INDIANA UNIVERSITY - BLOOMINGTON

SortMyTunes

A Proposal for a New Hybrid Music Classification System

Martin McCrory

10/10/2007

I propose a hybrid (combining manual entry elements with computer automation) system of music classification that directly addresses the exact needs of the user. After a brief training period, where the user classifies a small subset of music into clusters of the user's design, the SortMyTunes algorithm places the remainder of the music into the clusters according to their similarity to the music already contained within a cluster. The user would periodically intervene to manually move music from one cluster to another, or to create or remove clusters, and SortMyTunes would recalculate the contents of the clusters based on the resulting distribution. This allows SortMyTunes to gain accuracy over each iteration of the program, as the clusters become more and more well defined. This hybrid system has the advantages of being fully open-ended and content-driven, while maintaining relative efficiency--each track is compared only with the average values of all of the tracks in a given cluster.

CONTENTS

INTRODUCTION

As collections of music grow larger and larger, and more and more diverse, it is getting more and more difficult for existing genre classification and similarity algorithms to run with any degree of accuracy and efficiency. Music on this scale is simply too complicated to be understood well enough by the computer at this time. Additionally, manual classification systems are rapidly becoming outdated, with the population of even personal musical databases routinely exceeding 10,000 tracks. Manual classification is simply an impossible endeavor on this scale.

Therefore, it is worthwhile to pursue developing a hybrid classification system that harnesses the human brain's ability to understand music and how it should be classified, while also harnessing the computer's ability to automate mundane tasks. However, even given a knowledge of what musical features exist in the music, there is still an issue to overcome: the fact that each person has his or her own understanding of how she or he wishes to classify music.

One user might wish to classify music into categories such as "Music for driving", "Music for exercising", "Music for relaxing to", etc. Another user, say, a MIREX developer, might wish to classify music into categories such as "The 1950s," "The 1960s," "The 1970s," and so on, for the development of ground truth to develop their own genre classification algorithms. Yet another user might wish to classify music more traditionally, into categories such as "Alternative," "Rock," "Classical," "Jazz," etc.

A worthwhile classification system should address each of these users, and be able to accommodate them in some fashion. Therefore, I propose creating a system of classification that allows the user to precisely dictate what music falls into what categories (hereafter referred to as "clusters"), while using computer-based tools to automate the tedious classification process.

THE SYSTEM - OVERVIEW

WHAT ARE CLUSTERS?

A "cluster" in this system refers to an unordered list of music tracks that have been classified by the user as being similar in some way. How are the tracks similar? This is left entirely up to the user, though the algorithm aims to streamline this process. The user can create as many or as few clusters as he or she desires; clusters can be added or removed at any time during the entire process.

WHAT MUSIC GOES INTO THE CLUSTERS?

Initially, the user selects exactly what tracks are placed into which clusters. This training period would comprise a few percent of the entire database (the user would be able to choose exactly how many tracks are used during the initial training period. After this training period, SortMyTunes evaluates the music that the user placed into the clusters over many different musical aspects, and uses this data to classify the remaining music into the clusters.

IF SORTMYTUNES IS SO DEPENDENT ON HUMAN INTERVENTION, WHAT'S THE POINT?

The point is that SortMyTunes aims to automate (as much as possible) a process that is time-consuming for humans. SortMyTunes serves to function as an automated extension and approximation of the human brain. A human would take many hours to accurately classify thousands of pieces, but SortMyTunes has the potential to do the same in just seconds, with just a minimal level of intervention on the part of the user. As SortMyTunes iterates over the database, if the user wishes to change a classification, the user can do this and the algorithm will re-calculate the attributes of the clusters to match. Of course, SortMyTunes will likely never reach the accuracy and effectiveness of the human brain at classification. But, with the help of the user, SortMyTunes may become very good at understanding the clusters that the user has invented, while doing so in a fraction of the time it would take the user to accomplish the same quantity of results.

The System – Technical

The Training Period

Initially, the user creates any number of clusters. The user selects a reasonably sized sample (enough for the computer to "get a feel" for how each Cluster will be formed) to use as a training set. The user puts each track into one cluster, and then submits this data to SortMyTunes. SortMyTunes calculates the values of many musical features of each track, and stores these values in a matrix, with each track representing a row and each feature representing a column. The values contained in each column of the matrix undergo some mathematical process (such as being averaged, finding the mode/median, etc.) to produce a value for that particular feature of all the tracks in that cluster.

