Influence of Summarization on Music Classification Tasks

Francisco Afonso Raposo

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Doctor David Manuel Martins de Matos
Doctor Ricardo Daniel Santos Faro Marques Ribeiro

Examination Committee

Chairperson: Doctor Pedro Manuel Moreira Vaz Antunes de Sousa
Supervisor: Doctor David Manuel Martins de Matos
Member of the Committee: Doctor Bruno Emanuel da Graça Martins

October 2014

Acknowledgements

I would like to thank my advisor David Martins de Matos and co-advisor Ricardo Ribeiro for all the interesting and motivating discussions throughout this journey and for all their valuable advice and help. This thesis would not be possible without their input.

I would like to thank L2F and INESC-ID for providing me with the means to conduct my research.

I would like to thank all my colleagues at L2F. A special thanks to Jaime Ferreira for his practical input on implementation aspects and to António Lopes for his help on building the multiclass dataset.

I would like to thank all my friends and family for their support and belief in me.

Lisboa, October 29, 2014
Francisco Afonso Raposo

For my mother

Resumo

Classificação de música consiste em predizer classes de músicas (por exemplo artista, género) e é uma das tarefas principais na comunidade de Recuperação de Informação Musical. Vários sistemas para classificação de música foram implementados. No entanto, os classificadores de música não são infalíveis, às vezes classificando músicas erradamente. Os classificadores processam apenas um pequeno segmento da música (para poupar tempo) e são avaliados em datasets para obter um valor de exatidão determinando a performance desse classificador.

Sumarização de música é a tarefa de sumarizar música. Noutras palavras, consiste em selecionar as partes mais importantes de uma música e colá-las produzindo um sumário dessa música. Vários algoritmos de sumarização de música foram desenvolvidos no passado, com o objectivo de produzir um sumário final para ser disfrutado por ouvintes humanos. Também existe muito trabalho publicado acerca de algoritmos de sumarização genérica, os quais têm sido aplicados com sucesso em sumarização de texto e fala.

Esta tese avalia o impacto da sumarização na classificação de música. Dado que o classificador está limitado a processar apenas um pequeno segmento de cada música, vale a pena considerar que processar um segmento sumário em vez de um segmento aleatório melhora a exatidão do classificador. Resultados da avaliação de Average Similarity, LexRank, Latent Semantic Analysis (LSA) e Maximal Marginal Relevance (MMR) em 2 classificadores diferentes mostram que a sumarização melhora a exatidão de classificação.

Abstract

Music classification consists of predicting classes of songs (e.g. artist, genre) and is one of the main tasks in the Music Information Retrieval (MIR) community. Several systems for music classification have been implemented. However, music classifiers are not infallible, sometimes wrongly classifying songs. Classifiers process only a small segment of the song (to save time) and are evaluated on datasets to get an accuracy score determining that classifier's performance.

Music summarization is the task of summarizing music. In other words, it consists of selecting the most important parts of a song and glue them together producing a summary clip of the song. Several music summarization algorithms have been developed in the past, with the goal of producing a final summary to be enjoyed by human listeners. There is also a lot of work published regarding generic summarization algorithms, which have been successfully applied in text and speech summarization.

This thesis evaluates summarization's impact on music classification. Since the classifier is limited to processing just a small segment of each song, it is worth considering that processing a summary segment instead of a random segment improves the classifier's accuracy. Results of evaluating Average Similarity, LexRank, LSA and MMR on 2 different classifiers show that summarization improves classification accuracy.

Palavras Chave
Keywords

Palavras Chave

Sumarização de Música

Classificação de Música

Características de Sinais de Áudio

Recuperação de Informação Musical

Relevância

Redundância

Keywords

Music Summarization

Music Classification

Audio Signal Features

Music Information Retrieval

Relevance

Redundancy

Índice

List of Tables

1 Introduction

MIR is a research area that addresses automatic processing and extraction of musical information. Some common MIR tasks are genre and mood classification, cover song identification, music similarity and retrieval, onset detection, beat tracking, tempo estimation, query by singing/humming, structural segmentation, to name a few. These, and many other tasks, are annually addressed by the MIR community in conferences such as that from the International Society for Music Information Retrieval (ISMIR), where algorithms are published and evaluated. Music Information Retrieval Evaluation eXchange (MIREX) was created as part of the 6th ISMIR conference to provide an evaluation framework for many different MIR tasks. Music classification has been one of such tasks in many instances of MIREX. More specifically, tasks such as classical composer identification, US Pop music genre classification, and mood classification whose goal is to output a label/class according to the song's composer, genre, and mood, respectively.

Classification systems work by first training a model on a set of examples and then using this model to predict which class (from the pool of classes learned in training) a new object (song) belongs to. However, the classifier does not usually process the whole song, taking only a small segment (e.g. 30 s) from the beginning, middle or end of the song for processing. This is motivated by performance issues, since processing only 30 seconds of a song is much faster than processing all of it, making it accessible for mobile processing and reducing energy consumption. This may immediately hinder the classification task because the most relevant parts of the song may not be in those segments. Therefore, methods to detect and extract the most relevant parts of the song (i.e. summarization) are needed in order to still address those performance and energy requirements while, at the same time, not compromising the classification accuracy too much.

Several algorithms to summarize music have been published, mainly for popular music songs, whose structure is repetitive enough. However, these algorithms were designed having in mind the goal of producing a thumbnail of a song as its summary the same way anyone considers an image thumbnail as that image summary. In other words, in this type of summary, the goal is to produce a shorter version (time-wise) of the original music file so that people can quickly get the gist of the whole piece without listening to all of it. Note that the word "people" was used in the previous sentence. This means that this type of summaries is human consumption oriented which, in turn, entails summary requirements such as coherence and

clarity that must be taken into account when extracting the summaries. There is a trade-off between these two requirements and the pair consisting of conciseness and non-redundancy, which means that these music-specific algorithms may not be ideal for automatic consumption oriented tasks, such as classification.

Generic summarization algorithms have also been developed and successfully applied in text and speech summarization. As these algorithms were not designed for music, their application, in music, to extract a thumbnail is not ideal because those human consumption extra requirements are not considered in these algorithms. This means that these algorithms will produce summaries which cannot be enjoyed by people as music. This is very promising because those summaries may avoid copyright issues inherent to a human oriented summary which, in turn, eases the dissemination of music datasets for MIR research. Nevertheless, these algorithms extract summaries that are both concise and diverse. Since human consumption is not our goal, generic summarization is worth considering for extracting music summaries for automatic consumption since it will only extract the most relevant information.

This thesis reviews summarization algorithms, both music-specific and generic, in order to summarize music for automatic instead of human consumption. This automatic consumption is a music classification task. Specifically, the influence of summarization on music classification is evaluated by comparing the accuracy of classifiers, when using the usual random segments of the song against using summaries of the same length. The idea is that a summary clip contains more relevant and less redundant information, thus, improving a classifier's performance. Results are presented on 2 classifiers showing that summarization improves music classification performance. The classification tasks are binary genre classification of Fado music and a multiclass genre classification of 5 different genres: Bass Music, Fado, Hip Hop, Indie Rock and Trance.

1.1 Thesis Structure

This document is organized as follows:

- Chapter 2 reviews related work on music-specific summarization. Structure-based strategies are presented in Section 2.1, followed by a clustering approach (Section 2.2), Average Similarity (Section 2.3) and Maximum Filtered Correlation in Section 2.4.
- Chapter 3 reviews related work on the following generic summarization algorithms: PageRank-based methods LexRank and TextRank (Section 3.1), Graph Random-walk with Absorbing StateS that HOPs among PEaks for Ranking (GRASSHOPPER) in Section 3.2, MMR (Section 3.3) and finally LSA in Section 3.4.
- Chapter 4 reviews classification setups from previous instances of MIREX on Section 4.1 and presents the approach taken for evaluating summarization in Section 4.2.

- Chapter 5 presents all experimental setups. Section 5.1 presents the datasets used in the experiments. Details regarding the selected algorithms for evaluation are presented in Section 5.2. Section 5.3 presents the parameterization and the classification features used in the experiments.
- Chapter 6 presents and analyzes the experimental results.
- Chapter 7 concludes this document, where the contributions of the performed experiments and future work are discussed.

2 Music Summarization

This section reviews music-specific summarization algorithms. Music summarization literature only evaluates (when it does) the extracted summaries from a human consumption point of view, i.e., it measures how easily people can identify the original song by hearing its summary or how well (in the subjective terms of conciseness, coherence etc.) the original song is summarized. This makes sense since these algorithms are developed for human consumption. Therefore, these state of the art algorithms are not evaluated in terms of automatic consumption (such as summarizing for classification purposes).

First, some Structure-based Strategies are presented. Then, a Clustering approach is discussed followed by a method based on the Average Similarity. The section is concluded with a method based on Maximum Filtered Correlation.

2.1 Structured-based Strategies

This type of summarization is done after the musical piece is segmented according to its structure. The assumption is that the song has a structure represented as a label sequence where each label represents a different part of the song (e.g. ABABCA where A is the chorus, B the verse and C the bridge). The summary can be extracted according to one of several strategies.

