	V			V		V	V	V	V	0.39	0.22	817.46	4424.00	0.45	0.26	803.04	5238.00	0.27	0.15	1083.81	2850.00
V	V		V	V		V		V		0.39	0.22	793.20	4305.00	0.45	0.26	779.61	5284.00	0.25	0.13	1038.94	2476.00
V	V	V	V	V			V	V	V	0.39	0.22	802.66	4348.00	0.45	0.26	787.45	5204.00	0.26	0.15	1118.35	2841.00
V	V			V		V	V	V		0.39	0.22	829.89	4358.00	0.45	0.26	813.66	5265.00	0.27	0.15	1095.20	2780.00
V			V	V			V	V	V	0.40	0.23	879.25	4539.00	0.45	0.26	872.43	5337.00	0.28	0.15	1129.78	3033.00
V	V	V	V	V		V	V	V	V	0.39	0.23	807.35	4409.00	0.45	0.26	790.89	5261.00	0.24	0.14	1131.44	2609.00
V	V		V	V				V	V	0.38	0.22	781.42	4263.00	0.45	0.26	770.04	5173.00	0.25	0.13	989.39	2502.00
V				V				V	V	0.40	0.23	917.78	4553.00	0.45	0.26	913.12	5301.00	0.28	0.16	1083.44	2896.00
V			V	V				V	V	0.40	0.23	871.34	4541.00	0.46	0.26	866.94	5383.00	0.26	0.15	1129.32	2827.00
V	V			V		V		V	V	0.39	0.22	815.59	4351.00	0.45	0.26	802.12	5234.00	0.25	0.13	1026.56	2502.00
V			**V**	**V**		**V**		**V**	**V**	**0.41**	**0.24**	**885.81**	**4611.00**	**0.46**	**0.27**	**878.73**	**5460.00**	**0.26**	**0.14**	**1132.61**	**2787.00**
V	V	V	V	V		V		V	V	0.39	0.22	803.79	4329.00	0.45	0.26	788.17	5287.00	0.24	0.13	1116.43	2572.00
V			V	V	V		V	V	V	0.39	0.22	815.57	4366.00	0.44	0.26	803.37	5203.00	0.26	0.14	1074.61	2752.00

Table 8.4: Performances of the 20 best subsets with respect to each combination rule and each evaluation metric. Observe how some subsets that do not consider all 10 input systems perform better than a global combination of all systems.

Method	Top-1 ↑	Top-10 ↑	Top-100 ↑	MR ↓	MRR ↑	MAP ↑
QMax (Baseline)	4,965	6,188	7,505	1,466	0.42	0.24
Rank Aggregation	3,956	6,132	7,591	1,011	0.37	0.21
Product	**5,460**	**6,816**	**8,269**	**878**	**0.46**	**0.27**
Improvement	+ 10 %	+ 10.2 %	+ 10.2 %	+ 40 %	+ 9.5 %	+ 12.5 %
Sum	4,611	6,401	8,136	885	0.41	0.23
Median	2,787	4,270	6,494	1,132	0.26	0.14

Table 8.5: Performance of each combination rule when considering a subset of CSI systems producing the best performance. The baseline (QMax) and Rank Aggregation here were computed with all systems. Product, sum and median were computed with the subset highlighted in Table 8.4. The improvement displays the performance gain with respect to each metric between the product and the baseline (QMax).

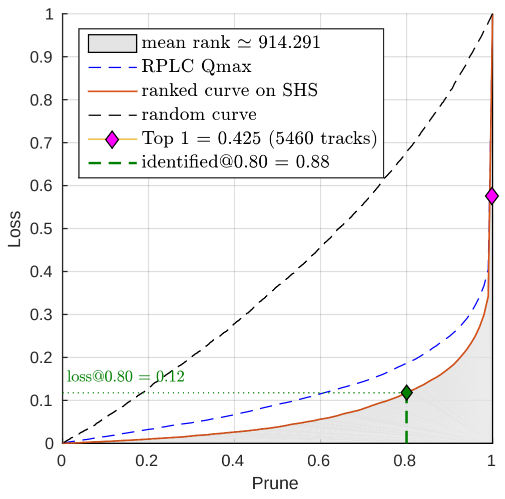


Figure 8.10: Best performance achieved with combination rules is the product, with 8 systems over 10, compared to QMax (in blue).

increase of $10\,\%$ in terms of identification rate in the Top-1, followed by an improvement of $10.2\,\%$ and $10.2\,\%$ for the Top-10 and Top-100 metrics. The MRR and MAP also increase, with an improvement of respectively $9.5\,\%$ and $12.5\,\%$. In terms of MR, this combination produces the lowest one, with an increase of $40\,\%$. Figure 8.10 shows the PL curve of the best combination. We removed classifier C_2 and C_3 as it is the combination that produces the best performance.

8.4 Final system

It is interesting to observe that systems based on the number of beats and a clustering of chroma features are almost never selected in the top-20 best subsets. On the other hand,

MR ↓		MAP ↑		MRR ↑		Identified@0.95 ↑	
method	**score**	**method**	**score**	**method**	**score**	**method**	**score**
DISCover	**878**	**DISCover**	**0.27**	**DISCover**	**0.46**	**DISCover**	**0.77**
2D-FTM	1,359	QMax	0.24	QMax	0.41	QMax	0.63
QMax	1,466	XCorr	0.09	XCorr	0.17	2D-FTM	0.55
Avg chroma	1,868	2D-FTM	0.08	2D-FTM	0.15	Avg chroma	0.44
Cluster	1,930	Timbre	0.05	Timbre	0.10	Simple	0.44
Simple	2,240	Cluster	0.026	Cluster	0.06	Cluster	0.42
XCorr	2,478	Avg chroma	0.023	Avg chroma	0.05	XCorr	0.39
Timbre	2,688	Simple	0.02	Simple	0.03	Timbre	0.34
Tempo	2,874	Tempo	0.001	Tempo	0.004	Tempo	0.23
Duration	2,900	Duration	0.001	Duration	0.003	Duration	0.19
Beats	3,075	Beats	0.001	Beats	0.003	Beats	0.19

Table 8.6: Ranking of 11 CSI systems with respect to the MR, MRR, MAP and Identification@0.95 performance. The arrows near the title of the columns indicate whether the score is ordered in ascending or descending order.

systems such as the 2D-FTM, Chroma-Mean, Timbre, QMax and SiMple are almost always used. This ordering of the performance based on subsets of initial classifiers demonstrates that using the presented rules and metrics, some features seem to be not relevant for an accurate CSI system.

