Influence of Summarization on Music Classification Tasks

Francisco Afonso Raposo

Instituto Superior Técnico

***Abstract*—Music classification is the task of predicting classes of songs (e.g. artist/genre identification). Current music classifiers only process segments of songs (for faster processing) and are evaluated to get an accuracy score because they are not infallible, sometimes wrongly classifying songs. Music summarization is the task of summarizing music, i.e., selecting the most important parts of the song and outputting only those in the audio summary. Current music summarization methods focus on producing summaries to be listened and enjoyed by people. Other (generic) summarization methods have been successfully applied in text and speech summarization. This paper evaluates some of these summarization algorithms' impact on music classification. It is worth considering that processing the most important part(s) (i.e. summaries) may improve the classifier's accuracy. Results are presented showing that Average Similarity, LexRank, Latent Semantic Analysis (LSA), and Maximal Marginal Relevance (MMR) have a positive impact on music classification.**

I. Introduction

Music Information Retrieval (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 others, are annually addressed by the MIR community in conferences such as that from theInternational 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 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 and mood classification whose goal is to output a label/class according to the song's composer 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 extra 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 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 paper reviews summarization algorithms, both music-specific and generic, in order to summarize music for automatic 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.

Section II reviews related work on summarization. Specifically, the following algorithms are reviewed: Average Similarity in Section II-A, LexRank in Section II-B, LSA in Section II-C and MMR in Section II-D. Section III describes the details of the experiments, including the classifiers used to extrinsically evaluate summarization. Section IV reports and discusses the results of the experiments and Section V concludes this paper with some remarks and future work.

II. Summarization

Several methods for summarization have been successfully applied in fields such as music, text, and speech summarization. Text and speech summarization methods are, in fact, generic summarization algorithms. These algorithms will eventually score/rank each sentence and then pick sentences (from any part of the signal) to include in the summary, until the desired length is reached. Since music summarization algorithms were developed for human listening of the resulting artifacts, they take into account human consumption related requirements such as coherence and clarity. None wants to listen to a music summary full of harsh discontinuities and other issues that arise from taking small segments from different parts of the song. Because of this, these summaries are usually continuous segments from the original song.

The Graph Random-walk with Absorbing StateS that HOP among PEaks for Ranking (GRASSHOPPER) [1] has been successfully applied in text summarization and also in social network analysis. Its focus is on improving diversity while ranking sentences. LexRank [2] and TextRank [3], based on Google's PageRank, are centrality-based methods that score/rank sentences according to the similarity between every sentence and are successfully applied in text summarization. LSA [4] is a method that builds a terms by sentences matrix representation of the text in order to later decompose it 3 other matrices encoding topics, their relevance, and each sentence relevance within each topic. This is achieved through Singular Value Decomposition (SVD). MMR [5] is a method that iteratively selects sentences according to a linear combination of: each sentence's similarity to a "query" sentence and each sentence's similarity to every previously selected sentence (i.e. each sentence's non-redundancy). This method is query-based allowing it to produce topic-oriented summaries and has been successfully applied in speech summarization.

Music summarization methods have also been successfully applied in the past, focusing on human consumption of the output summary. Consequently, literature on this topic only evaluates music summarization from this point of view, i.e., it measures how easily people can identify the original song by hearing its summary or how well the original song is summarized (according to the subjective terms of clarity, coherence etc.). This means that no evaluation has been performed, on these algorithms, in terms of automatic consumption of those summaries. These methods segment songs according to higher level semantic information (i.e. segmentation of chorus, bridge, etc.) and choose which segments (or transitions between segments) to include in the summary, while taking into account human consumption related requirements. These stages are not necessarily done separately.