Chai (2006) presents two approaches to segmentation. The first segments the song using a Hidden Markov Model to detect key changes between frames. The second uses Dynamic Time Warping to detect recurrent structure. After segmentation, using one of the two approaches, the summary can be extracted in one of the following ways:

- SBS (Section-beginning Strategy): find the most repeated section, take the first of these and truncate its beginning as the summary;
- STS (Section-transition Strategy)
  - STS-I: choose the first transition such that the sum of the repeated times of the two sections is maximized;
  - STS-II: choose the first most repeated transition;
  - STS-III: choose the first transition right before the most repeated section.

To evaluate the summarization algorithm, the authors consider 5 different criteria: (1) the percentage of generated thumbnails that contain a vocal portion; (2) the percentage of thumbnails that contain the song's title; (3) the percentage of generated thumbnails that start at the beginning of a section; (4) the percentage of thumbnails that start at the beginning of a phrase and (5) the percentage of thumbnails that capture a transition between different sections.

Cooper and Foote (2003) achieve segmentation by correlating a Gaussian-tempered "checkerboard" kernel along the main diagonal of the similarity matrix which will output segment boundaries. After those boundaries are discovered, a segment-indexed similarity matrix is built. This matrix is like a regular similarity matrix but instead of having the similarity between every frame, it contains the similarity between every detected segment. Singular value decomposition is applied to the segment-indexed similarity matrix to find its rank-K approximation. Then the segments are clustered outputting the song's structure (each cluster representing a label). The summary can be generated by taking one segment per cluster (to avoid redundancy) or just a subset of these to satisfy temporal constraints. Usually one segment from each of the two clusters with largest singular values are selected. No evaluation was performed on this algorithm.

In (Peeters, La Burthe, and Rodet 2002; Peeters and Rodet 2003), the song is segmented in 3 stages. First, a similarity matrix is built using dynamic features and is analyzed in search for fast changes outputting segment boundaries. These segments are further processed by a K-Means algorithm to output the "middle states" (these are already labeled). Finally a Hidden Markov Model is applied on these states, to introduce time constraints, producing the final segmentation. Summarization can be achieved using strategies such as: providing an example of state transitions, providing a unique example of each state and providing only an example of the most important state. No evaluation was performed in these papers.

2.2 Clustering

This approach, first introduced by Chu and Logan (2000), uses clustering to group and label similar segments in order to find a summary for the musical piece. The input song is divided into non-overlapping 1-second segments. 13 Mel Frequency Cepstral Coefficient (MFCC)s are calculated for each frame. Note that a 1-second segment consists of a sequence of frames. Then, the segments are clustered using the following iterative procedure:

1. Every segment has its own cluster
2. Compute and store the distortion measure between every two clusters
3. Pick the pair with the lowest distortion

4. If the distortion is less than a predefined threshold, combine the two clusters and go to step 2.

5. If not, end

After this processing, the summary is extracted by taking the longest sequence of segments belonging to the same final and most frequent (with more segments in it) cluster in the musical piece. The distortion measure is a modification of the KL (Kullback Leibler) divergence. This modification ensures symmetry.

$$KL2\left(A;B\right) = KL\left(A;B\right) + KL\left(B;A\right) \tag{2.1}$$

The KL divergence can be calculated the following way:

$$KL\left(A;B\right) = E\left[\log\left(pdf\left(A\right)\right) - \log\left(pdf\left(B\right)\right)\right] \tag{2.2}$$

Assuming the pdfs A and B are Gaussians:

$$KL2\left(A;B\right) = \Sigma A/\Sigma B + \Sigma B/\Sigma A + \left(\mu A - \mu B\right)^2\left(1/\Sigma A + 1/\Sigma B\right) \tag{2.3}$$

Σ* and μ* denote variance and mean respectively. When a cluster B does not have enough data points, the Mahalanobis distance is used:

$$M\left(A;B\right) = \left(\mu A - \mu B\right)^2/\Sigma A \tag{2.4}$$

This algorithm has been successfully tested with subjective evaluations from people that would rate the summaries as "good", "average" or "bad". This evaluation is, therefore, human consumption oriented. The algorithm's complexity is $O\left(n^2\right)$ making it very expensive as the initial number of clusters (initial segments) increases. The initial segmentation limits the algorithm's resolution. This effect of hard segmentation can be mitigated by reducing the size of the segments or segmenting the song according to its tempo.

2.3 Average Similarity

This is another "thumbnail" approach. Its goal is finding a fixed-length continuous music segment, of duration L, most similar to the entire song. This method was introduced by Cooper

and Foote (2002) and later used in other research efforts such as that from Glaczynski and Lukasik (2011). The method consists of building a similarity matrix for the song and then calculating an aggregated/average measure of similarity between the whole song and every L seconds long segment.

Cooper and Foote (2002) compute 45 MFCCs but only the fifteen coefficients with highest variance are kept as signal features. The cosine distance measure is used to calculate the pairwise similarity of every frame and build the similarity matrix.

Glaczynski and Lukasik (2011) use the first 13 MFCCs and the spectral centre of gravity (sound "brightness"). After some tests, the *Tchebychev* distance was picked for building the similarity matrix, as it yielded the most satisfactory results.

Once the similarity between every frame is calculated, a similarity matrix, with elements $S(i,j)$ equal to the similarity value between feature vectors (from frames i and j) v_i and v_j, is built:

$$S(i,j) = s(v_i, v_j) \tag{2.5}$$

The average similarity measure can be calculated by summing up columns (or rows, since the similarity matrix is symmetric) of the similarity matrix, according to the desired summary length L, starting from different initial frames. The maximum score will correspond to the segment that is most similar to the whole song. To find the best summary of length L, the score $Q_L(i)$ must be computed:

$$Q_L(i) = \bar{S}(i, i+L) = \frac{1}{NL} \sum_{m=i}^{i+L} \sum_{n=1}^{N} S(m,n) \tag{2.6}$$

Where N is the number of frames in the entire piece. The index of the best summary starting frame is:

$$q_L^* = \arg \max_{1 \leq i \leq n-L} Q_L(i) \tag{2.7}$$

The evaluations of this method in the literature are subjective (human) evaluations that take into account whether the generated summaries include the most memorable part(s) (also called *hooks*) of the song (Cooper and Foote 2002). Other subjective evaluations are averages of scores given by test subjects regarding specific qualities of the summary such as Clarity, Conciseness and Coherence (Glaczynski and Lukasik 2011).

2.4 Maximum Filtered Correlation

This approach, introduced by Bartsch and Wakefield (2005), also extracts "thumbnails" from songs. Given a summary duration L, it finds the most strongly repeated section in the song. This method consists of building a similarity matrix, then building a *filtered time-lag matrix* and finally extracting the most strongly repeated section of the song.

Bartsch and Wakefield (2005) use the chromagram vectors as the signal frame features. The similarity matrix is built according to the correlation between each feature vector. Specifically, the element $S(i,j)$ is calculated as:

$$S(i,j) = v_i^T v_j \tag{2.8}$$

After building the similarity matrix, the similarity between extended segments separated by a constant lag is calculated and embedded in a *filtered time-lag matrix*. The calculation is as follows:

$$F(i,j) = \sum_k S(i+k, i+j+k)\, w(k) \tag{2.9}$$

Where $w(k)$ is a symmetric rectangular windowing function defining the impulse response of the filter. $F(i,j)$ denotes the similarity between the segment starting at the ith frame and the segment starting at the $(i+j)^{th}$ frame. The support size of $w(k)$ is chosen according to the desired length of the "thumbnail" as it will determine the length of the segments to be compared. There are other possible window functions but the uniform moving average filter yielded the best performance according to the authors' experiments.

To extract the "thumbnail", the maximum value in F must be found. The row index corresponding to the maximum value and length of the filtering window define the position and length of the summary. However, some heuristics were applied here to improve the results: given the most similar pair of segments, the algorithm will select the first of them as the summary; there is a minimum lag value empirically set to one-tenth of the song's length and the upper limit for the starting time of the summary is three-fourths of the song's length.

This method was evaluated by manually selecting truth interval(s) in the song and then comparing the extracted summary with those intervals. The comparison takes into account how much of the truth intervals is contained in the summary and how much of the summary is not contained in the intervals.

2.5 Summary

This section reviewed several music specific summarization algorithms used for creating music thumbnails for human enjoyment. Though these algorithms are not suitable for extracting summaries for automatic consumption, they do offer some insight on feature selection for music information. Specifically, MFCCs are widely used for these summarization tasks as they are good for modeling the timbre of the sound.

3 Generic Summarization

In this chapter, unsupervised approaches to summarization are presented. Most of these algorithms were developed and applied in the context of speech and text document summarization. First, some PageRank-based methods are discussed: LexRank and TextRank. Then the GRASSHOPPER algorithm is reviewed followed by MMR. This section is concluded with LSA.