Compared to the QMax system, which is considered as state-of-the-art, we manage to get an improvement of 10 % in the top-1, which corresponds to 5,460 queries identified over 12,856. This means that using our system, the proportion of queries identified at the first position equals 42,4 %. In top-10 and top-100, we manage to identify respectively 6,816 and 8,269 queries, which corresponds to 53 % and 64.3 %. This means that when we return the top-10 closest tracks to any query, the probability of identifying the query is superior to 53 %, which is a significant achievement, compared to other works.

This final composite CSI system, based on the 10 initial systems is the one we select as the achievement of this thesis. We implemented it in a working application, named *DISCover*, and available online[1]. Table 8.6 ranks our method at the first position with respect to the 10 initial systems presented in this thesis. For each metric, *DISCover* outperforms any other systems. Experimenting with the application also allows to realize that while we are still far from a fully working system, the actual application allows to find a lot of songs that are similar to a query, even if they are not cover songs. As such, we believe it could even be used for other purposes, such as music recommendation. We will discuss that in the general conclusion of the thesis.

This final combination was tested on our SHS subset containing 12,856 songs. While this

[1]http://auralie2.montefiore.ulg.ac.be

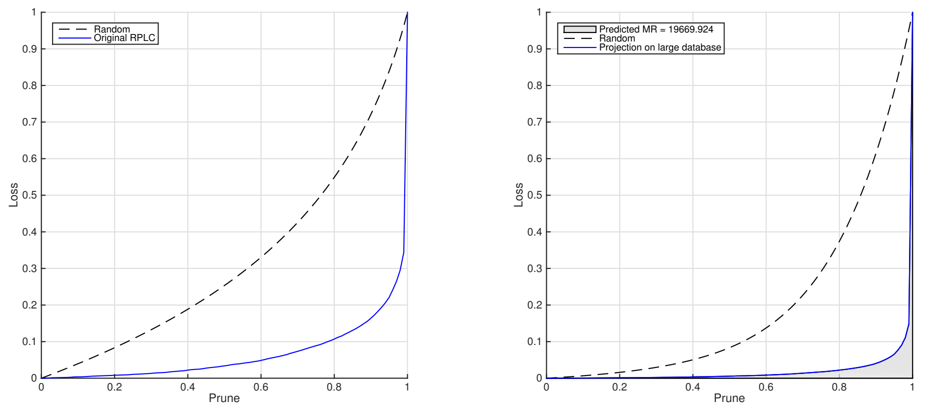


Figure 8.11: Final combination using the product rule on the SHSD (on the left), and projection (on the right) on a hypothetical database randomly generated containing 1,000,000 songs, using the RPLC-ROC bijection.

dataset is bigger than any other datasets on which CSI systems have been evaluated (to our best knowledge), it is still fat from the size of commercial databases, such as Spotify, which claim to contain 15 millions of songs. We could not run our system on the full Million Song Dataset (MSD), partly because it was absolutely impossible to run the QMax algorithm over 1M songs. However, we highlighted in Chapter 3 and 4 a way of projecting a PL curve on another dataset.

We generated a random dataset containing 1 million songs. Remember that we do not need anything else than the signature of the dataset to use our projection. The signature is a simple vector containing the sizes of the cliques in the dataset. We therefore generated a database signature containing 199,983 cliques and 1,000,001 tracks, with an average clique size of 5 and a standard deviation of 2. Figure 8.11 shows the original curve on the SHSD on the left, and the projected curve on our fake database, on the right. Keep in mind that this projection is highly hypothetical, and no database fulfilling the characteristics that we defined exist in real life. However, it is interesting to have an idea of the shape that would have our final combination on a much bigger database. From the figure, one can tell that the predicted mean rank, measured as the area under the curve, is approximately 19,669, which is still high. However, the amount of pruning that can be achieved with a minimal loss is significant. One can indeed observe that a pruning up to 90 % still produces a loss inferior to 10 %, which seems acceptable.

8.5 Conclusion

In this chapter, we studied how to make use of probability estimates to combine multiple CSI systems in order to improve the performance. Following the description of the problem, we first encountered the problem that our CSI systems do not return probabilities. It was therefore necessary to calibrate our systems, using a mapping function built from the probability density estimation of the scores for both classes, similar and dissimilar. We have reviewed several ways of calibrating systems, and ended up with our own method based on the Bayes theorem, and a probability density estimation function proposed by Botev et al. [17]. Having found a way of mapping scores to probabilities, we studied several probabilistic combining rules and experimented with them using our similarity estimators. The most efficient combination was produced using the product rule, which can be directly derived from the probability theory, in opposition to the sum rule. In our experiments, we found out that considering a subset of our 10 input methods produced an increased performance. We empirically found out the classifiers subsets that return the best performance, with respect to all metrics and combination rules, in order to avoid over-fitting. We found out that some methods decrease the performance in terms of the chosen metrics when added to the pool of systems to be combined. We eventually end up with a global CSI system, which is the combination of 8 systems over 10, combined with the probabilistic product rule.

CHAPTER 9

Conclusions

This chapter presents the final conclusions for this thesis. We remind the principal steps involved in our system, as well as key numbers. We continue with the presentation of a practical application that implements our system on a large database. It further allows the user to select audio queries and listen to the produced list of results. Finally, we close the chapter with a discussion on future perspectives in the field of cover song identification.

9.1 Summary

9.1.1 Cover Song Identification systems

In this thesis, we have studied in details the problem of cover song identification. We first explained the principles of CSI, and discussed different ways of quantifying and comparing the performance of such systems. We ended up with a new evaluation space, based on the notions of pruning and loss (see Chapters 3 and 4), in order to compare multiple systems on a common database. The evaluation space is named *Prune-Loss* space (PL) and represents the proportion of queries for which no cover has been found at all for each pruning threshold. We designed two versions of the space: *Normalized PL* (Chapter 3) and *Ranked PL* (Chapter 4). The former is recommended for researchers, to evaluate a system on a personal database, not necessarily representative of existing music, and gives the same weight to all cover groups in the search database. Thus, if some songs are more represented than others (in terms of number of versions) in the database, the performance will be normalized by the number of versions for these respective songs. The normalized space is designed so that the evaluated performance reflects how a *final system* would perform on *any provided database*.

The latter is designed for an evaluation on a specific database, taking into account the number of versions per song, without normalizing. Thus the probability that a query matches a song of the database depends on the structure of the collection. It is thus recommended for evaluating on a large database, or a commercial database used in a usable application. The ranked space is further compatible with standard metrics based on ranked lists of results and can therefore be used for existing state-of-the-art results. Using that ranked space, we further proposed an easy interpretation and visualization of standard metrics.

