
SORTMYTUNES 2.0 – A “HYBRID” MUSIC CLASSIFICATION SYSTEM

Martin McCrory | 1 May 2008

TABLE OF CONTENTS

Abstract	2
Predispositions in Music Classification	2
SortMyTunes 2.0.....	2
Introduction.....	2
Tag-Based Classification	3
The Iterative Process	4
Technical Features	4
Conclusion	5
Future Tasks for SortMyTunes.....	5

ABSTRACT

Music Classification is a significant problem in the field of Music Informatics. Currently, many classification systems exist. These systems tend to operate either by manual analysis (Pandora, etc) or by automatic metadata analysis (Last.fm, MyStrands, etc). SortMyTunes is a highly scalable system of a “hybrid” nature, which not only combines manual and automatic classification (by allowing the user to set the initial ground-truth for the classification process), but also scales well, as SortMyTunes can potentially become more and more accurate through more and more iterations of the classification process.

PREDISPOSITIONS IN MUSIC CLASSIFICATION

The most popular music classification systems, such as Pandora and Last.fm, are different, but excellent systems. These systems possess a large repository of information with which they use to project how the user will want their music to be classified. Usually, these predictions are correct (thanks both to the large quantities of data from which they draw, and to the highly powerful classification algorithms they use).

However, current systems have limitations. One such limitation is the need to have lots and lots of metadata, which slows down query retrieval times and requires lots of server space. *This aspect of the system essentially represents the system attempting to replace the vast amount of information about music that can be stored in the human brain.* Another limitation is the need to create complex classification algorithms to perform the classification. *In essence, these algorithms replace the human brain’s natural ability to classify music.* This is a problem not unique to music classification; it can be found in just about any subdiscipline of Music Information Research.

This problem parallels the “human flight” analogy, where humans who want to fly can either (1) become a bird (impossible) (2) build a set of Icarus flight wings, and attempt to fly with them (likely ending in a hot waxy death) or (3) build a machine that can transport humans through air, effectively simulating flying.

There is, however, a fourth solution: hire birds to carry the flier through the sky! Though in this particular analogy this solution is both humorous and impossible, in music classification this idea (of harnessing the human brain’s natural ability to classify music, while taking advantage of the powerful capabilities of classification algorithms) has a great deal of merit, and is the foundation for SortMyTunes.

SORTMYTUNES 2.0

INTRODUCTION

SortMyTunes started as the genesis of the above fundamental idea, that an efficient and accurate classification algorithm should be centered on the ability of the human brain (analogous to the bird) to classify music (analogous to the act of knowing how to fly). SortMyTunes, therefore, is a “hybrid”

system of classification, incorporating both manual classification and automated algorithm-based classification.

SortMyTunes is designed to perform classification on a “Personal Music Collection,” or PMC. A PMC is defined as a relatively static collection of music that on which a single user is interested in performing classification. Some examples of a PMC include:

- Collections of music on one’s personal computer
- Music stored on an online playlist
- Collections of albums in a large business database
- Ground-truth music in a research dataset

Note that many music collections can be considered PMCs for the purpose of SortMyTunes.

When classifying a PMC with SortMyTunes, the user performs a manual classification over a small portion of the PMC. SortMyTunes then iterates over a portion of the PMC and performs automatic classification on that portion. The user makes any changes and corrections, and then SortMyTunes, now having more accurate classification information, iterates over a larger portion of the PMC. This process repeats, with larger and larger iterations until the entire PMC is classified.

The first attempt at SortMyTunes used a fixed feature-based classification system, with features built-in such as “Genre,” “Mood,” “Key,” “Meter,” etc. This proved to create several problems:

- Programming difficulty
- Missing metadata from candidate tracks
- Different classification methods for different features

Eventually, it was decided that feature-based classification was not the most efficient way for SortMyTunes to operate, so the decision was made for SortMyTunes 2.0 to move to an entirely “tag-based” classification system.

TAG-BASED CLASSIFICATION

A “tag,” in the case of SortMyTunes, is a manually created fragment of metadata that describes a track in a certain descriptive way. Each track may have many tags associated with it, and each tag may be associated with many tracks. There are no restrictions on how a tag may describe a track. For example, some tags that may be found associated with “Gimme Moah” by Britney Spears may be <Britney, Spears, Pop, R&B, Rock, Live, Pregnant, Justin, Fat, Bald, Awesome, Stupid, Sucks> (source: last.fm). Notice that the tags encompass a wide gamut of descriptions of the song, from the genre, to the method of performance, to the listener’s reaction to the song, to the artist’s name, to tags not related to the music at all (e.g. Justin).