3.1 PageRank-based

Centrality-as-Relevance methods detect the most relevant sentences by detecting the most central ones. Within this family Centroid-based methods can be found which basically assert that the most relevant sentences are the ones most similar to a centroid (pseudo) sentence. Early work on multi-document summarization by Radev, Jing, and Budzikowska (2000) represents each document as Term Frequency - Inverse Document Frequency (TF-IDF) (term frequency-inverse document frequency) vectors and clusters them. Then, the centroid of each cluster is also a TF-IDF vector with values corresponding to the weighted average of the vectors of the documents in the cluster. Sentences containing words of the centroids are the best representatives of that cluster and hence the best candidates for the summary.

$$centrality\,(x) = similarity\,(x, centroid) \tag{3.1}$$

Another type of centrality-based methods relies on the similarity between every sentence instead. LexRank (Erkan and Radev 2004) and TextRank (Mihalcea and Tarau 2004) are two of them. These centrality-based methods are based on Google's PageRank (Brin and Page 1998) algorithm for ranking web pages. The output is a list of ranked sentences from which it is possible to extract the highest ranked ones to produce a summary. Both of these methods were developed for text summarization. First, all sentences are compared, normally represented as TF-IDF scores vectors, to each other using a similarity measure. LexRank uses the cosine similarity while TextRank uses a measure based on the sentences' overlap (terms that appear in both sentences) normalized by the length of each sentence. After this step, a graph is built where each sentence is a vertex and edges are created between every sentence according to their pairwise similarity. Usually the similarity score must be higher than some threshold to create

an edge. LexRank was experimented with both weighted and unweighted edges. TextRank uses continuous similarity scores. Then, the following calculation is performed iteratively for each vertex until convergence is achieved (when the error rate of two successive iterations is below a certain threshold for every vertex):

$$S(V_i) = \frac{(1-d)}{N} + d \times \sum_{V_j \in adj[V_i]} \frac{Sim(V_i, V_j)}{\sum_{V_k \in adj[V_j]} Sim(V_j, V_k)} S(V_j) \tag{3.2}$$

Where d is a damping factor to guarantee the convergence of the method, N is the total number of vertices and $S(V_i)$ is the score of vertex i. Note that this is the case where edges are weighted. If using unweighted edges the equation is simpler:

$$S(V_i) = \frac{(1-d)}{N} + d \times \sum_{V_j \in adj[V_i]} \frac{S(V_j)}{D(V_j)} \tag{3.3}$$

Where $D(V_i)$ is the degree of vertex i (i.e. number of edges/connected vertices). A summary can be constructed by taking the highest ranked sentences until a certain summary length is reached.

These methods are based on the fact that sentences recommend each other. A sentence very similar to many sentences will get a high score. Moreover, a sentence score is also determined by the score of the sentences recommending it.

3.2 GRASSHOPPER

The GRASSHOPPER (Zhu, Goldberg, Gael, and Andrzejewski 2007) is a method that has been successfully applied in both text summarization and social network analysis. Its focus is on improving diversity in ranking.

As input, the algorithm takes a graph W represented by an $n \times n$ weight matrix where w_{ij} is the weight on the edge from i to j, a probability distribution r in which prior ranking can be encoded (in this case, GRASSHOPPER can be viewed as a re-ranking method) and a weight $\lambda \in [0, 1]$ to balance between W and r. The first ranking sentence is found by teleporting random walks. This is very similar to the PageRank-based methods. A $n \times n$ raw transition matrix $\tilde{P}$ is created by normalizing the rows of W. In matrix notation, the teleporting random walk P is defined as follows:

$$P = \lambda \tilde{P} + (1-\lambda) 1r^T \tag{3.4}$$

Where $1r^T$ is the outer product of an all-1 vector and the prior ranking. P has a unique sta-

tionary distribution $\pi = P^T\pi$. The first ranking sentence will be the state with largest stationary probability $\arg\max_{i=1}^{n} \pi_i$.

Instead of taking the following highest ranked sentences, GRASSHOPPER uses a different approach that guarantees diversity among the selected sentences. It consists of turning the already selected sentences into absorbing states, calculating the number of expected visits in every remaining state and taking the one that maximizes it. States are turned into absorbing states by setting $P_{gg} = 1$ and $P_{gi} = 0, \forall i \neq g$. P can be arranged so that ranked sentences are listed before unranked ones:

$$P = \begin{bmatrix} I_G & 0 \\ R & Q \end{bmatrix} \tag{3.5}$$

Where G is the set of sentences ranked so far and I_G is the identity matrix on that set. R and Q correspond to rows of unranked sentences from Equation 3.4. The *fundamental matrix* N must be calculated to get the expected number of visits. N_{ij} is the expected number of visits to state j, if the random walk started at state i:

$$N = (I - Q)^{-1} \tag{3.6}$$

Then, the average over all starting states is taken to get the expected number of visits to state j, v_j. The sentence to be selected is the one that satisfies:

$$\arg \max_{i=|G|+1}^{n} v_i \tag{3.7}$$

After selecting the sentence, its state becomes an absorbing state and this process is repeated until all sentences are ranked.

3.3 Maximum Marginal Relevance

This method, introduced by Carbonell and Goldstein (1998), selects sentences from the original signal according not only to their relevance, but also according to their diversity against the already selected sentences in order to output low-redundancy summaries. This approach has been used in speech summarization by Zechner and Waibel (2000), Murray, Renals, and Carletta (2005). It is a query-specific summarization method, though it is possible to produce generic summaries out of it by taking, for instance, the centroid vector of all the sentences (as in (Murray, Renals, and Carletta 2005)) as the query.

MMR works by iteratively selecting the sentence that satisfies the following mathematical model:

$$\arg\max_{S_i} \left[\lambda \left(Sim_1 \left(S_i, Q \right) \right) - \left(1 - \lambda \right) \max_{S_j} Sim_2 \left(S_i, S_j \right) \right] \tag{3.8}$$

Where Sim_1 and Sim_2 are the, possibly different, similarity metrics; S_i are the unselected sentences and S_j are the previously selected ones; Q is the query and λ is a configurable parameter that allows the selection of the next sentence to be based on its relevance, its diversity or a linear combination of both. In other words, in each iteration, this method selects the sentence, from the set of sentences which have not been previously selected, which has the highest score in that iteration. That score is a linear combination of 2 scores: relevance, represented by the similarity of a sentence to the query sentence; and diversity, represented by the similarity of a sentence to its most similar already selected sentence. Usually, sentences are represented as TF-IDF scores vectors. The cosine similarity is frequently used as Sim_1 and Sim_2.

$$sim_{cos} \left(x, y \right) = \frac{x \cdot y}{||x||||y||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{3.9}$$

3.4 Latent Semantic Analysis

LSA is a method based on a mathematical technique called Singular Value Decomposition (SVD) that was first used for generic text summarization by Gong and Liu (2001). SVD is used to reduce the dimensionality of an original matrix representation of the text. To perform LSA-based text summarization a T terms by N sentences matrix must be built:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{T1} & \cdots & a_{TN} \end{bmatrix} \tag{3.10}$$

Each element of A has two weight components: a local weight and a global weight. The local weight is a function of the number of times a term occurs in a specific sentence and the global weight is a function of the number of sentences that contain a specific term.

$$a_{ij} = L_{ij} G_i \tag{3.11}$$

Then, SVD is applied to matrix A, which will result in a decomposition formed by three matrices: U, a $T \times N$ matrix of left singular vectors (its columns); Σ, a $N \times N$ diagonal matrix of singular values; and V^T, a $N \times N$ matrix of right singular vectors (its rows).

$$A = U \Sigma V^T \tag{3.12}$$

It turns out that the singular values are sorted by descending order in matrix Σ. These singular values determine topic relevance. Each latent dimension corresponds to a topic. The Rank K approximation is calculated by taking the first K columns of U, the $K \times K$ sub-matrix of Σ and the first K rows of V^T. Then, the most relevant sentences are extracted by iteratively selecting sentences corresponding to the indices of the highest values for each (most relevant) right singular vector.

Steinberger and Jezek (2004) discuss two limitations of this approach: the fact that K is equal to the number of sentences in the summary, which as it increases, it tends to include less significant sentences; and that sentences with high values in several dimensions (topics), but never the highest, will never be included in the summary. To compensate for these problems, a sentence score was introduced and K is chosen so that the Kth singular value does not fall under half of the highest singular value.

$$score\,(j) = \sqrt{\sum_{i=1}^{k} v_{ij}^2 \sigma_i^2} \tag{3.13}$$

3.5 Summary

This section reviewed several generic summarization algorithms which were successfully applied to other media such as text and speech. These algorithms extract the most relevant and diverse information (according to each algorithm's definition of relevance and diversity) from the input, without taking other requirements (such as coherence or clarity) into account. This is done by scoring and ranking sentences in a sentence-based representation of the signal and then picking the ones with highest scores to include in the summary. This means that some mapping from frame representation to sentence representation must be done prior to applying this algorithms to music.

4 Music Classification

This chapter starts by presenting music classification setups from previous editions of MIREX and finishes with an explanation of the approach taken to evaluate the impact of summarization on music classification.

4.1 MIREX Setups

Music Classification is the task of assigning a class to songs. Theses classes usually represent semantic information about the piece. Usual tasks include Genre/Mood Classification and Artist/Composer Identification. In this section, some interesting music classification evaluation setups are reviewed for the previously mentioned tasks.