Having designed an evaluation space that allows to visualize the global performance of a CSI system, we implemented 10 systems based on features spanning multiple musical facets. Rather than focusing only on commonly used chroma features (See Chapter 2), we implemented systems based on timbre, tempo, duration, rhythm, structure and harmony. Among them, we implemented existing methods from the literature, and we designed new systems based on global low-dimensional features, making use of supervised machine learning algorithms to build a similarity model. In addition, we designed an algorithm based on the clustering of chroma features, making used of a bag-of-features approach. These methods are described in Chapter 5.

We evaluated all 10 systems on a common database, containing 12,856 tracks in the collection. The database is a subset of the Second Hand Song dataset, which is a standard database of cover songs, for which multiple features are pre-computed. We reported the performance in Chapter 5, in terms of pruning and loss, and in terms of standard metrics, such as the mean rank, the mean reciprocal rank, the mean average precision and the top-k identification rate.

The reported performance for each system clearly outlines that no system is usable in a commercial setup at this time. Indeed, most systems perform poorly in terms of the amount of identified queries in the top-1. The best studied system is based on the alignment of sequences of chroma features (QMax, see Section 5.1.2) and identifies 4,965 queries over 12,856 in the first position (38% of the queries). The chapter also demonstrates that when keeping 50% of the initial database, all systems show a similar performance, with 67% of the queries being identified, on average, with a standard deviation of 9%. Systems based on global features such as the tempo, the number of beats and the duration do not perform well when considered individually. However, their performance is better than random, outlining that they bring some useful information.

In addition to the global performance of each system, we also evaluated the robustness of several CSI systems in a context of audio degradation. We thus degraded audio signals by applying quadratic distortion, or adding multiple levels of ambient noise. We applied 6 audio degradations (described in Chapter 6). Results tend to show that all studied systems can be considered stable against all applied audio degradations.

9.1.2 Combining systems

Having shown that most systems produce a low performance, we investigated in Chapters 7 and 8 ways of aggregating multiple systems together, in order to build a composite system that outperforms existing systems. We highlighted two methods for combining systems.

The first one is based on rank aggregation techniques (Chapter 7), and combines ordered lists of results, based on the position of the tracks in the resulting lists. Considering 10 lists of results, ordered from the most similar to the least similar, rank aggregation applies simple mathematical operations on the ranks to compute an *aggregated rank*, that will be used in the combined resulting list. Operations corresponds to the mean of the 10 initial ranks, the median or the minimum. While such simple operations do not improve the performance, with respect to the baseline (QMax system, see Section 5.3.1.2), we show that an optimization step, named *local-Kemenization* (Section 7.4), allows to bring many tracks to the top of the list of results. This is particularly impressive for the minimum rule, which identifies 930 songs in the top-1, then 3,956 after applying the optimization step. However, the performance is still inferior to the baseline, at that specific thresholds. By first aggregating global features with a mean rule, and aggregating the resulting list with all other systems with a minimum rule with optimization, we manage to identify 4,187 tracks in the top-1, 6,433 in the top-10 and 7,790 in the top-10 (Section 7.5.2). Although the performance is close to the baseline, with respect to metrics related to thresholds close to the top-1, we show that rank aggregation significantly lowers the PL curve at lower thresholds, which means that it is able to remove a smaller portion of the database, while guaranteeing a low loss. This is confirmed by the mean rank metric, which is lowered from 1,466 for the baseline, to 983 after aggregation, thus improved by 33% (Section 7.6). Rank aggregation is thus useful for early pruning applications, rather than direct identification near the top-1.

The second combination technique we investigated in this thesis is based on probabilistic combination rules. Instead of combining full permutations of the database, we consider the score obtained for each reference track, with respect to the query, and map that score to a posterior probability, using calibration techniques (Section 8.1.1). As calibration produces interpretable posterior probabilities for our 10 CSI systems, we can make use of standard probabilistic combination rules, such as the sum, product, and median rules (Section 8.2). In our experiments, the product rule produced the best performance, with an improvement of 0.8% (5,007 tracks identified), 5.4% (6,525 tracks identified) and 9.2% (8,197) respectively in the top-1,10,100. We further improved the overall performance by selecting a subset of the 10 initial CSI systems. We found out experimentally that some systems degrade the overall performance when considered in the probabilistic combination (Section 8.3.2). Thus, we identified that CSI systems based on the beats and the clustering of chroma features are not meaningful for the final combination. The best subset of systems, with respect to the top-1 metric, drops four systems, respectively corresponding to the chroma-mean, the beats, the clustering and the cross-correlation of chroma sequences. Using the product rule, that subset identifies 5,460 tracks in the top-1, which is an improvement of 10%, compared to the baseline. In terms of top-10 and top-100, we identify 6,816 tracks (+ 10.15%) and 8,269 tracks (+10.18%). The most significant improvement is in terms of mean rank, with an improvement of 40%, thus going from 1,466 to 885. In terms of MRR and MAP, the scores are respectively 0.46 (+9.5%) and 0.27 (+12.5%).

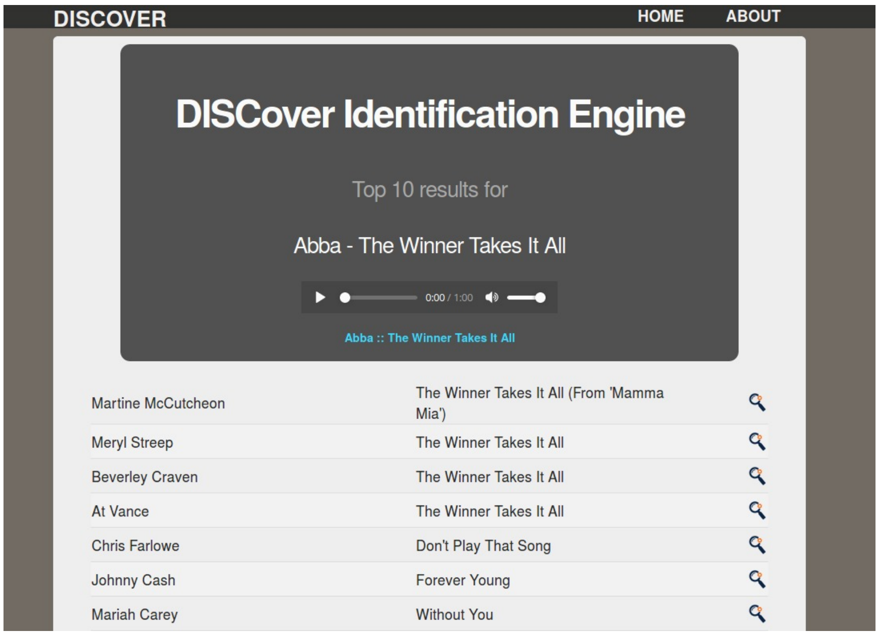
The last combination, using a probabilistic product of a subset of our 10 initial system corresponds to our final performance. We implemented it in a practical application, named *DISCover*, which we present in the next section.