The First Iteration

After the training period is concluded, a small iteration occurs, perhaps 5% of the database by default (the user would be able to specify the size of the iteration). SortMyTunes calculates the values of each musical feature for each candidate from the database, and compares it with the composite values representing the musical features of a given cluster. Whichever cluster matches up closest with the values of the candidate is the cluster in which the candidate is placed. This process repeats until the iteration concludes, at which point the composite values of the cluster are recalculated (but not before, to avoid cluster "stray" during the iteration, and to save CPU time). After the iteration concludes, the user can view the results of the iteration and make any manual changes required, including adding or deleting clusters.

This user intervention allows for each iteration to conclude with 100% accuracy, enabling further iterations to be more and more accurate in their classification.

Later Iterations

Each subsequent iteration can be larger than the first, due to the increased level of classification accuracy of each iteration. The user would be able to specify the size of each iteration. For example, the user could specify that the second iteration would classify 10% of the database into clusters, the third would classify 15%, then 20%, then 25%, then 25% (after a 5% initial iteration).

Programming the system

The framework of the system will initially be programmed in Java, for simplicity's sake. The framework is simple, with each track being represented as an instance of a class Track. The comparison is done on a mathematical level, with simple numeric comparisons of each normalized musical attribute occurring to determine similarity.

Initial prototypes of the system will operate on a metadata level, with the musical values being manually input (from a database such as Musicbrainz for example). Hopefully, later versions will include an algorithm that can intelligently deduce the musical values from the audio signal alone; this would likely be accomplished using Max/MSP.

The System – Musical

Musical Attributes

The system will calculate many musical attributes for each track. This list is still in development at this time, but it will include attributes such as:

```
int length;
String[] lyrics;
String[] instrumentsUsed;
boolean containsMaleSoloVocal;
boolean containsFemaleSoloVocal;
boolean containsEnsembleVocals;
boolean isTonal;
String[] containsKeys;
boolean isMetered;
String[] containsMeters;
String[] moodsInMusic;
String[] moodsInLyrics;
```

Determining Attributes

Algorithms for attributes such as mood detection, vocal detection, instrument detection are very difficult to program and "not even remotely reliable yet"[1]. Since SortMyTunes depends so much on correctly and efficiently determining these attributes, I plan on using a metadata approach to determining these attributes, at least initially. In the same way that Pandora works, expert users would listen to music and assign values to these attributes.

Or, perhaps more feasibly, SortMyTunes would tap into existing sources of metadata, such as Musicbrainz, and import that data onto the local machine for calculation. This would drastically increase the hard drive footprint of the track, but it would also drastically increase the efficiency of the algorithm, as integer or String comparisons can be done effectively in linear time, compared to

[1] Byrd, Donald A. Personal correspondence with author.

many time factors greater for even simple signal processing functions. Additionally, SortMyTunes could access the metadata online, eliminating the need for local storage of metadata.

Uses and Applications

Personal

SortMyTunes may be useful for users classifying a personal music database, compared to existing classification systems. Many only allow the user to place music into pre-existing categories, while others are bulky, inaccurate and time-consuming to use. Manually using ID3 tags or other metadata-based manual classification are more accurate, but this manual input is not as efficient as a computerized classification system. SortMyTunes may allow the user the flexibility of an ID3 tagging system (customized categories, manual control), while offering the automation features of the fully computerized classification systems.

Business

SortMyTunes should scale easily to business applications. Placing 10,000+ tracks into clusters such as “Best-sellers,” “Rapid Chart Risers,” “Flops,” “Popular With Grandparents,” etc. may be just as easy as using any other more traditional designations for clusters. Additionally, once the values of the musical attributes of each cluster are firmly established, SortMyTunes approaches the functionality of a fully automated system, with little intervention required from the user after each iteration. Batches can be hundreds, thousands, tens or hundreds of thousands of tracks without changing the functionality of the program.

Academic

SortMyTunes may allow for effective ground truth establishment for the development of new classification algorithms. For example, a developer of a mood classification algorithm could create clusters such as “Content,” “Anxious,” “Melancholy,” “Dark,” “Brooding,” etc. to determine with a high degree of accuracy which tracks in her test database can be classify with the appropriate mood. Similarly, a developer of an artist classification algorithm can create clusters such as “Beethoven,” “Brahms,” “Shostakovich,” “Ives,” etc. to associate his ground truth files with those artists with a very high degree of accuracy.

Conclusion

SortMyTunes will draw from the best of both manual classification systems and automated classification systems, to create a hybrid program that has the capability for efficient, yet customizable categorization. With many applications, from the personal to the business to the academic, SortMyTunes may have a wide range of possible uses. I anticipate that hybrid classification programs such as SortMyTunes will begin to see more and more use in the future, as music databases continue to increase in number, size and scope, and as existing automated classification systems fail to keep up.