Some music-specific summarization methods start by structurally segmenting songs and then selecting meaningful segments (e.g. chorus, bridge) to include in the summary. In [6], segmentation is achieved by using a Hidden Markov Model (HMM) to detect key changes between frames and Dynamic Time Warping (DTW) to detect repeating structure. In [7], a Gaussian-tempered "checkerboard" kernel is correlated along the main diagonal of the song's self-similarity matrix, outputting segment boundaries. Then, a segment-indexed similarity matrix is built, containing the similarity between detected segments. SVD is applied to that matrix to find its rank-K approximation. Segments are, then, clustered to output the song's structure. In [8], [9], songs are segmented in 3 stages. First, a similarity matrix is built and analyzed for fast changes, outputting segment boundaries. These segments are clustered to output the "middle states". Finally, an HMM is applied to these states, producing the final segmentation. These algorithms then follow various strategies to select the appropriate segments. [10] groups (based on the Kullback-Leibler (KL) divergence) and labels similar segments of the song. The summary is the longest sequence of segments belonging to the same cluster. In [11], [12], Average Similarity is used to extract a thumbnail L seconds long that is the most similar to the whole piece. Another method for this task, Maximum Filtered Correlation [13], starts by building a similarity matrix and then a filtered time-lag matrix, whose maximum value marks the start of the summary. [14] classifies music as pure or vocal, in order to perform type-specific feature extraction. The summary, created from 3s to 5s subsummaries (built from frame clusters), takes into account musicological and psychological aspects, since it differentiates between types of music based on feature selection and specific duration. This promotes human enjoyment when listening to the summary.

The next sections review each of the 4 selected algorithms for evaluation: Average Similarity, LexRank, LSA and MMR.

A. Average Similarity

This method is also referred to as a "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 in [11] and later used in [12]. 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.

In [11] 45 Mel Frequency Cepstral Coefficient (MFCC)s are computed 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.

In [12], the first 13 MFCCs and the spectral centre of gravity (sound "brightness") are used. 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: $S(i,j) = s(v_i, v_j)$.

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:

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

Where N is the number of frames in the entire piece. The index i of the best summary starting frame is the one that maximizes $Q_L(i)$.

The evaluations of this method in the literature are human evaluations taking into account whether the generated summaries include the most memorable part(s) (also called *hooks*) of the song [11]. Other subjective evaluations are averages of scores given by test subjects regarding specific qualities of the summary such as Clarity, Conciseness and Coherence [12].

B. LexRank

LexRank [2] is a centrality-based method which relies on the similarity between every sentence to determine each sentence centrality, thus, each sentence relevance. This method is based on Google's PageRank [15] algorithm for ranking web pages and was developed in the context of text summarization. The output is a list of ranked sentences from which it is possible to extract the highest ranked ones to produce a summary. First, all sentences, normally represented as Term Frequency - Inverse Document Frequency (TF-IDF) scores vectors, are compared to each other using a similarity measure (usually the cosine distance). 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. 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):

$$S(V_i) = \frac{(1-d)}{N} + d \times \sum_{V_j \in a[V_i]} \frac{Sim(V_i, V_j)}{\sum_{V_k \in a[V_j]} Sim(V_j, V_k)} S(V_j) \qquad (2)$$

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

$$S(V_i) = \frac{(1-d)}{N} + d \times \sum_{V_j \in a[V_i]} \frac{S(V_j)}{D(V_j)} \qquad (3)$$

Where $D(V_i)$ is the degree of vertex i (i.e. number of edges/connected vertices). A summary is built by taking the highest ranked sentences until the 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.

C. Latent Semantic Analysis

LSA is a method based on a mathematical technique called SVD that was first used for generic text summarization in [16]. 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:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{T1} & \cdots & a_{TN} \end{bmatrix} \qquad (4)$$

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: $a_{ij} = L_{ij} G_i$.

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): $A = U \Sigma V^T$.

After this decomposition, the singular values are sorted in 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 can be extracted by selecting sentences corresponding to the indices of the highest values for each (most relevant) right singular vector.

In [4], two limitations of this approach are discussed: the fact that K is equal to the number of sentences in the summary, which as it increases, tends to include less significant sentences; and that sentences with high values in several 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 K^{th} singular value does not fall under half of the highest singular value.

$$score(j) = \sqrt{\sum_{i=1}^{k} v_{ij}^2 \sigma_i^2} \qquad (5)$$

D. Maximal Marginal Relevance

This method, introduced in [5], selects sentences from the 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 in [17], [18]. 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 [18]) as the query.