Note that this thesis led to the implementation of multiple features and matching algorithms. As it is sometimes difficult to get access to the original implementation, and because we needed performance, we implemented every single method by ourselves, thus leading to a complete C++ evaluation framework. The framework contains 16,239 lines of C++, 912 lines of SH scripting code, 805 lines of python and 408 lines of C language, for a total of 18,364 lines for the final version.

9.2 Practical demonstrator

In order to practically experiment how our final composite system performs, we implemented a practical demonstrator giving us the possibility to actually listen to the list of returned tracks, with respect to a selected query. The application allows a user to select an artist, and proposes a list of available songs for that artist. It then returns the list of tracks that are considered the most similar to the chosen query. The user can then listen to each resulting track. Figure 9.1 shows a screen shot of the demonstrator, implemented as a web application. The database used in the application corresponds to the test set we used throughout this thesis, containing 12,856 songs.

Experimenting with the application, one can realize that in most cases, even if cover versions are not always found, the top-10 tracks are musically similar to the query. In particular, we were able to identify songs that are so similar to the query that they actually could be cover versions. Consider for example the song "*Sometimes*", played by Britney Spears. The first



DISCover Identification Engine

Top 10 results for

Abba - The Winner Takes It All

0:00 / 1:00

Abba :: The Winner Takes It All

Martine McCutcheon	The Winner Takes It All (From 'Mamma Mia')
Meryl Streep	The Winner Takes It All
Beverley Craven	The Winner Takes It All
At Vance	The Winner Takes It All
Chris Farlowe	Don't Play That Song
Johnny Cash	Forever Young
Mariah Carey	Without You

Figure 9.1: Screenshot of the DISCover prototype implementing the final combination of this thesis. The example shows results for the query song "The Winner Takes it All" by ABBA. One can observe that the five first returned results are cover versions of the query. The reference database contains 12,856 songs.

returned match is "*No one else comes close*" by the Backstreet Boys, which is not a cover. It is however surprisingly close to the query, melodically speaking.

Using the application, we were able to identify cover versions of queries sang in different languages. For instance, we found a cover version for the French song "*Le Gorille*", from Georges Brassens, in Italian language. The identified version is sang by Piccola Bottega Baltazar, and is named "*Il Gorilla*". Note that this particular version is musically similar to the query, and is therefore identified at the first position. The instrumentation is however quite different, which shows that the system is quite robust against such changes. Another particularly interesting example is the song "*Missing you*", played by the band Brooks & Dunn. Both first returned tracks are covers: the first one is sang in English, while the second one is an Italian version, "*Mi Manchi Tu*", played by Paola Turci.

From a musical point of view, we noticed that the rhythm and tempo are often very similar to the query, even when no cover version is found. Consider for example the song "*Monkey Man*", by Amy Winehouse. The first song that is identified is "*Brown Eyed Girl*", by Steel Pulse, and although it is a different song, the rhythm is similar. The second identified song is "*Singin' the Blues*", by Eric Clapton, which is definitely a different song, yet showing similarities in terms of rhythm.

The particular case of jazz and blues music

When listening to jazz music queries, we noticed a different behavior than for popular western music. For jazz music, often no cover version is found, with respect to the query. However, the list of results in the top-10, top-20, etc. highlights songs that are all musically similar to the query. As an example, the song "*I Wonder*" by Steve Tyrell taken as a query produces a list of songs that are all musically similar (to our very subjective opinion) to the query, while being sufficiently different so that we do not have the feeling of listening to the same song. It thus produces an interesting playlist of jazz songs, which is pleasant to hear. The system behaves similarly for blues music. It often returns a list a blues tracks, all close to the query while being different, also producing an interesting playlist. As an example, one can consider the song "*Rock me Baby*" by Luther Allison, which produces a nice playlist of blues songs.

Note that this analysis is purely subjective, and we do not provide in this thesis any theoretical explanation for such a behavior. It is however known that blues and jazz music often highlight similar patterns, often standard in these musical genres, which might explain such a behavior. For the interested reader, Foster et al. [41] provide an evaluation of cover song identification systems on a jazz music dataset, confirming that the behavior on that musical genre is different than more popular music.

Alternative applications

Our cover song identification system is able to identify songs that are musically similar to the query, even if they are not cover versions. It therefore opens perspective for different use cases, which we highlight below. Other applications are possible, and we only propose three of them to the reader.

Playlist generation

As pointed out in the previous section about jazz and blues music, we realized that our system is able to produce a coherent list of songs musically similar to the query. While it is clearly a *raw* list of results, with no specific ordering of the tracks to produce a relevant playlist, we believe the system can be tuned to be used as a playlist generator.

Composition helper

Most musicians and bands are inspired by previously recorded music. Nowadays, it is common to hear new songs on the radio that sound very similar to some other songs. Indeed, most standard popular and rock music patterns have been used numerous times, which explain such similarities, and thus makes it more difficult for musicians to be original and creative. One could use a cover song identification system, or a general musical similarity algorithm to compare a new composition to a reference collection, and find out that existing songs sound similar. With an iterating process, the composer could then end up with a totally new and original song.

Plagiarism detection

Finally, a straightforward application is plagiarism detection. As a CSI system returns musically similar songs, it might help in plagiarism detection, by automatically identifying potential copies of an original song.

9.3 Discussion and perspectives

Let us now close this thesis with a discussion on possible perspectives. Let us also remind that our research work was done in a specific context, using a very particular database with no audio data available. We thus had to restrict ourselves to the set of features provided with the database. The features themselves were unfortunately not documented, so we could not analyze the way they were computed. Considering that, one straightforward way of improving the performance of a CSI system would simply be to have access to audio data. This would allow researchers to design powerful features and learn similarity on understandable data, which would most likely lead to an increase of the performance.

Unfortunately, this is still a huge issue in the MIR community at this time, and music companies are not willing to freely share audio data for the research community. However, the Million Song Dataset, and its subset, the Second Hand Song dataset, are valuable resources for the community, and this thesis shows that it can help leading to new contributions. Below, we highlight some perspectives that could be considered for improving cover song identification.

