MULTIMODAL SENTIMENT ANALYSIS ON SONGS USING ENSEMBLE CLASSIFIERS

BY

ESTEBAN GOMEZ-SAAVEDRA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Electrical and Computer Engineering
in the Undergraduate College of Engineering of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Minh N. Do

ABSTRACT

We consider the problem of performing sentiment analysis on songs by combining audio and lyrics in a large and varied dataset, using the Million Song Dataset for audio features and the MusicXMatch dataset for lyric information.

The algorithms presented on this thesis utilize ensemble classifiers as a method of fusing data vectors from different feature spaces. We find that multimodal classification outperforms using only audio or only lyrics. This thesis argues that utilizing signals from different spaces can account for inter-class inconsistencies and leverages class-specific performance. The experimental results show that multimodal classification not only improves overall classification, but is also more consistent across different classes.

Keywords: Music Information Retrieval; Sentiment Analysis; Multimodal Classification; Classification Algorithms; Multimodal Fusion

ACKNOWLEDGMENTS

I want to thank Prof. Minh N. Do, Ben Chidester, and the rest of Prof. Minh Do's research group for all the guidance and help they have provided through the development of this project.

TABLE OF CONTENTS

1

INTRODUCTION

In recent years, music-based services have been trying to make their applications more user-centered. One way to make sure that the user experience is foremost is to provide services that match the user's current emotional state. This can be implemented by understanding the emotional content of the songs and playlists that are being played.

Sentiment analysis is the portion of music informational retrieval (MIR) where an algorithm recognizes the main emotions that a song evokes. Emotions are subjective, so classifying individual songs into distinct groups is a challenging problem. Most human subjects agree in broad strokes on emotional classifications. However it is not uncommon to find songs where there is no consensus, leading to inconsistencies in class groupings. As a result, sentiment analysis is still an open problem that requires the investigation of alternative methods that employ different modalities to counteract class inconsistencies. Success in this approach would allow a more robust classification algorithm that is better suited for real-world applications.

The motivation behind the presented research is to find a method that accounts for these inter-class inconsistencies across a large dataset. During our initial investigation, it became apparent that certain unimodal classifiers perform well on specific sentiments and fail to accurately classify others. This research seeks to produce more consistent classification by combining features and classifiers of different modalities thus improving the reliability of the overall system.

This thesis will be focusing on the employment of support vector machines (SVM), Gaussian naive bayes and multinomial naive bayes classifiers to recognize emotional information. The audio features that will be employed are based on mel-frequency cepstral coefficients (MFCC), which are obtained from the response of the song's windowed spectrogram to a set of basis functions. A bag-of-words vector is used to represent the lyric portion of the song. The modal fusion will be tested through both feature fusion and classifier fusion.

The dataset used for this thesis is called the Million Song Dataset and was compiled by Labrosa [1]. This dataset contains a million different songs summarized by their pitch, loudness, and timbre. The songs are also accompanied by much metadata such as artist, release date, and tags. Sentiment classification was obtained from these tags. If a sentiment was used to describe a song, it is assumed that the song conveyed that sentiment. The lyric information was obtained from the MusicXMatch dataset [2], which provides lyric information unordered in a bag-of-words format. Since there was no way to obtain semantic information from an unordered bag-of-words representation, this research did not focus on the impact of semantics on sentiment classification.

This report starts with a brief overview of previous work performed on this topic, followed by a description of the methods and the algorithms developed, then a section detailing the actual experiments, and a section on results and analysis. It closes with a section on conclusions and suggestions for future work.

2

PREVIOUS WORK

There has been considerable amount of work done in the field of multimodal sentiment analysis. This Chapter will briefly cover a portion of the relevant research that was considered during the development of the presented methodologies. The relevant topics that were researched for this thesis were: audio sentiment analysis, text sentiment analysis, and multimodal classification.

Audio Sentiment Analysis

The study of the relationship between emotional content and audio signals is a very mature field. Researchers have expanded on the success found in the speech recognition community of using mel-frequency cepstral coefficients (MFCC) to explore their uses in music modeling [3]. MFCCs are currently a staple in audio processing and are commonly used in MIR applications such as genre classification [4], since they are a quantifiable method for comparing the timbral texture of songs. Timbre has been used with some success to classify the emotional content of songs [5]; however, class inconsistencies have proven to be a difficult challenge, causing substantial misclassification between edge cases. Timbre has also been used to generate songs that evoke particular emotions [6]. These vectors have been commonly classified using support vector machines (SVM) and naive Bayes classifiers.

Text Sentiment Analysis

Similarly, the study of the relationship between text and emotional content is quite developed, with applications ranging from predicting Yelp ratings based on the sentiment expressed in a given review [7] to extracting the emotional progression of major literary pieces [6]. There are many methods to represent and extract emotional information from texts. The Yelp experiment uses statistical word vectors to capture word semantics and emotions as a probability. Other researchers have represented textual information in a bag-of-features framework and used naive Bayes, SVMs and maximum entropy classifiers to recognize positive or negative valances [8].

Researchers have extended text processing methods to better capture emotional subtleties. For example, a word can have different emotional values depending on its context. Analyzing this information requires the creation of complex sentiment vectors that encode how meanings change based on semantics [9]. Similarly, researchers have improved classification accuracy by preprocessing text [10] and using the cleaned data to capture emotional subtleties, like the use of negation and modifiers to emotional words [11].