The system described by Mandel and Ellis (2005) was ranked first in MIREX 2005 on the Audio Artist Identification task. This task consists of training a classifier with songs from several artists and then classify different songs in terms of their corresponding artist/group. The datasets used in the evaluation of the system were the *Magnatagatune* and the *uspop2002* datasets. *Magnatagatune* contains over 25000 clips from over 400 albums and 200 artists. *uspop2002* contains a set of 706 albums an 8764 tracks from 400 artists. This Support Vector Machine (SVM) classifier is based on 20-coefficient MFCC features. The system's performance was measured calculating its confusion matrices (explained in Section 4.2.2) from which an overall accuracy of 72.45% (3.88% more than the second ranked submission) was calculated.

In the same edition of MIREX, another system by Bergstra, Casagrande, and Eck (2005) was the first ranking system on the task of Audio Genre Classification. This task is similar to the previously described one except the system must determine each song's genre. The datasets used in these experiments were the same as for Artist Identification. However, this classifier is based on a different set of (timbral) frame-level features such as RCEPS, MFCCs, Linear predictive coefficients and Low-frequency Fourier magnitudes. The same (final) metrics as for Artist Identification were calculated for this task (confusion matrices, accuracies). The overall accuracy measured in MIREX 2005 for this system was 82.34% (3.53% more than the second ranked submission).

The system that best performed in MIREX 2013 Audio Classical Composer Identification task is described by de Leon and Martinez (2013). This task is the same as Audio Artist Identification except for the fact that only classical music pieces (and hence composers) are considered.

The dataset used for this task consists of 2772 clips from 11 composers (252 clips per composer) such as Bach and Mozart. This system also relies on a lot of timbral features such as MFCCs, Spectral Contrast, Sub-band flux and a lot of other spectral features. The confusion matrix was also computed and the overall accuracy of the system was 70.31% (0.61% more than the second placed submission).

In that same year, an upgraded system by Wu (2013), previously submitted in MIREX 2011 and 2012, was the best performing system in both of the tasks: Audio US Pop Music Genre Classification and Audio Mood Classification. The first task consists of assigning a genre label to each musical piece such as Blues, Jazz and Rock. The second task consists of classifying a song according to its mood such as passionate, humorous, fun and witty. Each of these tasks has its own dataset in order to evaluate the performance of the submitted algorithms. The first is the MIREX 2007 Genre dataset which consists of 7000 clips from 10 different genres (700 clips per genre). The second is the MIREX 2007 Mood dataset which consists of 600 clips from 5 different mood categories (120 clips per category). This system relies not only on acoustic features but also on visual features. The acoustic feature used is the GSV (Gaussian Super Vector) which is based on the lower-level feature MFCC. The visual features employed in this system are song-level-based texture features and beat-level-based texture and heterogeneity features. The same metrics as stated before were computed for these tasks. This system achieved an accuracy of 68.33% and 76.23% for Mood and Genre classifications respectively (0.5% and 1.26% more than the second placed submissions).

Since the inception of MIREX (2005) until the writing of this thesis (2014) the evaluation procedures and metrics haven't changed much. The datasets themselves are rarely updated unless a new task is defined (it wouldn't make sense to use a mixed genre dataset in a classical composer identification task). However, the features used for classification evolved over the years and are still an active research topic.

4.2 Approach

In order to evaluate the summarization's influence on music classification, a baseline needs to be established for comparison. To get to that baseline, a music dataset with ground truth information (classification-wise) attached to it is necessary. 3 different datasets were used for this: 2 Fado datasets, which contain 250 Fado songs (shared by both of them) and 250 non-Fado songs (these are different between them); and a 5-class dataset of 1250 songs (250 songs per genre) consisting of Bass Music, Fado, Hip hop, Indie Rock, and Trance.

After getting the datasets, the classifier's performance is evaluated by testing it, on those datasets, using the usual segments (in our experiments, the beginning, middle, and end) of the songs. These evaluations will be the baselines (for each dataset). After this, every piece in the datasets is summarized. Then, the classification performance is evaluated again, using the

summaries, instead of the previous segments (of the same duration). Now, a comparison can be made between those evaluations to check for improvements in classification accuracy. This is done for each summarization method and parameter combination (of those methods) chosen for evaluation. The evaluation setup will be similar to MIREX's. K-fold cross-validation was used and classification accuracy was computed.

4.2.1 Cross-Validation

Cross-validation is a model validation technique whose goal is to define a validation dataset to test the model during the training phase. It is used to limit problems such as overfitting and to give an estimation on how generalized the model is against unknown datasets.

K-fold cross-validation partitions the training set into k preferably equally-sized subsets. Then, iteratively, the model is trained with $k-1$ subsets and tested with the remaining subset. This is done once per subset (i.e. each subset is used exactly once as the validation/test subset). The results of the tests can be averaged to output a single performance measure.

4.2.2 Confusion Matrix

A confusion matrix, also called error matrix, is a table that allows visualization of a classifier's performance. It relates the true number of instances per class (as its rows) with the predicted number of instances per class (as its columns). The sorting of rows and columns is the same so the ideal confusion matrix (with maximum accuracy) is a diagonal matrix.

4.2.3 Accuracy

Accuracy is a measurement that indicates how close a measured quantity value is to the corresponding true value. It is a combination of precision and trueness. Precision indicates how close several measured quantity values of the same object are. Trueness, on the other hand, means how close the average of several measured quantity values is to a reference quantity value. We can measure the accuracy of a classifier by calculating the ratio of correct classifications using the confusion matrix values:

$$Accuracy = \frac{\sum_{k=1}^{N} M_{k,k}}{\sum_{i=1}^{N} \sum_{j=1}^{N} M_{i,j}} \tag{4.1}$$

Where M is the confusion matrix, $M_{i,j}$ is the element of M in row i and column j and N is the the number of rows/columns of matrix M.

4.3 Summary

This section reviewed some music classification systems and setups from previous MIREX editions. These setups provide insight into feature selection for classification and evaluation measures, defined by the MIR community, for the classification task itself.

5 Experimental Setup

This chapter describes the experimental setup. Two classifiers were used, along with different datasets. The experimental setup was also different between both experiments so they are presented separately.

5.1 Datasets

Our experiments covered 2 classifiers: a binary Fado classifier and a 5-way genre classifier. The binary classifier was tested on 2 datasets and the multiclass genre classifier was tested on 1 dataset.

5.1.1 Binary Datasets

Dataset 1 consists of 500 songs: 250 Fado songs and 250 songs from 10 other balanced genres (Classical, Rock, Jazz, Pop, Folk, Medieval, Blues, Country, Tribal, Electronic) (Antunes, de Matos, Ribeiro, and Trancoso 2014). Dataset 2 contains the same 250 Fado songs but the non-Fado half of the dataset is mostly composed of pieces from the Renaissance (around 180), which are timbrically very similar to Fado (making it more difficult to correctly classify this dataset), and some pieces from the 10 genres mentioned before. Both datasets are encoded in mono, 16-bit, 22050 Hz Microsoft WAV files.

5.1.2 5-class Genre Dataset

This dataset used consists of a total of 1250 songs from the 5 different genres (or classes) described in Section 5.1.3. Each class is represented by 250 songs from several artists within each genre. Some artists present in the dataset are Datsik, Savant and Skrillex on Bass Music; Amália Rodrigues, António Zambujo and Mariza on Fado; 2Pac, Kanye West and Nas on Hip Hop; The Antlers, Beirut and Jack Johnson on Indie Rock; Armin van Buuren, W&W and Tiësto on Trance. The dataset is encoded in mono, 16-bit, 22050 Hz Microsoft WAV files.

5.1.3 Musical Classes

The classes present in the 5-class dataset are Bass Music, Fado, Hip Hop, Trance, and Indie Rock. Bass Music is a generic term referring to several specific styles of electronic music, such as Dubstep, Drum and Bass, Electro and more. These styles have similar timbral characteristics such as deep basslines and the "wobble" bass effect. However, the tempo of these genres is different, but the features used for classification (in this classifier) do not take that musical feature into account. Fado is a portuguese music genre whose instrumentation consists solely of stringed instruments, such as the classical guitar and the Portuguese guitar. Hip Hop consists of drum rhythms (usually built with samples), the use of turntables and spoken lyrics on top of it. Indie Rock usually consists of guitar, drums, keyboard and vocal sounds and was influenced by punk, psychedelia, post-punk and country. Trance is an electronic music genre characterized by repeating melodic phrases, and a musical form that builds up and down throughout a track. In that sense, it is similar to Bass Music. However, Bass Music contains much more bass sounds and distortion making it timbrically different from Trance.

5.2 Summarization Algorithms Details

The algorithms chosen for evaluation were the Average Similarity, LexRank LSA and MMR. Every algorithm was implemented in C++ using several libraries: OpenSMILE (Eyben, Weninger, Gross, and Schuller 2013) was used for feature extraction (for summarization), Armadillo (Sanderson 2010) was used for matrix operations, and Marsyas (Tzanetakis and Cook 1999) was used for synthesizing the summaries.

Average Similarity, a music-specific summarization algorithm, was chosen for comparison purposes. As stated before, this type of algorithm extracts summaries for human listening. This algorithm's implementation is straightforward as the only thing that needs to be done is to store frame-wise similarities in a matrix and then iterate it to compute average scores of segments. The parameters of this algorithm are: features (type/size); and framing (frame and hop sizes).