Refining the definition of a cover song

One big difficulty in the field of CSI is that there is no clear definition of what a cover song is, with respect to the original version. Using our demonstrator, *DISCover*, we noticed that some supposed cover versions of original tracks are sometimes significantly different from the original track. Should we then consider that such a song is still a cover and should be identified as such ? What if even a human being is not able to identify as a cover song ? Such questions are useful, and answering them would definitely help to re-define the problem of cover song identification. Especially for evaluation, if such songs are removed from the database, and all cover versions follow the *same tonal progression* constraint, maybe existing systems would perform much better. Researchers in musicology might be able to refine that definition.

Designing new features

Assuming that a large-scale audio database will become available in the future, new features would become necessary to improve cover song recognition. So far, chroma features have been the most widely used features for cover song identification, mainly because they encode meaningful harmonic information, while showing a high degree of robustness against changes in instrumentation, voicing, and timbre in general. However, we showed in this thesis that other features bring also meaningful information. For instance, features derived from timbre, as the ones proposed by Tralie et al. [105], show an interesting performance and are always selected in our final subset of systems to be considered for combination. We also showed that features based on the musical structure produce results. So it is possible to improve the performance by considering totally different features, not even related to the harmonic content. Chroma features might be limited for cover song identification, at some point, and new features should be considered. One possibility would be to consider perceptual features, such as the Cochlear Pitch Class profile [26] or the features provided by Van Balen et al. [3]. Including features related to the melody to a combined system would also probably improve overall performance.

Designing new similarity estimation algorithms

Following the previous section on new features, designing appropriate matching algorithms is also mandatory. We have seen in this thesis that using multiple distance measures and scores, even when using similar base features, helps to improve the performance. Therefore, one can imagine designing multiple matching algorithms for the same base features in order to increase the recognition rate.

These perspectives are simple ideas for improving state-of-the-art performance. We believe that a lot of relevant work was realized in that area, and evaluating existing systems with good features, extracted straight from audio files, would lead to a huge improvement. Special care should also be considered for the runtime requested to compare a query to a large database. In this thesis, we are making use of time-consuming alignment algorithms, such as QMax, which are not usable on very large databases, containing millions of songs. For a commercial system, filter-and-refine methods, or index-based methods should probably be preferred. However, before dealing with such optimization, understanding deeply the problem, and why it has not been solved yet, is mandatory. While this thesis brings a contribution, we believe a lot of interesting work can still be done in that field of research, and we encourage researchers to help solving the fascinating problem of cover song identification.

Nomenclature

2D-FTM	2 Dimensional Fourier Transform Magnitude
ADT	Audio Degradation Toolbox
CBRS	Content Based Retrieval System
CPCP	Cochlear Pitch Class Profile
CRP	Cross Recurrence Plot
CSI	Cover Song Identification
DTW	Dynamic Time Warping
FFT	Fast Fourier Transofrm
FNR	False Negative Rate
FPR	False Positive Rate
HMM	Hidden Markov Model
HPCP	Harmonic Pitch Class Profile
IRS	Information Retrieval System
ISMIR	International Symposium on Music Information Retrieval
LSH	Locality-sensitive hashing

LS	Learning Set
MFCC	Mel-frequency cepstrum coefficients
MIREX	Music Information Retrieval Evaluation eXchange
MIR	Music Information Retrieval
NPLC	Normalized Prune Loss Curve
OTI	Optimal Transposition Index
PCA	Principal Component Analysis
PCP	Pitch Class Profile
PDF	Probability Density Function
PLC	Prune Loss Curve
PL	Prune Loss
PR	Precision - Recall
ROC	Receiver Operating Characteristic
RPLC	Ranked Prune Loss Curve
SHSD	Second Hand Song Dataset
SIFT	Scale Invariant Feature Transform
SMR	Single Match Retrieval
STFT	Short Time discrete Fourier Transform
TBRS	Text Based Retrieval System
TNR	True Negative Rate
TP, TN, FP, FN	True positive, True negative, False positive, False negative
TPR	True Positive Rate
TS	Test Set

Bibliography

[1] T. Ahonen. Combining Chroma Features for Cover Version Identification. In *International Conference on Music Information Retrieval (ISMIR)*, pages 165–170, Utrecht, Netherlands, 2010.

[2] T. Ahonen. Compression-Based Clustering of Chromagram Data : New Method and Representations. In *International Symposium on Computer Music Multidisciplinary Research*, pages 474–481, Queen Mary University, London, UK, 2012.

[3] J. Balen, D. Bountouridis, F. Wiering, and R. Veltkamp. Cognition-inspired Descriptors for Scalable Cover Song Retrieval. In *International Conference on Music Information Retrieval (ISMIR)*, pages 379–384, Taipei, Taiwan, 2014.

[4] M. Barreno, A. Cardenas, and J. Tygar. Optimal ROC Curve for a Combination of Classifiers. In *Proceedings of the 2008 Conference on Neural Information Processing Systems*, pages 56–64, Vancouver, BC, Canada, 2008.

[5] J. Bello. Audio-Based Cover Song Retrieval Using Approximate Chord Sequences: Testing Shifts, Gaps, Swaps and Beats. In *International Conference on Music Information Retrieval (ISMIR)*, pages 239–244, Vienna, Austria, 2007.

[6] J. Bello. Measuring Structural Similarity in Music. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2013–2025, 2011.

[7] P. Bennett. Assessing the Calibration of Naive Bayes' Posterior Estimates. Technical report, Carnegie Mellon University, Pittsburgh, PA, 2000.

[8] P. Bennett. Using Asymmetric Distributions to Improve Classifer Probabilities: A Comparison of New and Standard Parametric Methods. Technical report, Carnegie Mellon, School of Computer Science, 2002.

[9] P. Bennett. Using Asymmetric Distributions to Improve Text Classifier Probability Estimates. In *International Conference on Research and Development in Informaion Retrieval*, pages 111–118, Toronto, Canada, 2003.

[10] A. Berenzweig, B. Logan, D. Ellis, and B. Whitman. A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures. *Computer Music Journal*, 28(2):63–76, 2004.

[11] T. Bertin-Mahieux. *Large-Scale Pattern Discovery in Music*. PhD thesis, Columbia University, 2013.

[12] T. Bertin-Mahieux and D. Ellis. Large-Scale Cover Song Recognition Using Hashed Chroma Landmarks. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 117–120, New Paltz, NY, USA, 2011.

[13] T. Bertin-Mahieux and D. Ellis. Large-Scale Cover Song Recognition Using the 2D Fourier Transform Magnitude. In *International Conference on Music Information Retrieval (ISMIR)*, pages 241–246, Porto, Portugal, 2012.