Although there is a great body of research on how to obtain rich sentiment vectors from text, the goal of this thesis is demonstrate the added advantage of a multimodal approach. For that end, the lyric vectors were kept simple to clearly underline the benefit of combining them with audio information. In addition, it is necessary to point out that obtaining a large enough dataset of lyrics is difficult due to legal restrictions. As a result, the features used will be the unordered representation of the lyrics in a bag-of-words vector provided by the MusicXMatch dataset [2].

Multimodal Classification

Multimodal classification is the task of using feature vectors from different modalities, for example text and audio, to reach a single classification. There are two main methods of combining the information from the features of both modalities: feature fusion and classifier fusion [12].

Feature fusion is the technique that takes signals from different feature spaces and joins them to train a single multimodal classifier. The standard fusion method is called "series fusion", which consists of concatenating the vectors together and training the classifier on the union of both spaces. Several alternatives have been suggested to maintain the same amount of expressibility in the fused vector while keeping the resulting vector space as small as possible. Instead of concatenating the vectors together, it is possible to join vectors in parallel [13] by making vectors from the linear combinations of a real-valued feature with another complex-valued feature. The benefit of the series fusion over the parallel method is that many diverse features can be fused together to obtain more robust data. As seen in the research by Liang et al. [14], genre classification was improved by joining five different vectors, all resulting from different preprocessing methods for text and audio vectors.

Classifier fusions train an array of unimodal classifiers and using some function to consolidate the predictions [15]. This method seamlessly fuses features from very different spaces. Caridakis et al. [16] combined facial expressions, body gestures, and speech by having a classifier voting system where the class with most votes and highest probability was chosen amongst all the decisions. The final decision-making process can be taken a step further by adding an additional classifier that learns from the decisions provided from the classifier array [17]. The algorithms presented in this research were largely based on this last approach.

Multimodal classification has been successful in improving the accuracy of classification [12] [18]. However, some of the previous work either ran the experiments on highly homogenous datasets, where all the music was in the same language, belonged to the same genre, or was carefully classified by a single subject, thus eliminating class inconsistencies. The goal of this research is to obtain improved classification and reliability for a varied dataset through ensemble classifiers.

3

METHOD

Emotions are highly nuanced since there are many experiences that do not fall neatly within a category. The difference between bittersweet and nostalgic, for example, is difficult to quantify. In a study of gestures across different cultures, Ekman et al. [19] concluded that there are six basic emotions, called the Ekman emotions, which are joy, sadness, anger, fear, disgust, and surprise. For this research we will focus on classifying songs based on the first three emotions (joy/happiness, sadness, and anger), mainly because there are not enough sample songs whose main emotions are the last three (fear, disgust, and surprise).

Features Used

The audio features used for the experiment are the timbre data in the Million Song Dataset (MSDS) developed by EchoNest. These features are zero-mean vectors of length 12, extracted by windowing the spectrogram in constant-width segments. Although the segment width varies from song to song, it is typically less than a second long, and it is determined so that the timbre and harmony are relatively uniform throughout.

The song is broken up into a series of non-overlapping windows $\mathcal{S}_i$, where i is the index of the segment. The images in Figure 3.1 are a visual representation of the basis functions g_j that were applied to each segment. An audio feature f is extracted from each segment and has the form of a vector of length 12, where each element of the vector is defined as:

$$f_j(x) = \sum_{p \in \mathcal{S}_i} g_j(p) \bullet x(p), \qquad j \in \{1, 2, ..., 12\}. \tag{3.1}$$

The lyric information was obtained from the musicXmatch database [2] which provides the songs in a bag-of-words format. This format consists of a vector of length 5000 representing a stemmed dictionary where the value at each element is the frequency of that particular word in the song. Data is distributed in this form so that it respects artists' rights over the ordering of the words while still being able to represent the content.

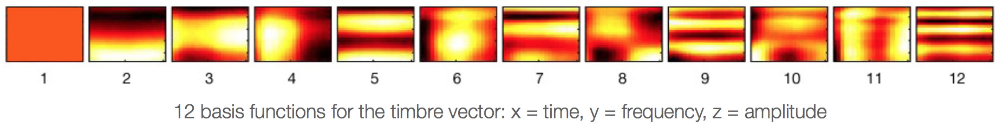

Figure 3.1: EchoNest's Basis Functions Used to Generate MFCC-like Vectors.[1]

Classifiers

As mentioned in Chapter 2, similar problems have been address with a certain amount of success using naive Bayes and SVMs. For the experiment, the classifiers selected were the Gaussian naive Bayes, multinomial naive Bayes, and kernel SVM with histogram intersection as the kernel.

Naive Bayes

The naive Bayes classifier is a statistical classifier that selects the most-likely class given a particular data point. The classifier builds a statistical model for each class that follows a given distribution. Under the assumption that each feature is statistically independent, it computes the conditional probability for the input vector given each model and returns the model with the highest posterior probability.

$$\hat{y} = \text{argmax}_y P(y)\Pi_{i=1}^{n} P(x_i|y) \tag{3.2}$$

The distinction between the Gaussian naive Bayes and the multinomial naive Bayes is the distribution that is assumed for the conditional probability $P(x_i|y)$. The Gaussian naive Bayes classifier, as the name suggests, assumes that data is normally distributed given the class and that the covariance matrices are diagonal. This model is a good starting point when little information is known about the data.

