

SortMyTunes

A New Hybrid Music Classification System

Martin McCrory

12/13/2007

SortMyTunes (SMT) is a “hybrid” music classification system, which harnesses the human being’s innate ability to understand how to cluster music, with the capabilities of today’s computer technology. Using SMT, a user can create clusters of music which can then be used to establish a comparison point for the k-means clustering algorithm. Using k-means, the tracks are sorted into the clusters that the user creates in an iterative process. SMT has various practical and academic uses, and can be used for a wide range of tasks, including personal classification, metadata ground-truth establishment and business applications.

CONTENTS

Introduction to Music Classification Systems.....	3
The Classification Process.....	4
The Training Period	4
The First Iteration.....	4
Later Iterations	4
System Programming.....	4
Uses and Applications.....	5
Personal	5
Business	6
Academic.....	6
Results.....	6
Appendix A: K-Means Algorithm	7
Appendix B: Acknowledgements	7
Appendix C: Bibliography	7

INTRODUCTION TO MUSIC CLASSIFICATION SYSTEMS

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 Additionally, manual classification systems are rapidly becoming outdated, with the population of even personal musical databases routinely exceeding 10,000 tracks.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 recognizes both that the human being is still more accurate than a computer at classification, and that a computer algorithm can classify at a much faster rate than the human. Allowing the (human) user significant input as to how the computer algorithm performs the classification is the key to attaining a balance between classification speed and accuracy.

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, and this understanding currently cannot be shared with a computer in a significantly literal way.

An example of a field within Music Information Retrieval that is taking advantage of this fact is the field of Music Recommendation systems. The latest MR systems, such as systems developed by Oscar Celma at Universitat Pompeu Fabra in Spain, use a “hybrid” tagging system (hybrid in the sense that the tags are applied by users, but recommendations occur through an automated process based on the manually established metadata) that allows users to express their feelings toward a particular musical entity, while offering a recommendation system that closely mimics the emotional behaviors of a human recommender.

However, In the field of Music Classification, this “hybrid” idea can be extended a step higher conceptually, from the users themselves specifying the tags (classifiers, in the case of Music Classification), to the users beginning the classification process from the “top”, and then the computer algorithm filling in the rest of the process from the “top down.” The user doesn’t have an interest in the features themselves, just how they are classified.

Therefore, the classification system outlined in this paper, SortMyTunes (SMT), allows the user to specify the categories and features for classification, and then based on the behind-the-scenes metadata about the tracks (harvested from a 3rd party source), the computer algorithm classifies the database according to the user’s exact specifications.

THE CLASSIFICATION PROCESS

THE TRAINING PERIOD

Initially, the user creates any number of clusters. The user then selects a reasonably sized sample of the database (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 SMT. SMT calculates the values of many musical features of each track, and stores these values in the form of an “average track”.

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). SMT 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 is basically a version of the k-means algorithm (though in truth, it is more similar to the k-medoids algorithm, since SMT creates an entirely new data point, the “average track”, with which it performs the classification comparisons). See appendix A for a description of the k-means algorithm.

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).

SYSTEM PROGRAMMING

The framework of the system is programmed in Java. Metadata about tracks is stored in textual form, without any audio feature extractors (in the same vein of a tag-based music recommendation system). The framework is not a complicated in and of itself, but the addition of dozens of feature classifiers is what would potentially create major hurdles in the construction process. Additionally, the iterative portion of the algorithm is not complete at this time—currently, SMT sorts all tracks in the database in one iteration, though this functionality is not conceptually difficult to implement.

In essence, each track is stored in a cluster (called a “Pod” in the system), and each feature has its own class within the framework. The Pod class performs the classification first, by determining the average track in each pod:

```
public void setAverageSong()
{
    Song toReturn = new Song();

    Meter mostCommonMeter = Meter.getMostCommonMeterValue(this);
    Key mostCommonKey = Key.getMostCommonKeyValue(this);
    Instruments mostCommonInstruments = Instruments.getMostCommonInstruments(this);
    toReturn.setMeterOfPiece(mostCommonMeter);
    toReturn.setKeyOfPiece(mostCommonKey);
    toReturn.setInstrumentsUsed(mostCommonInstruments);

    this.averageSong = toReturn;
}
```

Once that is accomplished, each candidate is compared to the average track in each pod according to k-means. Here is an example of how the track attribute Meter is compared:

```
public static int compare(Meter meterOne, Meter meterTwo)
{
    int toReturn = 0;
    if(Meter.areMetersEqual(meterOne, meterTwo))
        toReturn = 10;
    else if(Meter.areMetersEquivalent(meterOne, meterTwo))
        toReturn = 9;
    else if(Meter.areMetersSimilar(meterOne, meterTwo))
        toReturn = 8;
    else if(meterOne.numBeats == meterTwo.numBeats)
        toReturn = 6;
    else if(meterTwo.valOfBeats == meterOne.valOfBeats)
        toReturn = 3;
    else
        toReturn = 0;

    return toReturn;
}
```

Eventually, all of the features are compared using algorithms similar to the above. Currently, all of the features are text-based (there are no audio-based feature extractors).

USES AND APPLICATIONS

PERSONAL

SMT can 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. SMT can 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

SMT 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, SMT 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 with the same relative ease.

ACADEMIC

SMT can 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 high degree of accuracy.

RESULTS

SMT draws 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, SMT may have a wide range of possible uses. The k-means algorithm is an efficient way of clustering the tracks according to the user’s specifications.

Additionally, there is a significant possibility for SMT to draw directly from existing music recommendation systems that use a tagging process to create a “virtual identity” for tracks uploaded to the internet. SMT could adapt to incorporate a tagging system to give each song its own identity, rather than using a set of discrete features and risk incomplete or inadequate data for the features used. This way, SMT could collect as few or as many tags for each track, and use the comparison of these tags to determine appropriate clustering. This, however, would be a fundamental change in the framework of SMT, and would require major reprogramming.

I anticipate that hybrid classification programs such as SMT 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 with the increasing amount of music all around us.

APPENDIX A: K-MEANS ALGORITHM

Following is a description of the k-means clustering algorithm, taken from Wikipedia on December 13, 2007:

[k-means] starts by partitioning the input points into k initial sets, either at random or using some heuristic data. It then calculates the mean point, or centroid, of each set. It constructs a new partition by associating each point with the closest centroid. Then the centroids are recalculated for the new clusters, and algorithm repeated by alternate application of these two steps until convergence, which is obtained when the points no longer switch clusters (or alternatively centroids are no longer changed).

APPENDIX B: ACKNOWLEDGEMENTS

I would like to acknowledge Donald Byrd, for his extensive comments on the revisions of this paper, and for his many suggestions throughout each phase of this project. Also, I would like to thank Oscar Celma for sharing his work on music recommendation systems with me.

APPENDIX C: BIBLIOGRAPHY

Celma, Oscar. Music Recommendation: a multi-faceted approach. 2006.