Applying generic summarization algorithms to music requires additional steps. Since these algorithms operate on the concepts of word and sentence, some processing must be done to map the frame representation obtained after feature extraction to a word/sentence representation. For each song being summarized, a vocabulary is created, through clustering, using the frames feature vectors. The mlpack's (Curtin, Cline, Slagle, March, Ram, Mehta, and Gray 2013) implementation of the K-Means algorithm was used for this step. After clustering, a vocabulary of musical words is obtained (each word is a frame cluster centroid). From now on, each frame is assigned its own cluster centroid, effectively mapping the frame feature vectors to a word from the vocabulary. This mapping changes the real/continuous nature of each frame

(when represented by a feature vector) to a discrete nature (when represented as a word from a vocabulary). Then, the song is segmented into fixed-size sentences (e.g., 5-word sentences). Since every sentence contains discrete words from a vocabulary, it is possible to represent each as a vector of word occurrences/frequencies (depending on the type of weighting chosen), which is the exact representation used by generic summarization algorithms. Sentences are compared using the cosine distance. The parameters of any of these algorithms include: features; framing; vocabulary size (final number of clusters of the K-Means algorithm); weighting (e.g. TF-IDF); and sentence size (number of words per sentence).

In MMR's implementation, the similarity between every sentence is computed only once and then the algorithm is iteratively applied until the desired summary length is reached. This algorithm has an additional parameter: λ which is a value between 0 and 1 allowing for the selection of the next sentence to be based on a linear combination of relevance and diversity.

In LexRank's implementation, the similarity between every sentence is also computed only once and then the sentence scores are iteratively updated by the algorithm until convergence. Then, sentences are picked until the desired summary length is reached. LexRank has additional parameters that were fixed during all experiments: damping factor (0.85) and the convergence threshold (0.0001).

LSA's implementation uses the Armadillo's (Sanderson 2010) implementation of the SVD operation. After sentence/word segmentation, SVD is applied to the term by sentences matrix (column-wise concatenation of all sentence vectors). Then, the rank-K approximation of the decomposition is taken where the Kth singular value is not smaller than half of the $(K-1)$th singular value. Sentence scores are calculated, according to Equation 3.13, and sentences are picked according to the scores until the desired summary length is reached.

5.3 Classification Tasks

The experimental setup was different between the binary and the 5-class classifiers. Both classifiers are SVMs (Chang and Lin 2011). However, each uses a different set of features for classification. For each experiment, we also calculate classification accuracy for 30 seconds continuous segments extracted from the beginning, middle, and end of the songs as well as the full songs themselves for comparison purposes.

5.3.1 Binary Classification Setup

Average Similarity was tested with 6 different framing schemes: 3 different sizes (0.25, 0.5 and 1 s) with 50% and no overlap. MFCC vectors of sizes 12 and 24 were used.

MMR was tested with 2 values for λ (0.5, and 0.7) and 2 different weighting types: binary (term presence) and "dampened" TF-IDF (same as TF-IDF but takes logarithm of TF instead of

TF itself). LexRank was tested for the same weighting types as for MMR. LSA was tested with raw and binary weighting.

LexRank, LSA and MMR were all tested for the following parameter values: frame/hop sizes pairs of 0.25/0.25, 0.5/0.25, and 0.5/0.5 (in seconds); vocabulary size of 25, 50, and 100 words; and sentence size of 5, 10, and 20 words. MFCC vectors (of size 12) were used as features for these experiments as they are widely used in many MIR tasks including music summarization in (Cooper and Foote 2003; Chu and Logan 2000; Cooper and Foote 2002; Glaczynski and Lukasik 2011). 5-fold cross validation was used for calculating classification accuracy in these experiments. For each summarization parameter combination presented, classification was done using the first 10, 20 and 30 seconds of the 30 seconds summaries.

5.3.2 Multiclass Classification Setup

As some insight on parameter selection was gained after the first (binary) experiment, different parameter values were chosen for testing in this experiment.

Average Similarity was, again, tested with 6 different framing schemes: 3 different sizes (0.25, 0.5 and 1 s) with 50% and no overlap. MFCC vectors of sizes 12, 20, and 24 were used.

MMR was tested with 2 values for λ (0.5 and 0.7) and the "dampened" TF-IDF weighting was used. LexRank was tested using binary and "dampened" TF-IDF weighting. LSA was tested with binary weighting.

LexRank, LSA and MMR were tested with all combinations of the following parameter values: vocabulary sizes of 25, 50, and 100 words; and sentence sizes of 5, 10, and 20 words. The framing was fixed on these experiments: frames of 0.5s with no overlap. MFCC vectors (of sizes 12 and 20) were used as features. Another set of features was also tested: 12/20 MFCCs concatenated with the 9 features enumerated in Section 5.3.4. This allowed for comparing results using these 2 sets in both the summarization and classification steps of the experiment. 10-fold cross validation was used for calculating classification accuracy.

5.3.3 Binary Classification Features

The features used by this SVM consist of a 32-dimensional vector per song, which is a concatenation of 4 features: average vector of the first 13 MFCCs; Root Mean Square (RMS) energy; high frequencies 9-dimensional rhythmic features; and low frequencies 9-dimensional rhythmic features (Antunes, de Matos, Ribeiro, and Trancoso 2014). This feature set is fixed during all these experiments and it is used exclusively for classification. The rhythmic features are computed from Fast Fourier Transform (FFT) coefficients on the 20 Hz to 100 Hz range (low frequencies) and on the 8000 Hz to 11025 Hz range (high frequencies). Assuming v is

a matrix of FFT coefficients with frequency varying through columns and time through lines, each component of the 9-dimensional vector is:

- $maxamp$: max of the average v along time
- $minamp$: min of the average v along time
- number of v values above 80% of $maxamp$
- number of v values above 15% of $maxamp$
- number of v values above $maxamp$
- number of v values below $minamp$
- mean distance between peaks
- standard deviation of distance between peaks
- max distance between peaks

Fado does not have much information in the low frequencies as it does not contain, for example, drum kicks. Furthermore, due to the string instruments used, Fado information content is higher in the high frequencies, making these features good for distinguishing it from other genres. These features were extracted in a Matlab implementation (Antunes, de Matos, Ribeiro, and Trancoso 2014).

5.3.4 Multiclass Classification Features

The features used by this SVM consist of a 38-dimensional vector per song, which is a concatenation of several statistics, over all frames of the song, of features used in (de Leon and Martinez 2013). The average of the first 20 MFCCs concatenated with statistics (average and variance) on these features describe the timbral texture of the song:

- Spectral Centroid
- Spectral Spread
- Spectral Skewness
- Spectral Kurtosis
- Spectral Flatness
- Spectral Flux

- Spectral Rolloff
- Spectral Brightness
- Spectral Entropy

These statistics are computed based on those features' extraction on 50 ms frames with no overlap. This set of features and a smaller set only containing the MFCCs averages were tested in classification. All musical genres in our dataset differ in timbre from each other making this set of timbral features a good candidate for classification. These features were extracted with OpenSMILE (Eyben, Weninger, Gross, and Schuller 2013).

5.4 Summary

This section detailed experimental setups. A description of the datasets used and the classes they represent was provided. Then, some details about the adaptation of the generic algorithms to music were given along with library dependencies for running the experiments. The section ended with an enumeration of all parameters tested as well as what features were used for classification.

6 Results

This chapter presents the most interesting results from the performed experiments. Since the evaluation setup for each different classifier was different, the results are presented separately. The "Frame/Hop Size" columns indicate the frame/hop sizes in seconds, which can be interpreted as overlap (e.g., the pair 0.5, 0.25 stands for frames of 0.5s duration with a hop size of 0.25s, which corresponds to a 50% overlap between frames). The "Usage" column indicates how much of the summary was processed for classification (i.e. how many seconds were used for feature extraction in the classification step), therefore this variable is not a summarization parameter but rather a classification one.

6.1 Binary Classification Results

This experiment involved 2 different datasets (as explained in Section 5.1.1) so each table presented here provides 2 Accuracy columns (1 and 2) corresponding to the Accuracy obtained for that algorithm/parameter combination for dataset 1 and dataset 2, respectively.

Table 6.1: Binary Classification Baselines

Section	Accuracy 1 (%)	Accuracy 2 (%)
beginning	86.4	88.8
middle	**91.8**	**92.2**
end	87.6	88.2
full	92.2	91.0

Table 6.1 presents the baseline results. These results are obtained when classification is done using the usual beginning, middle and end segments of the songs (with 30 seconds duration) for both binary datasets. These are the values used for comparison. More specifically, the middle section results will be used for comparison since they were the best in both datasets. These accuracies were 91.8% and 92.2%, on datasets 1 and 2, respectively. An experiment which uses full songs was also done. Classifying with full songs is very slightly better for dataset 1 (92.2%) but worse on dataset 2 (91%). This suggests that dataset 2 is, in average, more difficult to classify (i.e. the classes represented are more similar). In fact, the non-Fado half of dataset 2 is timbrically more similar to Fado, as discussed in Section 5.1.1. However, summarizing

dataset 2 by simply extracting the 30 seconds middle segments, already proves to be beneficial both in terms of processing time as well as in accuracy, with a 1.2% improvement against using full songs. On dataset 1, although using the middle sections decreased accuracy, it did so only by 0.4%, a reasonable price to pay for getting faster classification processing during feature extraction. These results support the use of segments, which basically are simple summaries, instead of the whole audio signal.