[1]`http://developer.echonest.com/docs/v4/_static/AnalyzeDocumentation.pdf`

The multinomial naive Bayes supposes that data follows a multinomial distribution, which is a generalization of the binomial distribution. The difference between binomial and multinomial is that in binomial the outcome of each of the k trials is either "yes" with a probability of p or "no" with a probability of $(1-p)$. Under the multinomial distribution, the outcome of each of the k trials can be one of n different classes such that $\sum_{i=1}^{n} p_i = 1$. Since an event cannot occur a negative number of times, the multinomial distributions is well suited for problems where the vectors are non-negative such as for histogram analysis.

Support Vector Machines

Support Vector Machines are classifiers that construct hyperplanes that maximize the margin to the labeled data points. The margin is defined to be the shortest distance from the hyperplane to any of the points. As a result, a linear SVM finds a function that linearly separates the data points into clusters for classification. Soft SVMs exist that tolerate data that is not linearly separable by allowing some misclassification.

An SVM can be non-linear if the data points are preprocessed non-linearly before creating the hyperplane. Non-Linear SVMs rely on functions called kernels that map data points into a higher dimensional space where the points can be separated by a single hyperplane.

For the experiments detailed on this paper, we used a non-linear SVM with a histogram intersection kernel. Histogram intersection is a method that finds the minimum intersection between two histograms [20], which means that it is generally a more accurate similarity metric than euclidean distances for histograms. The histogram intersection kernel is defined as:

$$k_{HI}(h_a, h_b) = \sum_{j=1}^{d} min(h_a(j), h_b(j)). \tag{3.3}$$

Equation 3.3 details the metric used to compare the intersection of two histograms. The distance of a particular pair of histograms is the sum of minimum value of each column of the histogram. By computing the distance between every pair of histograms using this metric, the SVM can easily group similar data points together.

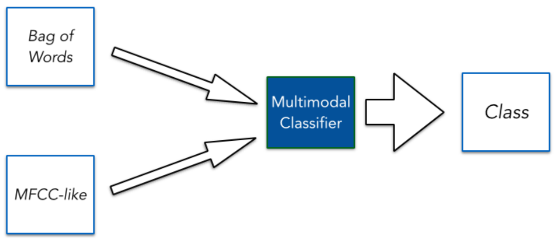

Figure 3.2: Series Fusion

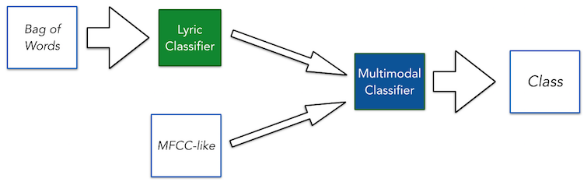

Figure 3.3: Lyrics Only Partial Ensemble

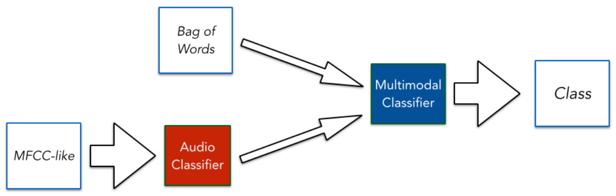

Figure 3.4: Audio Only Partial Ensemble

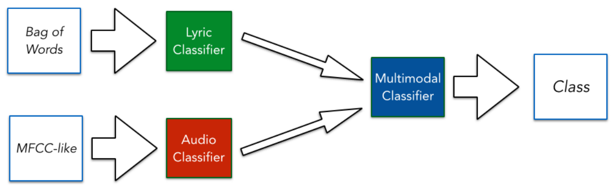

Figure 3.5: Full Ensemble

Fusion Methods

Using the features and the classifiers detailed above, four different fusion methods were used for classification: series fusion, two separate partial ensembles and full ensemble. These were defined by concatenating the feature vectors at different levels of the classification. Figure 3.2 shows two raw vectors used to train a multimodal classifier. Figure 3.3 and Figure 3.4 show one level of pre-classification. An initial classifier is used on either the lyric or the audio raw vectors, respectively, to create an intermediate vector. This resulting feature is joined with the remaining raw vector to train a multimodal classifier. Figure 3.5 shows two levels of pre-classification, where both raw vectors are classified and a multimodal classifier is used on the intermediate vectors.

It is important to note that out of the classifiers detailed above, only Gaussian naive Bayes is suited to classify negative-valued features. As a result not all the classifiers can be used for all the vectors. The multinomial naive Bayes classifier requires that audio segments be normalized to be non-negative to be fitted to the multinomial distribution. We introduce the following classifier sets:

$\mathcal{A} = \text{set of audio classifiers} = \{\text{Gaussian NB}, \text{Multinomial NB}\}$

$\mathcal{L}$ = set of lyric classifiers = $\{\text{Gaussian NB}, \text{Multinomial NB}, \text{SVM}\}$

$\mathcal{M}$ = set of multimodal classifiers = $\{\text{Gaussian NB}, \text{Multinomial NB}, \text{SVM}\}$.

The set $\mathcal{A}$ is used for audio features, the set $\mathcal{L}$ is used for lyric features, and the set $\mathcal{M}$ is used for multimodal features. In Figures 3.2-3.5, the squares labeled as 'Multimodal Classifier', 'Audio Classifier' and 'Lyric Classifier' represent a sampled classifier from the corresponding set.

Series Fusion

The series fusion method is the basic approach discussed in Chapter 2. This method concatenates each raw audio vector with the corresponding raw lyric vector. As a result each song has hundreds of segments, each of which is a vector of length 5012. This large dataset can be computationally prohibitive for certain classifiers. The classifier set for this structure is :

$$Series = \mathcal{M} \setminus \{SVM\}.$$

The reason why SVMs were not included in this test was because the runtime complexity of training a linear SVM is at worst $O(d^2n)$ where d is the number of dimensions and n is the number of samples [21]. With a hundred vectors per song where each vector has 5012 dimensions, it becomes impractical to run this test.