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

$$\arg\max_{S_i}\left[\lambda\left(Sim_1\left(S_i,Q\right)\right)-\left(1-\lambda\right)\max_{S_j}Sim_2\left(S_i,S_j\right)\right] \quad (6)$$

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.

III. Experiments

This section 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. The 2 classifiers are: 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. Both classifiers are Support Vector Machine (SVM)s [19]. 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.

A. Binary Datasets

Dataset 1 consists of 500 songs: 250 Fado songs and 250 songs from 10 balanced genres (Classical, Rock, Jazz, Pop, Folk, Medieval, Blues, Country, Tribal, Electronic) [20]. 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 have a very similar timbre to Fado (making it more difficult to correctly classify them), and some pieces from the 10 genres mentioned. Both datasets are encoded in mono, 16-bit, 22050 Hz Microsoft WAV files.

B. 5-class Genre Dataset

This dataset consists of 1250 songs from 5 different genres: Bass Music, Fado, Hip Hop, Indie Rock and Trance. Each class is represented by 250 songs from several artists. 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.

C. 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 [21] was used for feature extraction (for summarization), Armadillo [22] was used for matrix operations, and Marsyas [23] 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 [24] 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 [22] 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 5, and sentences are picked according to the scores until the desired summary length is reached.

D. 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 λ values (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 these same weighting types. 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 [7], [10]–[12]. 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.

E. 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 tested with 6 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: 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 III-G. 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.

F. Binary Classification Features

The features 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 [20]. 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 [20].

G. 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 [25]. 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 [21].

IV. Results

This section 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.

A. Binary Classification Results

This experiment involved 2 different datasets (as explained in Section III-A) 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.

The baseline 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 III-A. 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 I
BINARY CLASSIFICATION AVERAGE SIMILARITY RESULTS

# MFCC	Frame Size (s)	Hop Size (s)	Acc. 1 (%)	Acc. 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 I presents some results for Average Similarity. A total of 12 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 I) and 89.8% (row 10 of Table I), 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 II presents results for LexRank. 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 II) which corresponds to a 0.8% increase.

TABLE II
BINARY CLASSIFICATION LEXRANK RESULTS

Framing (s)	Voc. Size	Sent. Size	Weighting	Usage (s)	Acc. 1 (%)	Acc. 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 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 II), 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 III
BINARY CLASSIFICATION LSA RESULTS

Framing (s)	Voc. Size	Sent. Size	Weighting	Usage (s)	Acc. 1 (%)	Acc. 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

Table III 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, 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 III), corresponding to a 0.8% increase.

For dataset 2, 24 out of 54 combinations of parameters yielded an improvement in classification accuracy. Interestingly, all those combinations used 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 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 latent topic. Binary weighting alleviates these problems because it only checks 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 III), 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 IV
BINARY CLASSIFICATION MMR RESULTS

Framing (s)	Voc. Size	Sent. Size	Weight.	λ	Us. (s)	Acc. 1 (%)	Acc. 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

Table IV presents results for MMR. A total of 108 summarization parameter combinations was tested. The MFCC vector size is fixed at 12. The results presented include the best combinations for each dataset and variations of those 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 IV), 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 IV), corresponding to an increase of 1.6%.

Although summarization improved classification accuracy on this classifier, it did so by a small margin. This may be caused by the features extracted during the classification stage. Those features include rhythmic information, which is based on statistics of an initial onset detection processing, including periodicity through mean distance between peaks and others mentioned in Section III-F. The sentence segmentation stage does not take into account any type of 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 IV-B.

Note that, 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.

B. Multiclass Classification Results

This experiment involved 2 feature sets (explained in Sections III-E and III-G) 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.

The baseline 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 results (70.88% and 82.64% for the MFCC and Timbral sets, respectively) will be used for comparison since they were the best. Full songs were also tested yielding accuracies of 78.24% and 91.12% for the MFCC and Timbral sets, respectively. 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 V presents Average Similarity results. 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 accuracy. Every combination using the Timbral set (for summarization) failed to

TABLE V
5-WAY CLASSIFICATION AVERAGE SIMILARITY RESULTS

# MFCC	Frame Size (s)	Hop Size (s)	Acc. 1 (%)	Acc. 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

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 V), an increase of 5.2% and only 2.16% away from reaching the performance obtained by using full songs.