Table 6.2: Binary Classification Average Similarity Results

# MFCC	Frame Size (s)	Hop Size (s)	Accuracy 1 (%)	Accuracy 2 (%)
12	0.25	0.125	88.4	88.6
24	0.25	0.125	89.0	88.8
12	0.25	0.25	89.0	88.2
24	0.25	0.25	88.2	89.6
12	0.5	0.25	89.6	88.8
24	0.5	0.25	89.4	89.6
12	0.5	0.5	**90.2**	88.6
24	0.5	0.5	88.8	89.0
12	1.0	0.5	88.8	88.4
24	1.0	0.5	89.2	**89.8**
12	1.0	1.0	88.6	89.0
24	1.0	1.0	88.6	88.8

Table 6.2 presents some results for the Average Similarity algorithm. A total of 12 different summarization parameter combinations was tested. The results presented here use the whole 30 seconds summaries and represent all tested combinations for this experiment.

Average Similarity was not able to improve classification performance for any combination tested, yielding best scores of 90.2% (row 7 of Table 6.2) and 89.8% (row 10 of Table 6.2), on datasets 1 and 2, respectively. This represents a decrease in accuracy of about 1.6% on dataset 1 and 2.4% on dataset 2. Average Similarity, however, is designed to produce a thumbnail of a song for human consumption. This constraint forces the algorithm to extract a continuous segment (that is most similar to the whole song, according to an average measure of similarity), which hinders the extraction of the most important and non-redundant parts of the songs.

Table 6.3 presents some results for LexRank. A total of 54 summarization parameter combinations was tested with MFCC vector size fixed at 12. The results presented include the best combinations for each dataset and variations of those combinations in all possible directions (parameters), one at a time.

On dataset 1, LexRank improved classification accuracy for 3 out of 54 combinations of parameter values. The best accuracy obtained this way was 92.6% (row 16 of Table 6.3) which corresponds to a 0.8% increase.

Table 6.3: Binary Classification LexRank Results

Frame/Hop Size (s)	Vocabulary Size	Sentence Size	Weighting	Usage (s)	Accuracy 1 (%)	Accuracy 2 (%)
0.25 / 0.25	25	20	dampTF	20	91.0	93.0
0.25 / 0.25	100	20	dampTF	30	90.0	91.0
0.5 / 0.25	25	20	dampTF	10	89.2	92.4
0.5 / 0.25	25	5	dampTF	20	88.8	91.6
0.5 / 0.25	25	10	dampTF	20	89.2	91.6
0.5 / 0.25	25	20	binary	20	90.0	92.4
0.5 / 0.25	25	20	dampTF	20	90.0	**94.4**
0.5 / 0.25	25	20	dampTF	30	90.0	92.2
0.5 / 0.25	50	20	dampTF	20	90.0	91.8
0.5 / 0.25	50	20	dampTF	30	90.4	92.4
0.5 / 0.25	100	5	dampTF	30	89.0	90.2
0.5 / 0.25	100	10	dampTF	30	90.6	93.6
0.5 / 0.25	100	20	binary	30	92.0	93.2
0.5 / 0.25	100	20	dampTF	10	89.6	91.0
0.5 / 0.25	100	20	dampTF	20	90.8	93.6
0.5 / 0.25	100	20	dampTF	30	**92.6**	93.4
0.5 / 0.5	25	20	dampTF	20	89.6	91.0
0.5 / 0.5	100	20	dampTF	30	90.8	91.2

On dataset 2, LexRank improved classification accuracy for 34 out of 54 combinations. Considering this dataset is more difficult to classify, this means that summarization is extracting relevant information for classification. The best result was 94.4% (row 7 of Table 6.3), corresponding to an increase of 2.2%.

LexRank had a bigger impact on dataset 2, just like taking the middle section. This suggests that summarizing the signal is even better for more "difficult" datasets. Though "dampened" TF-IDF weighting is not always better than binary weighting, the best combinations in both datasets use it.

Table 6.4 presents some results for LSA. A total of 54 summarization parameter combinations were tested. MFCC vector size was fixed at 12. The results presented include the best combinations for each dataset and variations of those combinations in all possible directions (parameters), one at a time.

For dataset 1, LSA improved classification accuracy for only 1 combination of parameters. The best accuracy obtained was 92.6% (row 7 of Table 6.4), corresponding to a 0.8% increase.

For dataset 2, 24 out of 54 combinations of parameters yielded an improvement in classification accuracy. Interestingly, those combinations were all using binary weighting. No combination using raw weighting improved accuracy (in either dataset). In fact, any tested weighting that uses term frequency does not work well for LSA when applied to music. This is

Table 6.4: Binary Classification LSA Results

Frame/Hop Size (s)	Vocabulary Size	Sentence Size	Weighting	Usage (s)	Accuracy 1 (%)	Accuracy 2 (%)
0.25 / 0.25	25	20	binary	30	90.4	93.2
0.25 / 0.25	50	20	binary	30	90.0	92.2
0.25 / 0.25	100	5	binary	30	89.2	93.4
0.25 / 0.25	100	10	binary	30	90.6	91.8
0.25 / 0.25	100	20	binary	10	88.8	91.0
0.25 / 0.25	100	20	binary	20	90.2	92.2
0.25 / 0.25	100	20	binary	30	**92.6**	91.4
0.25 / 0.25	100	20	raw	30	83.8	86.0
0.5 / 0.25	25	20	binary	30	91.4	93.2
0.5 / 0.25	100	20	binary	30	89.6	91.2
0.5 / 0.5	25	5	binary	30	90.6	93.4
0.5 / 0.5	25	10	binary	30	90.8	91.8
0.5 / 0.5	25	20	binary	10	87.6	92.2
0.5 / 0.5	25	20	binary	20	90.6	92.6
0.5 / 0.5	25	20	binary	30	90.4	**94.0**
0.5 / 0.5	25	20	raw	30	82.8	83.8
0.5 / 0.5	50	20	binary	30	91.4	92.4
0.5 / 0.5	100	20	binary	30	91.4	92.8

because some musical sentences, usually at the beginning or end of the songs, are strings with the same term repeated all over, which increases term-frequency scores. Moreover, these terms usually do not appear anywhere else in the song, thus, increasing the inverse document frequency score (when also taken into consideration). This causes LSA to choose these unwanted sentences (usually noise or silence) because they will have a high score on a certain latent topic. Binary weighting alleviates these problems because we only check for the presence of terms in sentences, opposed to considering their frequency and inverse document frequency. The best accuracy obtained was 94% (row 15 of Table 6.4), an increase of 1.8%.

These results and analysis clearly show that LSA performs better when using binary weighting. Just like LexRank and the sections, LSA had a bigger impact on dataset 2.

Table 6.5 presents some results for MMR. A total of 108 summarization parameter combinations was tested. The MFCC vector size is also fixed at 12. The results presented include the best combinations for each dataset and variations of those combinations in all possible directions (parameters), one at a time.

MMR improved classification accuracy for only 1 tested combination, on dataset 1. The best accuracy obtained was 92.4% (row 16 of Table 6.5), an increase of 0.6%.

On dataset 2, MMR improved accuracy for 24 out of 108 combinations. The best accuracy obtained was 93.8%. (row 8 of Table 6.5), corresponding to an increase of 1.6%.

Table 6.5: Binary Classification MMR Results

Frame/Hop Size (s)	Vocabulary Size	Sentence Size	Weighting	λ	Usage (s)	Accuracy 1 (%)	Accuracy 2 (%)
0.25 / 0.25	25	20	dampTF	0.5	20	89.6	91.0
0.25 / 0.25	50	20	dampTF	0.5	20	89.0	90.8
0.25 / 0.25	100	5	dampTF	0.5	20	89.6	90.2
0.25 / 0.25	100	5	dampTF	0.7	20	91.0	91.0
0.25 / 0.25	100	10	dampTF	0.5	20	90.4	92.6
0.25 / 0.25	100	20	binary	0.5	20	89.4	90.4
0.25 / 0.25	100	20	dampTF	0.5	10	83.6	89.2
0.25 / 0.25	100	20	dampTF	0.5	20	90.6	**93.8**
0.25 / 0.25	100	20	dampTF	0.5	30	89.0	89.2
0.25 / 0.25	100	20	dampTF	0.7	20	90.0	93.6
0.5 / 0.25	25	5	dampTF	0.7	20	90.4	92.2
0.5 / 0.25	50	5	dampTF	0.7	20	90.6	91.4
0.5 / 0.25	100	5	binary	0.7	20	89.8	91.4
0.5 / 0.25	100	5	dampTF	0.5	20	91.8	92.8
0.5 / 0.25	100	5	dampTF	0.7	10	89.6	92.4
0.5 / 0.25	100	5	dampTF	0.7	20	**92.4**	92.6
0.5 / 0.25	100	5	dampTF	0.7	30	90.2	89.8
0.5 / 0.25	100	10	dampTF	0.7	20	89.2	93.4
0.5 / 0.25	100	20	dampTF	0.5	20	88.2	91.0
0.5 / 0.25	100	20	dampTF	0.7	20	88.0	90.8
0.5 / 0.5	100	5	dampTF	0.7	20	89.0	91.6
0.5 / 0.5	100	20	dampTF	0.5	20	89.0	92.4

Although summarization improved classification accuracy on this binary classifier, it did so by a small margin. This may be caused by the features extracted during the classification stage of the experiment. Those features include rhythmic information, which is based on an onset detection initial processing and statistics on that information, including periodicity through mean distance between peaks and others mentioned in Section 5.3.3. The sentence segmentation stage does not take into account any type rhythm information (e.g. tempo), segmenting the song into fixed-size frames and then into fixed-size sentences of an integer number of frames. Therefore some rhythm information is destroyed because these algorithms pick sentences which are not contiguous in the original song. The multiclass experiments do not use rhythmic features and revealed a greater summarization impact on classification as shown in Section 6.2.