This fusion method involves taking a classifier from the set *Series* and training it on the dataset resulting from the concatenation of the audio vectors and bag of words.

Audio Partial Ensemble

The audio partial ensemble methods come from noting that the complexity of the series fusion method stemmed from the fact that the amount of vectors for each of the modes was lopsided. Each song has exactly one bag of words vector of length 5000, while having hundreds of audio segments of length 12. Audio partial ensemble explores classifying those hundreds of segments into emotions and representing them as a single classification histogram that can be combined with the bag of words.

This method initially classifies all the song's audio segments and builds a histogram for each song representing how many of these segments were classified for each emotion. As a result, the audio data goes from being represented by hundreds of vectors of length 12, to a single histogram of length 3 that provides some concrete information of the emotional content of the song.

The classifier set *Audio Partial Ensemble* is defined as all the combinations between the Multimodal classifier set and the Audio classifier set. The resulting set is as follows:

$$\textit{Audio Partial Ensemble} = \mathcal{A} \times \mathcal{M}.$$

For this fusion method, a classifier is taken from $\mathcal{A}$, and is used to build the emotional histogram from the audio segments. This histogram is then concatenated to the bag of words to create the multimodal vector, which is used to train the classifier from $\mathcal{M}$.

Lyric Partial Ensemble

The lyric partial ensemble configuration is very similar to the audio partial ensemble, with the modification that the initial classification is performed on the bag of words vector instead of the audio.

The classifier set *Lyric Partial Ensemble* is defined as all the combinations between $\mathcal{L}$ and $\mathcal{M}$:

$$\textit{Lyric Partial Ensemble} = \mathcal{L} \times \mathcal{M}.$$

This fusion method takes a classifier from $\mathcal{L}$ and determines the likelihood that each bag of words belongs to each class, resulting in a vector of length 3. This likelihood vector is concatenated to the MFCC-like vectors from the audio data to create the multimodal features. A classifier from $\mathcal{M}$ is used to make the final decision based on this vector.

Full Ensemble

Taking both partial ensembles to the next logical step, full ensemble performs an initial classification on both the audio and the lyric vectors independently and then concatenates the intermediate results into a single multimodal vector. The audio vectors are converted into the same emotional histogram used in the audio partial ensemble method. The bag of words, like in lyric partial ensemble, is used to create a likelihood vector. The emotional histogram and the bag of words classification are concatenated and used as a single vector for multimodal classification.

The classifier set for *Full Ensemble* is defined by all combinations between the $\mathcal{A}$, $\mathcal{L}$ and $\mathcal{M}$ sets. The resulting set is defined as follows:

$$\textit{Full Ensemble } = \mathcal{A} \times \mathcal{L} \times \mathcal{M}$$

This fusion method takes a classifier from $\mathcal{A}$ to build an emotional histogram of the song and a classifier from $\mathcal{L}$ to build a likelihood vector for the song. These two vectors are concatenated to form the multimodal vector used to train a classifier from $\mathcal{M}$.

4

EXPERIMENT

Given the size of the dataset, it was necessary to perform significant preprocessing for each test to run in a sensible amount of time. The experiments require that every song has audio vectors, a lyric vector, and a true emotional label. To get the subset of the Million Song Dataset (MSDS) that had these characteristics, it was necessary to perform two levels of filtering. First we needed to identify which songs had sentiment labels. From this subset, we needed to identify which songs were also included in the MusicXMatch dataset. As a final step, the dataset was truncated so that each class had the same number of samples to avoid having any misrepresentation.

Song labeling was performed by looking at the mbtags in the metadata provided by the MSDS. If the tags contained the word 'happy' , 'angry' or 'sad', the corresponding song was considered to be part of that class. This method resulted in three distinct sets. These sets were cross referenced with the MusicXMatch dataset to determine which songs contained both an emotion label and bag of words features. From the intersection of the two sets, 1000 songs for each sentiment were randomly selected.

The algorithms detailed in Chapter 3 were implemented in Python using the sklearn toolkit[1][22]. A python module was developed to handle sampling and extracting features and class vectors, training classifiers and creating the emotional histogram. These functions were built into a single module to ensure that all the tests ran on the same code, thus avoiding any issues with accuracy difference due to code conflicts. To streamline the training and testing processes, 100 randomly selected audio segments were sampled for each song. This allowed the test program to quickly identify where the prediction for one song ended and the next one started without much overhead.

The module was then used to implement five different programs: series fusion, partial ensemble, full ensemble, audio and lyrics unimodal. It was important that each program could take user input to select which classifier is to be used for what portion of the algorithm. A script was written around these programs to try all classifier combinations as described in Chapter 3.

It is important to mention that the audio unimodal benchmark showed that Gaussian naive Bayes consistently provided better results than the multinomial naive Bayes, which allowed us to trim a subset of the combinations used for the full ensemble tests by requiring audio initial classification to be done with a Gaussian naive Bayes classifier. As a result, we ran six different unimodal benchmarks, nine different configurations for full ensemble tests, eighteen different partial ensemble tests and two series fusions, for a total of thirty-five different runs.