[14] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The Million Song Dataset Challenge. In *International Conference Companion on World Wide Web*, pages 909–916, Lyon, France, 2012.

[15] T. Bertin-Mahieux, R. Weiss, and D. Ellis. Clustering Beat-Chroma Patterns in a Large Music Database. In *International Conference on Music Information Retrieval (ISMIR)*, pages 111–116, Utrecht, Netherlands, 2010.

[16] F. Bimbot, E. Deruty, G. Sargent, and E. Vincent. Methodology and Resources for the Structural Segmentation of Music Pieces into Autonomous and Comparable Blocks. In *International Conference on Music Information Retrieval (ISMIR)*, pages 287–292, Miami, USA, 2011.

[17] Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *Annals of Statistics*, 38(5):2916–2957, 2010.

[18] A. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien. Efficient Query Evaluation Using a Two-Level Retrieval Process. In *International Conference on Information and Knowledge Management (CIKM)*, pages 426–434, New Orleans, LA, USA, 2003.

[19] J. Burgoyne, J. Wild, and I. Fujinaga. An Expert Ground-Truth Set for Audio Chord Recognition and Music Analysis. In *International Conference on Music Information Retrieval (ISMIR)*, pages 633–638, Miami, USA, 2011.

[20] K. Cai, D. Yang, and X. Chen. Two-layer Large-scale Cover Song Identification System Based on Music Structure Segmentation. In IEEE, editor, *International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, London, UK, 2016.

[21] M. Casey, C. Rhodes, and M. Slaney. Analysis of Minimum Distances in High-Dimensional Musical Spaces. *IEEE Transactions on Audio, Speech and Language Processing*, 16(5):1015–1028, 2008.

[22] M. Casey and M. Slaney. Song Intersection by Approximate Nearest Neighbor Search. In *International Conference on Music Information Retrieval (ISMIR)*, pages 144–149, Victoria, Canada, 2006.

[23] M. Casey and M. Slaney. Fast Recognition of Remixed Music Audio. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1425–1428, Honolulu, USA, 2007. IEEE.

[24] M. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-Based Music Information Retrieval: Current Directions and Future Challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[25] S. Chapaneri. Spoken Digits Recognition using Weighted MFCC and Improved Features for Dynamic Time Warping. *International Journal of Computer Applications*, 40(3):6–12, 2012.

[26] N. Chen, J. Downie, H. Xiao, and Y. Zhu. Cochlear Pitch Class Profile for Cover Song Identification. *Applied Acoustics*, 99:92–96, 2015.

[27] N. Chen and H. Xiao. Similarity Fusion Scheme for Cover Song Identification. *Electronics Letters*, 52(13):1173–1175, 2016.

[28] J. Davis and M. Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *International Conference on Machine Learning (ICML)*, pages 233–240, Pittsburgh, Pennsylvania, USA, 2006.

[29] J. Downie. Music Information Retrieval. *Annual Review of Information Science and Technology*, 37:295–340, 2003.

[30] J. Downie. The Music Information Retrieval Evaluation Exchange (2005 2007): A Window into Music Information Retrieval Research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[31] J. Downie, A. Ehmann, M. Bay, and M. Jones. The Music Information Retrieval Evaluation Exchange: Some Observations and Insights. *Studies in Computational Intelligence*, 274:93–115, 2010.

[32] R. Duin and D. Tax. Experiments with Classifier Combining Rules. In *International Workshop on Multiple Classifier Systems (MCS)*, pages 16–29, Cagliari, Italy, 2000.

[33] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank Aggregation Methods for the Web. In *International Conference on World Wide Web (WWW)*, pages 613–622, Hong Kong, China, 2001.

[34] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank Aggregation Revisited. *Systems Research*, 13(2):86–93, 2001.

[35] D. Ellis and C. Cotton. The 2007 LabRosa Cover Song Detection System. In *MIREX extended abstract*, 2007.

[36] D. Ellis, C. Cotton, and M. Mandel. Cross-Correlation of Beat-Synchronous Representations for Music Similarity. In *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 57–60, Las Vegas, USA, 2008.

[37] D. Ellis and G. Poliner. Identifying Cover Songs with Chroma Features and Dynamic Beat Tracking. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1429–1432, Honolulu, USA, 2007.

[38] T. Fawcett and A. Niculescu-Mizil. PAV and the ROC Convex Hull. *Machine Learning*, 68(1):97–106, 2007.

[39] J. Foo and R. Sinha. Pruning SIFT for scalable near-duplicate image matching. In *Conferences in Research and Practice in Information Technology Series*, volume 63, pages 63–71, Ballarat, Victoria, Australia, 2007.

[40] P. Foster. *Information-Theoretic Measures of Predictability for Music Content Analysis*. PhD thesis, Queen Mary University of London, 2014.

[41] P. Foster. *Information-Theoretic Measures of Predictability for Music Content Analysis*. PhD thesis, Queen Mary University of London, 2014.

[42] P. Foster, S. Dixon, and A. Klapuri. Identifying Cover Songs Using Information-Theoretic Measures of Similarity. *IEEE Transactions on Audio, Speech and Language Processing*, 23(6):993–1005, 2015.

[43] Z. Fu, G. Lu, K. Ting, and D. Zhang. A Survey of Audio-Based Music Classification and Annotation. *IEEE Transactions on Multimedia*, 13(2):303–319, 2011.

[44] Z. Fu, G. Lu, K. Ting, and D. Zhang. Music Classification via the Bag-of-Features Approach. *Pattern Recognition Letters*, 32(14):1768–1777, 2011.

[45] T Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *International Computer Music Conference (ICMC)*, volume 9, pages 464–467, Beijing, China, 1999.

[46] J. Gao and P. Tan. Converting Output Scores From Outlier Detection Algorithms into Probability Estimates. In *International Conference on Data Mining (ICDM)*, pages 212–221, Hong Kong, China, 2006.

[47] M. Gebel and C. Weihs. Calibrating Classifier Scores into Probabilities. *Advances in Data Analysis*, pages 141–148, 2007.

[48] P. Geurts, D. Ernst, and L. Wehenkel. Extremely Randomized Trees. *Machine Learning*, 63(1):3–42, 2006.

[49] E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra, 2006.

[50] E. Gómez. Tonal Description of Polyphonic Audio for Music Content Processing. *INFORMS Journal on Computing*, 18(3):294–304, 2006.

[51] E. Gómez and P. Herrera. The Song Remains The Same: Identifying Versions of The Same Piece Using Tonal Descriptors. In *International Conference on Music Information Retrieval (ISMIR)*, pages 180–185, Victoria, Canada, 2006.