The reason that tags are critical to the classification process is that, by definition, *these tags represent how the listener perceives the track on some fundamental level*. A listener who tags a song with the tag

“awesome” obviously feels some connection to the song such that s/he tags it in that certain way (ignoring malicious tagging, of course). Therefore, when SortMyTunes classifies according to user-generated tags, SortMyTunes can expect that a certain degree of correlation between the tags and how the user perceives the music to occur.

Another reason that SortMyTunes uses a tag-based classification system is efficiency. Analyzing a set of a few dozen tags for a given track is many times more efficient than, say, performing feature extraction on the audio component of the track. Additionally, with feature extraction there is no guarantee that the features one is extracting represent how the user actually perceives the music. When classifying with tags the likelihood of a correlation is much higher.

THE ITERATIVE PROCESS

When first performing a classification with SortMyTunes, the user initially creates “pods” of music in which tracks will be placed. These pods can have whatever meaning the user wants them to have (for example, a user may choose pods with descriptions like “80s,” “Rock/Pop,” “Driving Music,” “Music I Hate” or “Baroque”). After creating the pods, the user selects a small number (from 1-10) tracks from the PMC to manually classify into each pod. These tracks represent a portion of the “manual” component of the SortMyTunes classification process, and by definition this classification has 100% accuracy (since the user has manually confirmed each classification).

After this manual component is complete, SortMyTunes iterates over a small portion of the PMC. SortMyTunes compares each classification candidate with the metadata contained in each of the pods. For each candidate, whichever pod has the most thorough match with the candidate’s metadata is the pod in which the candidate is classified.

Once this automatic component is complete, the user intervenes and makes any modifications to the resulting pod-set. The user may re-classify tracks, add or remove tracks, or even add or remove pods. Again, at this point, the classification has 100% accuracy by definition.

Once the user is satisfied with the configuration of the pods, SortMyTunes iterates over another, larger, portion of the PMC and classifies each candidate as above. SortMyTunes possibly afford to iterate over a larger portion of the database because SortMyTunes has more ground truth metadata information, and thus ideally could make a more accurate classification judgment. In other words, the size of the iteration is chosen such that with each iteration, the user has to make approximately the same number of changes.

This process continues until all of the tracks in the database have been classified, and the user is satisfied with the configuration of all of the tracks in the pods.

TECHNICAL FEATURES

SortMyTunes has two notable technical features. The first is that SortMyTunes is programmed entirely in Java. Java is used because Java, as an Object-Oriented language, is nicely suited to the demands of

this algorithm, which is heavily grounded in objects such as Tags, Pods, Tracks and Databases. Additionally, Java's capability for creating user interfaces and working with large databases of metadata is satisfactory.

The second notable technical feature is the choice of classification algorithm. SortMyTunes uses the *k-means* classification algorithm, which is well-suited for classification when (1) the number of clusters is known, and (2) the dataset does not change much during the classification process. Since both of these conditions are true for SortMyTunes, *k-means* was selected. For a summary of the *k-means* classification algorithm, please visit <http://en.wikipedia.org/wiki/k-means>.

CONCLUSION

SortMyTunes' hybrid nature is a response to the current state of music classification systems, which tend either to rely on manual or automatic feature extraction, which is either time-consuming or error-prone, or automatic metadata-based classification, which by definition can never approach the accuracy of a manual classification system. SortMyTunes attempts to bridge the gap between these two systems of classifications, incorporating the accuracy of a manual system with the efficiency of the complex classification algorithms of the automatic systems. The unique nature of SortMyTunes' iterative process also allows for efficiency to scale on a quasi-logarithmic level, with the size of each iteration increasing as the number of iterations increase.

FUTURE TASKS FOR SORTMYTUNES

Continued development of SortMyTunes will necessitate attention being paid to the following actionable items:

- Creation of a simple user interface. This should not be challenging in the Java environment, as there are many built-in tools to make this easy in Java.
- Improve the efficiency of the classification process, which, for no particularly good reason, is stuck at $O(n^2)$. In theory, *k-means* should operate at $O(n \cdot \log(n))$, which basically means that the SortMyTunes implementation of *k-means* needs to incorporate some more efficient processes.
- Test SortMyTunes on a larger dataset. Current tests have only been executed on small test sets.