Each run consisted of training the algorithm four times with different sized training sets to understand the associated learning curves. The training sets were comprised of 20, 50, 100 and 500 sample songs for each class. After training the algorithm, it was tested against a set of 300 different songs which was also comprised of an equal number of songs for each class. Each run was repeated thirty times to have many data points for each configuration, to be able to compare performances more accurately. The confusion matrices were recorded for further analysis.

[1]The histogram intersection kernel code was obtained from kuantkid's branch of the open source scikit-learn project.
`https://github.com/kuantkid/scikit-learn/commit/16c82d8f2fe763df7bfee9bbcc40016fb84affcf`

5

RESULTS

This Chapter will present the results of the experiments detailed in Chapter 4 while providing some insight into what differentiates the fusion algorithms from the unimodal benchmarks. To do so, two different metrics are used to compare the algorithms. Average accuracy is the standard method for comparing classification algorithms. Since one of the goals of this paper is to improve the reliability of the overall system in diverse datasets, it is not beneficial to have near perfect accuracy for one class and sub-random choice for another. This gap is smoothed out while comparing average accuracies, which is why the algorithms are also compared using intra-class standard deviation. The intra-class standard deviation is computed by taking the standard deviation of all class accuracies.

Table 5.1 contains a summary of the accuracies of all the configurations trained with 500 samples for each class. Each row represents an experiment, and it is summarized with the average of all the class accuracies, the standard deviation of the averages and the intra-class stand deviation mentioned above. The naming convention used to describe each algorithm is as follows:

$$\{\text{Method AudioClassifier - LyricClassifier - MultimodalClassifier}\}$$

Where 'Method' can be a unimodal configuration (Audio or Lyrics) or a multimodal configuration ('Full Ensemble', 'Audio Partial', 'Lyric Partial' or 'Series'). AudioClassifier is an element from the set $\mathcal{A}$, LyricClassifier is an element from the set $\mathcal{L}$, and MultimodalClassifier is an element from the set $\mathcal{M}$.

It is interesting to notice some trends in Table 5.1. First note that the average accuracy for fusion algorithms tends to be slightly higher than the accuracy for the unimodal benchmarks at the top of the table. This is fur-

Table 5.1: Accuracy Measured in Average Accuracy and Intra-Class StdDev Using 500 Samples per Class

		Average	StdDev	Intra-Class StdDev	Average Rank	Intra-Class Rank
1	Audio Gaus	53.5600	2.67	24.469	25	24
2	Audio Multi	54.0900	4.11	23.176	24	23
3	Audio SVM	35.0800	6.94	28.028	35	28
4						
5	Lyrics Gaus	49.0400	3.13	27.875	26	26
6	Lyrics Multi	56.7100	3.13	16.295	19	19
7	Lyrics SVM	62.300	3.05	8.636	9	6
8						
9	Full Gaus-Gaus-Gaus	54.3800	3.47	24.761	23	25
10	Full Gaus-Gaus-Multi	58.0700	2.82	13.418	15	14
11	Full Gaus-Gaus-SVM	57.3300	4.32	13.033	17	12
12	Full Gaus-Multi-Gaus	60.4700	2.92	11.147	10	9
13	Full Gaus-Multi-Multi	57.3300	2.51	13.343	18	13
14	Full Gaus-Multi-SVM	58.7200	2.57	9.840	14	8
15	Full Gaus-SVM-Gaus	63.3900	2.17	**6.82**	7	1
16	Full Gaus-SVM-Multi	58.8800	2.91	12.935	13	11
17	Full Gaus-SVM-SVM	63.8800	2.77	6.981	4	2
18						
19	Partial Audio Gaus-Gaus	48.1100	2.71	29.645	30	34
20	Partial Audio Gaus-Multi	63.7700	2.04	14.356	5	15
21	Partial Audio Gaus-SVM	**68.46**	2.86	8.09	1	5
22	Partial Audio Multi-Gaus	48.1200	2.94	29.591	29	32
23	Partial Audio Multi-Multi	59.9200	4.22	17.635	12	22
24	Partial Audio Multi-SVM	66.7700	3.93	11.983	2	10
25	Partial Audio SVM-Gaus	48.8800	3.00	28.497	28	30
26	Partial Audio SVM-Multi	60.2400	2.73	16.562	11	20
27	Partial Audio SVM-SVM	65.1700	2.73	9.661	3	7
28						
29	Partial Lyric Gaus-Gaus	49.0200	2.70	28.391	27	29
30	Partial Lyric Gaus-Multi	48.0200	2.70	28.011	31	27
31	Partial Lyric Gaus-SVM	38.2800	9.54	31.087	33	35
32	Partial Lyric Multi-Gaus	57.9600	3.04	14.915	16	17
33	Partial Lyric Multi-Multi	56.6200	2.39	14.743	20	16
34	Partial Lyric Multi-SVM	56.4600	3.86	17.249	22	21
35	Partial Lyric SVM-Gaus	62.8600	2.09	7.830	8	3
36	Partial Lyric SVM-Multi	63.600	2.74	7.888	6	4
37	Partial Lyric SVM-SVM	35.9800	7.69	29.197	34	31
38						
39	Series Gaus	47.800	2.78	29.643	32	33
40	Series Multi	56.5100	2.46	15.273	21	18