[52] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC Music Database: Popular, Classical, and Jazz Music Databases. In *International Conference on Music Information Retrieval (ISMIR)*, pages 287–288, Paris, France, 2002.

[53] P. Grosche and M. Müller. Toward Characteristic Audio Shingles for Efficient Cross-Version Music Retrieval. In *International conference on acoustics, speech and signal processing (ICASSP)*, pages 473–476, Kyoto, Japan, 2012. IEEE.

[54] P. Grosche, M. Müller, and J. Serrà. Audio Content-Based Music Retrieval. *Multimodal Music Processing*, 3:157–174, 2012.

[55] S. Haker, W. Wells, S. Warfield, I. Talos, J. Bhagwat, D. Goldberg-Zimring, A. Mian, L. Ohno-Machado, and K. Zou. Combining Classifiers Using their Receiver Operating Characteristics and Maximum Likelihood Estimation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 8, pages 506–514, Palm Springs, CA, USA, 2005.

[56] E. Humphrey, O. Nieto, and J. Bello. Data Driven and Discriminative Projections for Large-scale Cover Song Identification. In *International Conference on Music Information Retrieval (ISMIR)*, pages 4–9, Curitiba, Brazil, 2013.

[57] J. Jensen, M. Christensen, and S. Jensen. A Chroma-Based Tempo-Insensitive Distance Measure for Cover Song Identification using the 2D Autocorrelation Function. In *Proceedings of the Music Information Retrieval Evaluation Exchange (MIREX)*, 2008.

[58] M. Khadkevich and M. Omologo. Large-Scale Cover Song Identification Using Chord Profiles. In *International Conference on Music Information Retrieval (ISMIR)*, pages 233–238, Curitiba, Brazil, 2013.

[59] W. Khreich, E. Granger, A. Miri, and R. Sabourin. Iterative Boolean Combination of Classifiers in the ROC Space: An Application to Anomaly Detection with HMMs. *Pattern Recognition*, 43(8):2732–2752, 2010.

[60] W. Khreich, E. Granger, A. Miri, and R. Sabourin. Adaptive ROC-Based Ensembles of HMMs Applied to Anomaly Detection. *Pattern Recognition*, 45(1):208–230, 2012.

[61] S. Kim and S. Narayanan. Dynamic Chroma Feature Vectors with Applications to Cover Song Identification. In *Workshop on Multimedia Signal Processing (MMSP)*, pages 984–987, Los Angeles, USA, 2008.

[62] S. Kim, E. Unal, and S. Narayanan. Music Fingerprint Extraction for Classical Music Cover Song Identification. In *International Conference on Multimedia and Expo (ICME)*, pages 1261–1264, Los Angeles, USA, 2008.

[63] S. Kim, E. Unal, and S. Narayanan. Music Fingerprint Extraction for Classical Music Cover Song Identification. In *International Conference on Multimedia and Expo (ICME)*, pages 1261–1264, Los Angeles, USA, 2008.

[64] J. Kittler, M. Hater, R. Duin, and J. Matas. On Combining Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[65] F. Kurth and M. Müller. Efficient Index-Based Audio Matching. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):382–395, 2008.

[66] A. Langville and C. Meyer. *Who's #1? The Science of Rating and Ranking.* Princeton University Press, 2012.

[67] E. Law and L. Von Ahn. Input-Agreement: A New Mechanism for Collecting Data Using Human Computation Games. In *International Conference on Human Factors in Computing Systems*, pages 1197–1206, Boston, USA, 2009. ACM.

[68] K. Lee. Identifying Cover Songs from Audio Using Harmonic Representation. In *Proceedings of the Music Information Retrieval Evaluation Exchange*, Victoria, Canada, 2006.

[69] Y. Lu and J. Cabrera. Large Scale Similar Song Retrieval using Beat-aligned Chroma Patch Codebook with Location Verification. In *International Conference on Signal Processing and Multimedia Applications (SIGMAP)*, pages 208–214, Athens, Greece, 2010.

[70] C. Manning, P. Raghavan, and H. Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, 2008.

[71] M. Marolt. A Mid-Level Representation for Melody-Based Retrieval in Audio Collections. *IEEE Transactions on Multimedia*, 10(8):1617–1625, 2008.

[72] B. Martin, D. Brown, P. Hanna, and P. Ferraro. BLAST for Audio Sequences Alignment: A Fast Scalable Cover Identification Tool. In *International Conference on Music Information Retrieval (ISMIR)*, pages 529–534, Porto, Portugal, 2012.

[73] M. Mauch and S. Ewert. The Audio Degradation Toolbox and Its Application To Robustness Evaluation. In *International Conference on Music Information Retrieval (ISMIR)*, pages 2–7, Curitiba, Brazil, 2013.

[74] M. Mauch, R. Maccallum, M. Levy, and A. Leroi. The Evolution of Popular Music: USA 1960-2010. *Royal Society Open Science*, 2(5), 2015.

[75] B. Mcfee, E. Humphrey, and J. Urbano. A Plan for Sustainable MIR Evaluation. In *International Conference on Music Information Retrieval (ISMIR)*, pages 285–291, New York City, USA, 2016.

[76] M. Mueller, F. Kurth, and M. Clausen. Audio Matching via Chroma-Based Statistical Features. In *International Conference on Music Information Retrieval (ISMIR)*, pages 288–295, London, UK, 2005.

[77] A. Niculescu-Mizil and R. Caruana. Obtaining Calibrated Probabilities from Boosting. In *Uai*, pages 413–420, Edinburgh, Scotland, 2005.

[78] A. Niculescu-Mizil and R. Caruana. Predicting Good Probabilities with Supervised Learning. In *International Conference on Machine Learning (ICML)*, pages 625–632, Bonn, Germany, 2005.

[79] N. Orio, D. Rizo, R. Miotto, N. Montecchio, M. Schedl, and O. Lartillot. MUSICLEF: A Benchmark Activity in Multimodal Music Information Retrieval. In *International Conference on Music Information Retrieval (ISMIR)*, pages 603–608, Miami, USA, 2011.

[80] J. Osmalskyj, S. Pierard, M. Van Droogenbroeck, and J. Embrechts. Efficient Database Pruning for Large-Scale Cover Song Recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 714–718, Vancouver, BC, 2013.

[81] J. Osmalskyj, M. Van Droogenbroeck, and J. Embrechts. Performances of Low-Level Audio Classifiers for Large-Scale Music Similarity. In *International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 91–94, Dubrovnik, Croatia, 2014.