Classifying with the Timbral set (Accuracy 2) led to 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 V), 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 VI
5-WAY CLASSIFICATION LEXRANK RESULTS

# MFCC	Voc. Size	Sent. Size	Weighting	Acc. 1 (%)	Acc. 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

Table VI presents LexRank results. 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, 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 VI), corresponding to an increase of 8.72% which also means an increase of 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 VI), 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 VII
5-WAY CLASSIFICATION LSA RESULTS

# MFCC	Voc. Size	Sent. Size	Acc. 1 (%)	Acc. 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 VII 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 IV-A). 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 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 VII), 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 VII) corresponding to an increase of 7.2%, only 1.28% away from the performance obtained using full songs. Table VII also shows the MFCC set outperforming the Timbral set in summarization.

Table VIII 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.

TABLE VIII
5-WAY CLASSIFICATION MMR RESULTS

# MFCC	Voc. Size	Sent. Size	λ	Acc. 1 (%)	Acc. 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

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 VIII), 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 VIII), 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 VIII).

C. 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 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. These summaries are not suited for human consumption. In fact, people would not even consider the summary a piece of music because it is 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.

V. CONCLUSIONS AND FUTURE WORK

This paper reviewed summarization algorithms, both generic and music-specific, in order to apply them to music datasets and evaluate that summarization's influence in the process of classifying those datasets. Summarization algorithms, especially generic algorithms, positively influence music classification when measuring classification accuracy. Future work includes testing more summarization algorithms, feature types, classifiers and parameter values. Experimenting should 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.

REFERENCES

[1] X. Zhu, A. B. Goldberg, J. V. Gael, and D. Andrzejewski, "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*, 2007, pp. 97–104.

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

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

[4] J. Steinberger and K. Jezek, "Using Latent Semantic Analysis in Text Summarization and Summary Evaluation," in *Proc. of ISIM*, 2004, pp. 93–100.

[5] J. Carbonell and J. Goldstein, "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*, 1998, pp. 335–336.

[6] W. Chai, "Semantic Segmentation and Summarization of Music: Methods Based on Tonality and Recurrent Structure," *IEEE Signal Processing Magazine*, pp. 124–132, 2006.

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

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

[9] G. Peeters and X. Rodet, "Signal-based Music Structure Discovery for Music Audio Summary Generation," in *Proc. of the 29th International Computer Music Conference*, 2003, pp. 15–22.

[10] S. Chu and B. Logan, "Music Summary using Key Phrases," Hewlett-Packard Cambridge Research Laboratory, Tech. Rep., 2000.

[11] M. Cooper and J. Foote, "Automatic Music Summarization via Similarity Analysis," in *Proc. of the 3rd International Society for Music Information Retrieval Conference*, 2002, pp. 81–85.

[12] J. Glaczynski and E. Lukasik, "Automatic Music Summarization: A "Thumbnail" Approach," *Archives of Acoustics*, pp. 297–309, 2011.

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

[14] C. X. Xu, N. C. Maddage, and X. S. Shao, "Automatic Music Classification and Summarization," *IEEE Transactions on Speech and Audio Processing*, pp. 441–450, 2005.

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

[16] Y. Gong and X. Liu, "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*, 2001, pp. 19–25.

[17] K. Zechner and A. Waibel, "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*, 2000, pp. 186–193.

[18] G. Murray, S. Renals, and J. Carletta, "Extractive Summarization of Meeting Recordings," in *Proc. of the 9th European Conference on Speech Communication and Technology*, 2005, pp. 593–596.

[19] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, pp. 27:1–27:27, 2011.

[20] P. G. Antunes, D. M. de Matos, R. Ribeiro, and I. Trancoso, "Automatic Fado Music Classification," *CoRR*, vol. abs/1406.4447, 2014.

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

[22] C. Sanderson, "Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments," NICTA, Tech. Rep., 2010.

[23] G. Tzanetakis and P. Cook, "MARSYAS: A Framework for Audio Analysis," *Organised Sound*, pp. 169–175, 1999.

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

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