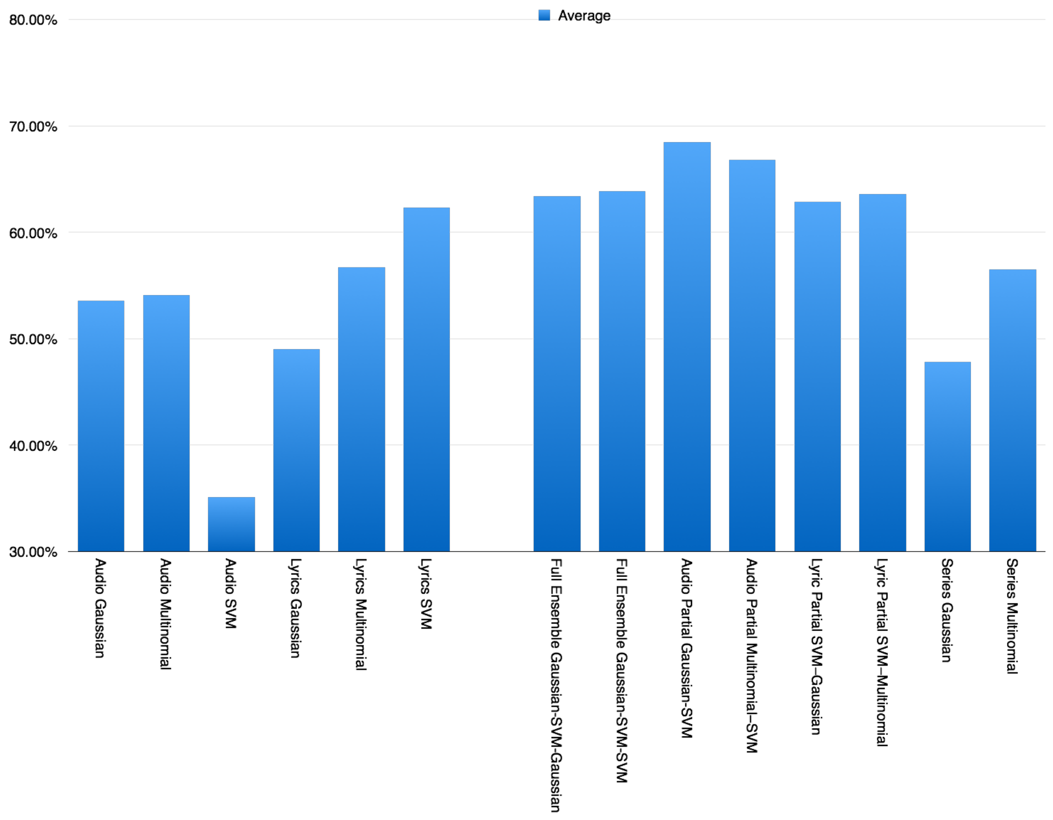

Figure 5.1: Average Accuracy for Top Algorithms at 500 Samples per Class. The left side contains the accuracies for the unimodal classifiers. The right side contains the accuracies for the top multimodal classifiers for each configuration.

ther explored by looking at the standard deviation of these accuracies; the benchmarks tend to have a higher intra-class standard deviation than the fusion experiments. Algorithms with a low intra-class standard deviation and a high average accuracy perform the best.

Figure 5.1 contains a representation of all the benchmarks and the top two algorithms for each configuration. One can quickly see that series methods do not perform very well. Looking at the other fusion methods, it becomes apparent that doing an initial classification of either raw audio or the raw bag of words before fusing the data had drastic benefits. The best unimodal method is the Lyrics SVM classifier shown in row 7 in Table 5.1. Even though it was the best unimodal classifier, it still ranked 9^{th} in accuracy. All algorithms with better ranking are multimodal methods.

Figure 5.1 shows the performance of the top classifiers and the top benchmarks using 500 training sample per class. It provides a good understanding of what configurations are most effective. Figure 5.2 contains the learning