[82] J. Osmalskyj, M. Van Droogenbroeck, and J. Embrechts. Enhancing Cover Song Identification with Hierarchical Rank Aggregation. In *International Conference on Music Information Retrieval (ISMIR)*, pages 136–142, New York, NY, USA, 2016.

[83] G. Peeters, J. Urbano, and G. Jones. Notes from the ISMIR 2012 Late-Breaking Session on Evaluation in Music Information Retrieval. In *International Conference on Music Information Retrieval (ISMIR)*, Porto, Portugal, 2012.

[84] J Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[85] J. Salamon and E. Gómez. A Chroma-Based Salience Function for Melody and Bass Line Estimation from Music Audio Signals. In *Sound and Music Computing*, pages 331 – 336, Porto, Portugal, 2009.

[86] D. Sculley. Rank Aggregation for Similar Items. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 587–592, Bethesda, Maryland, USA, 2007.

[87] J. Serrà. *Identification of Versions of the Same Musical Composition by Processing Audio Descriptions*. PhD thesis, Universitat Pompeu Fabra, 2011.

[88] J. Serrà and E. Gómez. Transposing Chroma Representations to a Common Key. In *IEEE Conference on The Use of Symbols to Represent Music and Multimedia Objects*, pages 45–48, Manno, Switzerland, 2008.

[89] J. Serrà, E. Gomez, P. Herrera, and X. Serra. Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification. *IEEE Transactions on Audio, Speech and Language Processing*, 16(6):1138–1151, 2008.

[90] J. Serrà, K. Holger, X. Serra, and R. Andrzejak. Predictability of Music Descriptor Time Series and its Application to Cover Song Detection. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):514–525, 2012.

[91] J Serra, M Müller, Peter Grosche, and Josep Arcos. Unsupervised detection of music boundaries by time series structure features. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1613–1619, 2012.

[92] J. Serrà, X. Serra, and R. Andrzejak. Cross Recurrence Quantification for Cover Song Identification. *New Journal of Physics*, 11(9), 2009.

[93] J. Serrà, M. Zanin, and R. Andrzejak. Cover song retrieval by cross recurrence quantification and unsupervised set detection. In *MIREX extended abstract*, 2009.

[94] D. Silva and G. Batista. Music Shapelets for Fast Cover Song Recognition. In *International Conference on Music Information Retrieval (ISMIR)*, pages 441–447, Malaga, Spain, 2015.

[95] D. Silva, C. Yeh, G. Batista, and E. Keogh. SiMPle: Assessing Music Similarity using Subsequences Joins. In *International Conference on Music Information Retrieval (ISMIR)*, pages 23–30, New York, USA, 2016.

[96] A. Singhal. Modern Information Retrieval: A Brief Overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):1–9, 2001.

[97] M. Slaney and M. Casey. Locality-Sensitive Hashing for Finding Nearest Neighbors. *IEEE Signal Processing Magazine*, 25(2):128–131, 2008.

[98] J. Smith, J. Burgoyne, I. Fujinaga, D. De Roure, and J. Downie. Design and Creation of a Large-Scale Database of Structural Annotations. In *International Conference on Music Information Retrieval (ISMIR)*, pages 555–560, Miami, USA, 2011.

[99] M. Squire, H. Muller, and W. Muller. Improving Response Time by Search Pruning in a Content-Based Image Retrieval System, Using Inverted File Techniques. In *IEEE*

Workshop on Content-Based Access of Image and Video Libraries (CBAIVL), pages 45–49, Hilton Head Island, SC, USA, 1999.

[100] R. Stewart and M. Sandler. Database of Omnidirectional and B-format Room Impulse Responses. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 165–168, Dallas, USA, 2010.

[101] B. Sumengen and B. Manjunath. Category Pruning in Image Databases Using Segmentation and Distance Maps. In *Signal Processing Conference (EUSIPCO)*, Antalya, Turkey, 2005.

[102] D. Tax, R. Duin, and M. Breukelen. Comparison Between Product and Mean Classifier Combination Rules. In *Proceedings of the Workshop on Statistical Pattern Recognition*, pages 165 –170, Prague, Czech, 1997.

[103] D. Tax, M. Van Breukelen, R. Duin, and J. Kittler. Combining Multiple Classifiers by Averaging or by Multiplying? *Pattern Recognition*, 33(9):1475–1485, 2000.

[104] D. Tingle, Y. Kim, and D. Turnbull. Exploring Automatic Music Annotation with "Acoustically-Objective" Tags. In *International Conference on Music Information Retrieval (ISMIR)*, pages 55–62, Philadelphia, USA, 2010.

[105] C. Tralie and P. Bendich. Cover Song Identification with Timbral Shape Sequences. In *International Conference on Music Information Retrieval (ISMIR)*, pages 38–44, Malaga, Spain, 2015.

[106] M. Truchon. An Extension of the Condorcet Criterion and Kemeny Orders. Technical report, Centre de Recherche en Economie et Finance Appliquees (CREFA), 1998.

[107] W. Tsai and H. Yu. A Query-by-Example Technique for Retrieving Cover Versions of Popular Songs with Similar Melodies. In *International Conference on Music Information Retrieval (ISMIR)*, pages 183–190, London, UK, 2005.

[108] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards Musical Query-by-Semantic-Description using the CAL500 Data Set. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 439–446, New York, NY, USA, 2007.

[109] G. Tzanetakis and P. Cook. Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[110] T. Walters, D. Ross, and R. Lyon. The Intervalgram: An Audio Feature for Large-Scale Cover-Song Recognition. In *International Symposium on Computer Music Modeling and Retrieval (CMMR*, pages 197–213, Marseille, France, 2013.

[111] B. Wang, A. Mezlini, F. Demir, M. Fiume, Z. Tu, M. Brudno, B. Haibe-Kains, and Anna Goldenberg. Similarity Network Fusion for Aggregating Data Types on a Genomic Scale. *Nature Methods*, 11(3):333–337, 2014.

[112] L. Ye and E. Keogh. Time Series Shapelets: A New Primitive for Data Mining. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 947–956, Paris, France, 2009.

[113] T. Yu. ROCS: Receiver Operating Characteristic Surface for Class-Skewed High-Throughput Data. *PLoS ONE*, 6(4), 2011.

[114] B. Zadrozny and C. Elkan. Learning and Making Decisions when Costs and Probabilities are Both Unknown. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 204–213, New York, NY, USA, 2001.

[115] B. Zadrozny and C. Elkan. Transforming Classifier Scores into Accurate Multiclass Probability Estimates. In *International Conference on Knowledge Discovery and Data Mmining (SIGKDD)*, pages 694–699, Edmonton, Canada, 2002.