in the signal. The following equations define the MFCC, where each frame is referred to by the superscript k and γ is set to a small value to avoid taking the logarithm of 0:

$$L_{mfcc}^{(k)} = log(F_{mel}^{(k)} + \gamma)$$

$$r_{mfcc} = dct(L_{mfcc})$$

To clarify, what I refer to as an MFCC is a 20 dimension vector which represents a near instant of sound. Each coefficient, or element in the vector, is a number which represents the loudness of a harmonic series in the signal. We may simply further and say the each coefficient represents loundness in a certian frequency range since lower frequency harmonics tend to mask higher frequency harmonics. In the remainder of the paper, we refer to a single vector as an MFCC and a collections of those vectors which represent an audible amount of audio (generally one or more entire songs) as MFCCs.

The data set used contains 400 artists, 8764 tracks, and 251 unique style tokens. The MP3s were downsampled to 22050 Hz, and mixed to mono using the mpg123 and feacalc utilities. Each 32ms frame is distilled into 20 coefficients. [5]

Given the a large number of MFCCs, we must now find a way to model them – there are too many for them to considered individually. The methodology chosen here is selected for both for its simplicity and effectiveness. Again drawing from the speech recognition community, the MFCCs are modeled as a bag of frames represented by a Gaussian Mixture Model. The GMM is trained using the well established technique of initializing with the K-means clustering algorithm and then honing in with Expectation Maximization. For further explanation, the reader is referred to the excellent survey of clustering techniques by Rui Xu and Donald Wunsch II [10]. Of note is that a well

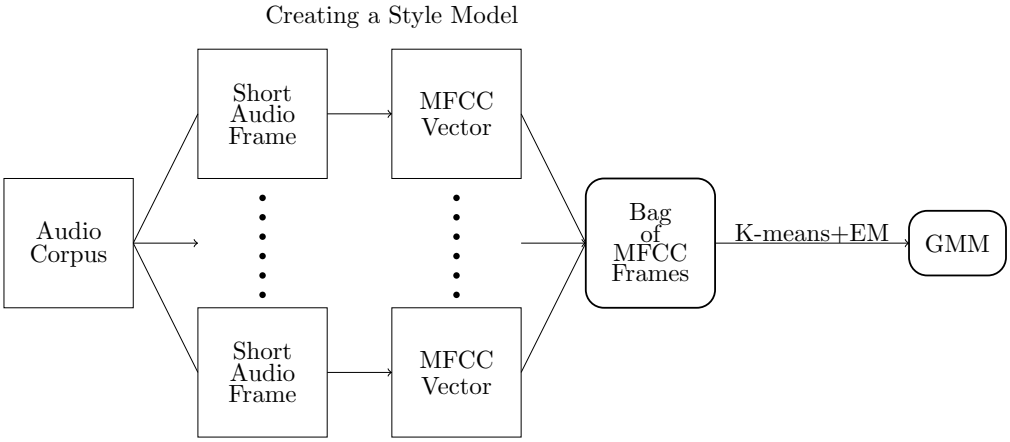

Figure 3.1: The figure shows the creation of a style model from an audio corpus containing the songs which represent that style.

known problem with EM is that sometimes an outlying point gets isolated to a single gaussian in the mixture model, sending its covariance to zero and resulting in division by zero errors. Though this problem did surface at one point, the Netlab toolkit I used provided an option of handling this issue [9].