Note that, in these experiments, every time the summaries were better than the sections, they were also better than using the full songs. This suggests that using full songs is worse because not all information in there is distinctive enough across different genres. Some of that information is actually bad for distinguishing between classes, thus, motivating the use of

summarization.

6.2 Multiclass Classification Results

This experiment involved 2 different feature sets (explained in Sections 5.3.2 and 5.3.4) during both summarization and classification steps of the experiment. Those sets are the MFCC set (only consisting of MFCCs) and the Timbral set (MFCCs concatenated with more timbral features). To distinguish between these sets, in summarization, the notation used is N when using the MFCC set and N+t when using the Timbral set, where N is the number of MFCCs. There are also 2 Accuracy columns: Accuracy 1 and 2, corresponding to classifying with the MFCC set and with the Timbral set, respectively.

Table 6.6: 5-Way Classification Baselines

Section	Accuracy 1 (%)	Accuracy 2 (%)
beginning	66.48	77.92
middle	**70.88**	**82.64**
end	61.20	77.36
full	78.24	91.12

Table 6.6 presents the baseline results. These results are obtained when classification is done using the usual beginning, middle and end segments of the songs (with 30 seconds duration) for the 5-class dataset. Just like the binary experiment, the middle section result will be used for comparison since it was the best on this dataset. Full songs were also tested for classification. As can be seen, the need for fast processing (through the use of 30 seconds segments) hinders the classification task by 7.36% in the first scenario and 8.48% in the second scenario (when comparing to the best 30 seconds section - the middle). This motivates the development of better ways of picking 30 seconds segments. These values show that this dataset is more difficult to correctly classify than both previous binary datasets. It is also clear that the Timbral set (Accuracy 2) is better for classifying than using only MFCCs (Accuracy 1).

Table 6.7 presents Average Similarity results. A total of 36 parameter combinations was tested. The results presented include the best combinations for each classification feature set and variations of those in all possible directions (parameters), one at a time.

When using the MFCC set for classification (Accuracy 1), 18 out of 36 combinations improved classification performance. Every combination using the Timbral set (for summarization) failed to improve accuracy while all others (using MFCC set) improved accuracy. This clearly dictates that using extra timbral features, besides the MFCCs themselves, hinders the summarization process. The best accuracy obtained this way was 76.08% (row 6 of Table 6.7), an increase of 5.2% and only 2.16% away from reaching the performance obtained by using full

Table 6.7: 5-Way Classification Average Similarity Results

# MFCC	Frame Size (s)	Hop Size (s)	Accuracy 1 (%)	Accuracy 2 (%)
12	0.25	0.125	74.00	83.84
12	0.25	0.25	74.72	84.16
12	0.5	0.25	74.40	84.96
12	0.5	0.5	74.32	85.28
12	1.0	0.5	74.72	84.64
12	1.0	1.0	**76.08**	85.20
20	0.25	0.125	74.00	83.92
20	0.25	0.25	74.00	84.48
20	0.5	0.25	74.64	85.20
20	0.5	0.5	74.64	84.88
20	1.0	0.5	75.36	**85.28**
20	1.0	1.0	75.20	84.96
24	1.0	0.5	75.36	85.04
24	1.0	1.0	75.28	84.72
12+t	1.0	0.5	66.80	77.04
12+t	1.0	1.0	65.12	77.12
20+t	1.0	0.5	66.80	77.12
20+t	1.0	1.0	65.12	76.96
24+t	1.0	0.5	66.80	77.04
24+t	1.0	1.0	65.12	77.12

songs.

Classifying with the Timbral set (Accuracy 2) led to very similar results: 18 out of 36 combinations improved classification performance. Those 18 were the same as for when classifying using the MFCC set: every combination using the Timbral set as summarization features, failed to improve accuracy. The best accuracy obtained was 85.28% (row 11 of Table 6.7), an increase of 2.64% and 5.84% away from using full songs.

Both classification scenarios for Average Similarity revealed that using only MFCCs, during the summarization process, is much better than using the Timbral set. Table 6.7 shows the difference of using these sets.

Table 6.8 presents LexRank results. A total of 72 parameter combinations was tested. The frame and hop sizes were fixed: 0.5 seconds frames with no overlap. The results presented include the best combinations for each classification feature set and variations of those in all possible directions (parameters), one at a time.

Classifying with the MFCC set led to a total of 68 parameter combinations improving classification performance. It is worth mentioning that the only 4 parameter combinations not improving accuracy use the Timbral set as features. The best accuracy obtained was 79.6% (row 7 of Table 6.8), corresponding to an increase of 8.72% which also means an increase of

Table 6.8: 5-Way Classification LexRank Results

# MFCC	Vocabulary Size	Sentence Size	Weighting	Accuracy 1 (%)	Accuracy 2 (%)
12	25	5	binary	79.20	89.20
12	25	5	dampTF	78.16	**89.68**
12	25	10	dampTF	77.20	89.68
12	25	20	dampTF	76.56	87.36
12	50	5	dampTF	78.56	88.96
12	100	5	dampTF	78.16	87.60
20	25	5	binary	**79.60**	88.88
20	25	5	dampTF	79.44	88.8
20	25	10	binary	76.64	88.32
20	25	20	binary	77.44	87.60
20	50	5	binary	78.80	88.48
20	100	5	binary	78.00	87.92
12+t	25	5	binary	75.52	86.40
12+t	25	5	dampTF	74.04	86.16
20+t	25	5	binary	75.60	86.32
20+t	25	5	dampTF	74.08	86.32

1.36% against using full songs.

When using the Timbral set for classification, LexRank improved classification accuracy for all parameter combinations. The best accuracy obtained was 89.68% (row 2 of Table 6.8), an increase of 7.04% and 1.44% away from using full songs.

Again, in both classification scenarios, the Timbral set was outperformed by the MFCC set (for summarization) which consistently yielded better classification results. Table 6.8 presents some results illustrating that difference.

Table 6.9 presents LSA results. A total of 36 parameter combinations was tested. The frame and hop sizes were fixed: 0.5 seconds frames with no overlap. The weighting was also fixed, since binary weighting is the only weighting scheme that works well with LSA (as explained in Section 6.1). The results presented include the best combinations for each classification feature set and variations of those in all possible directions (parameters), one at a time.

All combinations tested improved classification accuracy, in both classification scenarios (MFCC and Timbral sets as classification features). The best accuracy obtained, when classifying using the MFCC set, was 78.56% (row 8 of Table 6.9), an increase of 7.78%, corresponding to a 0.32% increase against using full songs. The best accuracy obtained, using the Timbral set for classification, was 89.84% (row 4 of Table 6.9) corresponding to an increase of 7.2%, only 1.28% away from the performance obtained using full songs.

Table 6.9 also shows the MFCC set outperforming the Timbral set in summarization.

Table 6.9: 5-Way Classification LSA Results

# MFCC	Vocabulary Size	Sentence Size	Accuracy 1 (%)	Accuracy 2 (%)
12	25	5	78.40	89.52
12	25	10	78.32	89.04
12	50	5	77.76	88.00
12	100	5	78.32	**89.84**
12	100	10	76.64	88.56
12	100	20	78.00	87.20
20	25	5	77.68	88.72
20	25	10	**78.56**	89.60
20	25	20	77.44	87.86
20	50	10	76.88	88.08
20	100	5	78.32	89.20
20	100	10	77.60	87.68
12+t	25	10	76.96	88.40
12+t	100	5	73.04	85.52
20+t	25	10	76.80	88.32
20+t	100	5	72.88	85.84

Table 6.10 presents results for MMR. 72 parameter combinations were tested. The frame and hop sizes were fixed: 0.5 seconds frames with no overlap. The weighting was also fixed as "dampened" TF-IDF. The results presented include the best combinations for each classification feature set and variations of those in all possible directions (parameters), one at a time.

Classification with the MFCC set was better than the baseline for 51 out of 72 combinations. Interestingly, only 2 of the 21 "bad" combinations rely on the Timbral set for summarization. This, along with the fact that the best results come from Timbral set combinations, suggests that the Timbral set is better for summarization for MMR. The best accuracy was 76.64% (row 13 of Table 6.10), which corresponds to an increase of 5.76% and to a 1.6% difference to using full songs.

When using the Timbral set for classification, all parameter combinations tested improved classification performance. The best accuracy obtained in this scenario was 89.76% (row 6 of Table 6.10), an increase of 7.12%, only 1.36% away from reaching full songs performance.