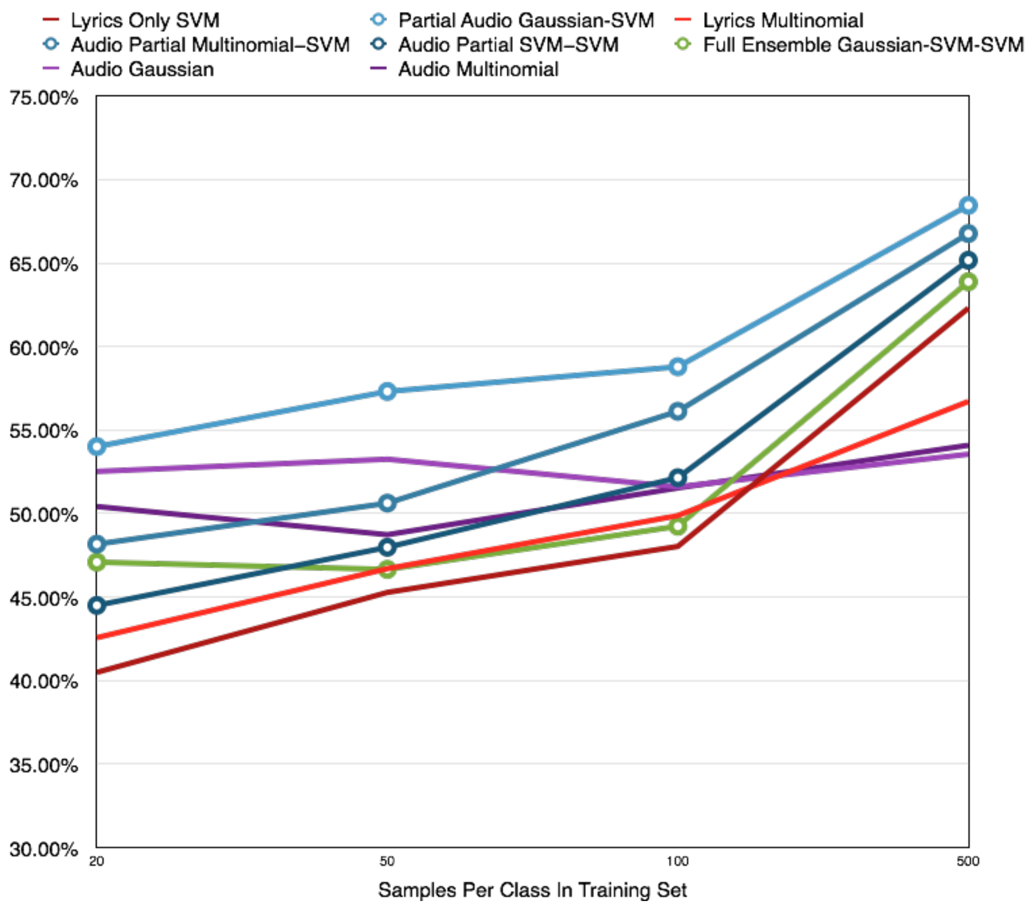

Figure 5.2: Learning Curves for Top Unimodal and Multimodal Classifiers. Plain lines represent unimodal classifiers. While the lines with circles represent the multimodal classifiers.

curve of some of the top classifiers for both unimodal and multimodal. Some of the unimodal classifier perform better at smaller datasets. However once the dataset reaches 500 samples per class, all the multimodal classifiers surpass the unimodal ones. Figure 5.2 also suggests that the training sizes used are not enough to overfit the data since the slope for most of the curves is increasing for all the points displayed. This behavior indicates that a larger dataset should provide better accuracies for many of these algorithms.

Better Reliability

Both the average and the intra-class variance serve as a good indication of the reliability of the system. A multi-class classification algorithm could possibly have strong average recognition accuracy as a result of performing really well for one class but poorly for the others. A high average and a high intra-class variance is highly indicative of this case. However, a high average paired with a low intra-class variance indicates that the classifier is both reliable and accurate.

Table 5.2 shows the average confusion matrices for the top two fusion algorithms and the top three benchmarks. The columns represent the true label "H", "S", "A" for the classes happy, sad and angry. The rows represent the algorithm's prediction "h", "s", "a" for happy, sad and angry.

Table 5.2 is very informative in terms of what classes are harder to recognize. Clearly, happy songs pose a greater challenge than angry songs. This makes sense because angry words tend to only be present in angry music. For example words like "wrath" or "rage" are more indicative of emotion than words like "sunny" or "cheer". The same occurs with the sad classification, but to a lesser extent.

Consider how the accuracy for angry songs remains relatively constant throughout all the experiments, but the accuracy for happy songs plummets in Table 5.2.d and Table 5.2.e. Table 5.2.e shows that the audio-only Gaussian classifier incorrectly selects the angry classification for 56.77% of happy songs and for 39.03% of sad songs. This behavior is evidence that the angry class is getting over-classified, since a disproportionate number of songs receive that classification. Similarly Table 5.2.d assigns happy songs to one of the three classes with almost equal probability, which suggests the classifier cannot identify this class as well as others.

These issues are at play in the case of unimodal classifiers, which would be undesirable for real world applications. As seen in Table 5.2.a and Figure 5.2.b, multimodal classifiers can identify happy songs more accurately without any noticeable loss in angry classification. Both Table 5.2.a and Figure 5.2.b also have a higher accuracy than the best unimodal classifier shown in Table 5.2.c. Additionally these classifiers do not select a particular class with random chance like Table 5.2.d, nor in a disproportionate manner like Table 5.2.e.

(a) Audio Partial Gaussian-SVM

	H	S	A
h	**58.93%**	15.60 %	16.97%
s	19.93%	**73.07 %**	9.67%
a	21.13%	11.33 %	**73.37%**

(b) Audio Partial Mutimodal-SVM

	H	S	A
h	**58.70%**	19.00 %	17.80%
s	18.70%	**67.10 %**	7.70%
a	22.60%	13.90 %	**74.50**

(c) Lyrics SVM

	H	S	A
h	**52.83%**	19.77 %	17.27%
s	23.13%	**63.93 %**	12.60%
a	24.03%	16.30 %	**70.13%**

(d) Lyrics Multinomial

	H	S	A
h	**36.93%**	13.37 %	7.87%
s	32.23%	**59.27 %**	18.20%
a	30.83%	27.37 %	**73.93**

(e) Audio Gaussian

	H	S	A
h	**24.23%**	7.80 %	7.53%
s	19.00%	**53.17 %**	9.20%
a	56.77%	39.03 %	**83.27**

Table 5.2: Confusion Matrices for Top Fusion and Top Benchmarks. Sub-tables (a) and (b) show that top fusion classifiers are more accurate and reliable than the top unimodal classifier in sub-table (b). Sub-tables (d) and (e) show the most prominent issues found in unimodal classifiers. The column 'H' in sub-tables (d) shows that the classifier is making random choices to classify the happy class. The row 'a' in table (e) shows that the classifier is over-classifying songs as angry.

6

CONCLUSION

Through the experiments and results presented in this research, we have reached the conclusion that multimodal classifiers improve both the accuracy and the robustness of the sentiment classification. We determined that some sentiments are harder to identify. It was also demonstrated that the recognition of these challenging sentiments can be improved greatly through multimodal classification without having a significant impact on the accuracy of other classes.

Although many configurations were introduced, the most promising were the 'Lyric Partial Ensemble' and the 'Audio Partial Ensemble'. These two groups dominated the ranks of the top ten in both accuracy and intra-class standard deviation. Based on the comparison metrics, 'Audio Partial Ensemble' architecture with an initial Gaussian classifier followed by an SVM with histogram intersection as a kernel is the strongest classifier explored.

7

FUTURE WORK

We explored the benefits of combining classification methods in different configurations. The features used for the paper underlined the added benefit of multimodal classification. Possible future work includes using more complex features that take into account sentence semantics to capture word modifiers. Another possible route for improvement on this paper would be to extend the number of classes used to all six Ekman emotions.

REFERENCES

[1] Thierry Bertin-Mahieux and Daniel P.W. Ellis and Brian Whitman and Paul Lamere, “The Million Song Dataset,” in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[2] MusicXMatch, “The MusiXMatch Dataset,” 2011. [Online]. Available: http://labrosa.ee.columbia.edu/millionsong/musixmatch

[3] B. Logan, “Mel-Frequency Cepstral Coefficients for Music Modeling,” in *In International Symposium on Music Information Retrieval*, 2000.

[4] G. Tzanetakis and G. Essl, “Automatic Musical Genre Classification Of Audio Signals,” in *IEEE Transactions on Speech and Audio Processing*, 2001, pp. 293–302.

[5] T. Li and M. Ogihara, “Detecting Emotion in Music.” presented at the International Society for Music Information Retrieval in Washington D.C., 2003.

[6] Hannah Davis and Saif M. Mohammad, “Generating Music from Literature,” presented at the European Chapter of the Association for Computational Linguistics (EACL) in Gothenburg, 2014.

[7] Jason Jong, “Predicting Rating with Sentiment Analysis,” 2011. [Online]. Available: http://cs229.stanford.edu/proj2011/Jong-PredictingRatingwithSentimentAnalysis.pdf

[8] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up?: Sentiment classification using machine learning techniques,” in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, ser. EMNLP ’02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002. [Online]. Available: http://dx.doi.org/10.3115/1118693.1118704 pp. 79–86.

[9] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning Word Vectors for Sentiment Analysis,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT ’11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002491 pp. 142–150.

[10] E. Haddi, X. Liu, and Y. Shi, “The Role of Text Pre-processing in Sentiment Analysis,” *Procedia Computer Science*, vol. 17, no. 0, pp. 26–32, 2013, First International Conference on Information Technology and Quantitative Management . [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050913001385

[11] Xia, Yunqing and Wang, Linlin and Wong, Kam-Fai, “Sentiment Vector Space Model for Lyric-Based Song sentiment Classification,” *International Journal of Computer Processing Of Languages*, vol. 21, no. 04, pp. 309–330, 2008.

[12] Zhong, Jiang and Cheng, Yifeng and Yang, Siyuan and Wen, Luosheng, “Music Sentiment Classification Integrating Audio with Lyrics,” *Journal of Information and Computational Science*, vol. 9, no. 1, pp. 35–44, 2012.

[13] Yang, Jian and Yang, Jing-yu and Zhang, David and Lu, Jian-feng, “Feature Fusion: Parallel Strategy vs. Serial Strategy,” *Pattern Recognition*, vol. 36, no. 6, pp. 1369–1381, 2003.

[14] Liang, Dawen and Gu, Haijie and O’Connor, Brendan, “Music Genre Classification with the Million Song Dataset,” 2011. [Online]. Available: http://www.ee.columbia.edu/~dliang/files/FINAL.pdf

[15] K. Ahmadian and M. Gavrilova, “A Multi-Modal Approach for High-Dimensional Feature Recognition,” *The Visual Computer*, vol. 29, no. 2, pp. 123–130, 2013.

[16] G. Caridakis, G. Castellano, L. Kessous, A. Raouzaiou, L. Malatesta, S. Asteriadis, and K. Karpouzis, “Multimodal Emotion Recognition from Expressive Faces, Body Gestures and Speech,” in *Artificial Intelligence and Innovations 2007: From Theory to Applications.* New York City, New York: Springer, 2007, pp. 375–388.

[17] Li, Shoushan and Zong, Chengqing, “Multi-Domain Adaptation for Sentiment Classification: Using Multiple Classifier Combining Methods,” in *Natural Language Processing and Knowledge Engineering, 2008. NLP-KE’08. International Conference on.* IEEE, 2008, pp. 1–8.

[18] X. Hu and J. S. Downie, "Improving Mood Classification in Music Digital Libraries by Combining Lyrics and Audio," in *Proceedings of the 10th Annual Joint Conference on Digital Libraries.* ACM, 2010, pp. 159–168.

[19] Ekman, Paul and Friesen, Wallace V and O'Sullivan, Maureen and Chan, Anthony and Diacoyanni-Tarlatzis, Irene and Heider, Karl and Krause, Rainer and LeCompte, William Ayhan and Pitcairn, Tom and Ricci-Bitti, Pio E and others, "Universals and Cultural Differences in the Judgments of Facial Expressions of Emotion," *Journal of Personality and Social Psychology*, vol. 53, no. 4, p. 712, 1987.

[20] Maji, Subhransu and Berg, Alexander C and Malik, Jitendra, "Classification Using Intersection Kernel Support Vector Machines is Efficient," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE, 2008, pp. 1–8.

[21] O. Chapelle, "Training a Support Vector Machine in the Primal," *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, 2007.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.