MMR results consistently show that, contrary to all other algorithms tested here, the Timbral set works better than the MFCC set during summarization. This might be due to the fact that the Timbral set is also used in one of the two classification scenarios tested here. However, even when using only MFCCs for classification, summarization with Timbral set works better (as can be seen by the results in Table 6.10).

Table 6.10: 5-Way Classification MMR Results

# MFCC	Vocabulary Size	Sentence Size	λ	Accuracy 1 (%)	Accuracy 2 (%)
12	25	5	0.7	72.40	87.68
12	25	10	0.7	70.72	84.16
12	25	5	0.7	71.68	86.32
12	25	10	0.7	69.28	84.64
12+t	25	5	0.5	76.24	88.88
12+t	25	5	0.7	74.32	**89.76**
12+t	25	10	0.7	76.48	89.36
12+t	25	20	0.7	74.48	86.96
12+t	50	5	0.7	75.92	87.76
12+t	100	5	0.7	72.56	86.72
20+t	25	5	0.7	74.16	89.76
20+t	25	10	0.5	74.80	88.72
20+t	25	10	0.7	**76.64**	89.28
20+t	25	20	0.7	74.56	86.88
20+t	50	10	0.7	74.72	87.68
20+t	100	10	0.7	72.16	86.16

6.3 Discussion

Although it is always possible to do a grid search on parameters to maximize the summarizers' performance (measured by classification accuracy), in most cases, those parameter values will not be the best for every setting. This might be due to the fact that the algorithms themselves, namely, generic summarization algorithms, are able to extract the most salient parts of the song, independently of parameter values such as sentence or vocabulary size. However, for some cases, conclusions could be arrived at: binary weighting clearly works best for LSA when applied to music this way in any of the experiments done; all algorithms work best when using only MFCCs during summarization (as shown in the multiclass experiments results), instead of a more complex feature set, except for MMR which exhibits a symmetric behaviour to this.

Despite parameterization details, the results of these experiments indicate that summarizing music with these algorithms improves classification performance against using random clips of the same duration, especially in the multiclass experiment. These experiments also indicate that using these summaries is almost as good as using the full songs themselves, sometimes even better. The fact that the tested algorithms look for relevance and/or diversity in music when producing the summary, along with the results of the experiments presented here, indicate that the summaries do contain more relevant and less redundant information than a random clip, allowing the classifier to train and predict better. Since these summaries sometimes are even better than using the full song (particularly on the binary classification task, but also on multiclass), this means that some information contained in the original signal, which

is not good for the classification task, is actually discarded in the process of extracting the summary. This means that the usual trade-off between efficiency (processing time) and classification performance (accuracy) is less influential when using summaries, since classification performance is almost the same as using the full songs by still processing only 30 seconds of it. An argument could be made, saying that summarizing music also takes processing and time, however, the idea is that summarization happens only once, during dataset construction, and then the dataset (of summaries) can be used for many tasks. Actually, other MIR tasks that rely on processing a portion of the whole audio signal may also benefit from this type of summarization.

This type of summarization may also be used for the distribution of music datasets between the MIR community. Nowadays, copyright issues force many datasets to be shared through the dataset's features instead of the audio signal itself, even if the features are extracted from 30 seconds audio clips. This can be a problem when someone wants the dataset but not those types of features or simply when someone wants to compare different features on the same dataset. Generic summarization may avoid those issues by tweaking some parameters (like sentence or frame size) so that the summaries are composed of very small sentences. This will yield summaries that are not suited for human consumption. In fact, people would not even consider the summary a piece of music because it's not coherent nor clear. They may realize that it is a concatenation of parts of a song but they do not enjoy listening to it. This fact may be decisive when considering copyright infringement.

6.4 Summary

This section presented the results of the experiments performed with binary and multi-class classification along with an analysis of them and ended with an overall discussion of the benefits of using summaries for classification tasks and dissemination of MIR datasets.

7 Conclusions

This thesis reviewed summarization algorithms, both generic and music-specific, in order to apply them to music datasets and evaluate that summarization's influence on the process of classifying those datasets. Summarization algorithms, especially generic algorithms, positively influence music classification when extrinsically measuring summarization, i.e., when measuring classification accuracy. These summaries may actually also benefit other MIR tasks besides classification. Furthermore, the resulting summaries are not suited for human listening, making this type of summarization a good candidate solution for distributing datasets for MIR research by avoiding copyright issues. Future work includes testing more summarization algorithms, feature types, classifiers and parameter values. More experimenting should also be done to determine if the vocabulary creation step may be improved by using Gaussian Mixture Models, or any other different clustering method, to model the MFCCs distribution more "naturally". Beat detection can also be used to potentially improve the initial frame segmentation step.

Bibliography

Antunes, P. G., D. M. de Matos, R. Ribeiro, and I. Trancoso (2014). Automatic Fado Music Classification. *CoRR abs/1406.4447*.

Bartsch, M. A. and G. H. Wakefield (2005). Audio Thumbnailing of Popular Music using Chroma-based Representations. *IEEE Transactions on Multimedia*, 96–104.

Bergstra, J., N. Casagrande, and D. Eck (2005). Two Algorithms for Timbre- and Rhythm-based Multi-resolution Audio Classification. In *Submission to Audio Classification (Train/Test) Tasks of MIREX 2005*.

Brin, S. and L. Page (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proc. of the Seventh International Conference on World Wide Web*, pp. 107–117.

Carbonell, J. and J. Goldstein (1998). The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proc. of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 335–336.

Chai, W. (2006). Semantic Segmentation and Summarization of Music: Methods Based on Tonality and Recurrent Structure. *IEEE Signal Processing Magazine*, 124–132.

Chang, C.-C. and C.-J. Lin (2011). LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 27:1–27:27.

Chu, S. and B. Logan (2000). Music Summary using Key Phrases. Technical report, Hewlett-Packard Cambridge Research Laboratory.

Cooper, M. and J. Foote (2002). Automatic Music Summarization via Similarity Analysis. In *Proc. of the 3rd International Society for Music Information Retrieval Conference*, pp. 81–85.

Cooper, M. and J. Foote (2003). Summarizing Popular Music via Structural Similarity Analysis. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 127–130.

Curtin, R. R., J. R. Cline, N. P. Slagle, W. B. March, P. Ram, N. A. Mehta, and A. G. Gray (2013). MLPACK: A Scalable C++ Machine Learning Library. *Journal of Machine Learning Research*, 801–805.

de Leon, F. and K. Martinez (2013). Using Timbre Models for Audio Classification. In *Submission to Audio Classification (Train/Test) Tasks of MIREX 2013*.

Erkan, G. and D. R. Radev (2004). LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 457–479.

Eyben, F., F. Weninger, F. Gross, and B. Schuller (2013). Recent Developments in openSMILE, the Munich Open-source Multimedia Feature Extractor. In *Proc. of the 21st ACM International Conference on Multimedia*, pp. 835–838.

Glaczynski, J. and E. Lukasik (2011). Automatic Music Summarization: A "Thumbnail" Approach. *Archives of Acoustics*, 297–309.

Gong, Y. and X. Liu (2001). Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In *Proc. of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 19–25.

Mandel, M. and D. Ellis (2005). Song-level Features and SVMs for Music Classification. In *Submission to Audio Classification (Train/Test) Tasks of MIREX 2005*.

Mihalcea, R. and P. Tarau (2004). TextRank: Bringing Order into Texts. In *Proc. of the 9th Conference on Empirical Methods in Natural Language Processing*, pp. 404–411.

Murray, G., S. Renals, and J. Carletta (2005). Extractive Summarization of Meeting Recordings. In *Proc. of the 9th European Conference on Speech Communication and Technology*, pp. 593–596.

Peeters, G., A. La Burthe, and X. Rodet (2002). Toward Automatic Music Audio Summary Generation from Signal Analysis. In *Proc. of the 3rd International Society for Music Information Retrieval Conference*, pp. 94–100.

Peeters, G. and X. Rodet (2003). Signal-based Music Structure Discovery for Music Audio Summary Generation. In *Proc. of the 29th International Computer Music Conference*, pp. 15–22.

Radev, D. R., H. Jing, and M. Budzikowska (2000). Centroid-based Summarization of Multiple Documents: Sentence Extraction, Utility-based Evaluation, and User Studies. In *Proc. of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, pp. 21–30.

Sanderson, C. (2010). Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments. Technical report, NICTA.

Steinberger, J. and K. Jezek (2004). Using Latent Semantic Analysis in Text Summarization and Summary Evaluation. In *Proc. of ISIM*, pp. 93–100.

Tzanetakis, G. and P. Cook (1999). MARSYAS: A Framework for Audio Analysis. *Organised Sound*, 169–175.

Wu, M.-J. (2013). MIREX 2013 Submissions for Train/Test Tasks (Draft). In *Submission to Audio Classification (Train/Test) Tasks of MIREX 2013*.

Zechner, K. and A. Waibel (2000). Minimizing Word Error Rate in Textual Summaries of Spoken Language. In *Proc. of the 1st North American Chapter of the Association for Computational Linguistics Conference*, pp. 186–193.

Zhu, X., A. B. Goldberg, J. V. Gael, and D. Andrzejewski (2007). Improving Diversity in Ranking using Absorbing Random Walks. In *Proc. of the 5th North American Chapter of the Association for Computational Linguistics - Human Language Technologies Conference*, pp. 97–104.