一　问题概述

在哼唱搜索的整个过程中，有两大重要的问题需要解决：第一，将人声通过处理得到每个音的基频，形成哼唱的频率序列；第二，将得到的哼唱的频率序列与数据库中每首歌曲进行比较，得到相似度高的歌曲。这里主要对第二个问题进行分析和实现。

二　分析与假设

1　可行性分析

根据之前的假设，我们只考虑单音，并不考虑音长等各种其他因素，故得到的哼唱频率序列仅仅是单音的基频序列，所以在这个问题中的输入即为一段频率序列。相应的，我们要想根据这个输入进行匹配，只能通过实际歌曲的单音的基频序列进行比较，两者才具有可比性。因此我们必须要得到每首歌的单音序列，而歌曲的乐谱我们是可以获得的，根据乐谱和标准音符频率对照表，就可以得到每首歌对应的单音的基频序列。所以哼唱的频率序列和歌曲是可以进行匹配的，能够满足问题的前提。

2　基本假设

（1）输入序列是哼唱的单音基频序列，即一组浮点数序列
（2）歌曲的模板序列可以通过乐谱和标准音符频率对照表获得
（3）哼唱频率序列是标准模板序列的一个连续子序列

3　问题简化

通过以上可行性的分析与假设，问题可以简化为：输入的一组浮点数序列，能否在另一组浮点数序列中找到与之相似的一组序列。故我们可以进一步将问题归纳为：如何衡量这两组浮点数序列的相似度。通过比较输入的哼唱频率序列和数据库中每首歌的相似度，从而找到用户实际所唱的那首歌。

4　困难与挑战

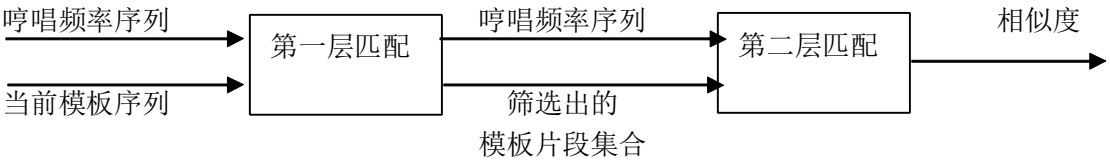
由于在这个问题中是非精确的匹配，故无法采用一般的匹配算法进行比较，用什么样的方法来衡量两个序列之间的相似度是需要仔细研究的问题。同时，还存在另外一个非常重要的问题：我们每个人在唱歌的时候，起的音调（即起始频率）是不一样的。也就是说，每个人所唱的歌在整体上的音调可能会有很大差异，与标准乐谱也极有可能存在很大差异，这就给两者之间的匹配提出了更严格的要求，要能够容许这种整体音调的差异。

三　模型建立

1　模型概述

基于上述对问题的分析和简化，为了解决现有的匹配问题，我们提出了分层匹配模型。整个模型的输入为哼唱频率序列和当前比较的标准模板序列，匹配过程分为两个层次，第一层进行粗略地筛选，找出当前模板序列中与哼唱频率序列相似度较高的模板片段集合，然后将这些模板片段再经过第二层的匹配，计算出每个模板片段与哼唱频率序列的相似度，选择相似度最大的那个模板片段，认为这个模板片段与哼唱频率序列的相似度最大。最后，综合两层计算的相似度，按照一定的权重计算出哼唱频率序列和当前模板序列的相似度。

2　模型框架图



3　第一层匹配建模

（1）初始化哼唱频率序列和模板序列

为了解决每个人唱歌的音调不同对匹配的影响，我们需要对获得的实际哼唱频率序列做一些变换。注意到，虽然每个人唱歌的起调不同会造成整体音调偏高或者偏低，但从唱歌的连续性考虑，如果画出 $f-t$ 图可以看到，单音频率的值随时间的变化所形成的图形是基本一致的，即哼唱频率序列的变化与模板序列的变化是一致的。故我们采用差分的思想来分别初始化哼唱频率序列和模板序列，用后一个频率值与前一个频率值的某种差的形式重新得到哼唱频率序列和模板序列。这种差的形式，我们将利用音符里面的十二平均律的相关知识进行，具体不再详述，仅列出差分公式如下：

$$n_i = 12 * \log_2(f_i / f_{i-1})$$

其中，n_i 为音程差（即两个频率值之间相隔的音符个数），f_i 和 f_{i-1} 为相邻两个频率值。

（2）计算哼唱音程差序列与相应模板音程差片段的相似度

①计算欧式距离

此时，从模板音程差序列中取出相应的一个片段，计算出此时哼唱音程差序列和这个片段的欧式距离，记为 D，即 $D=\sum_{i=1}^{m}\left|n_{1i}-n_{2i}\right|$ ，其中 m 为哼唱音程差序列的个数，n_{1i} 和 n_{2i} 分别为两段序列的每个对应音程差的值，两段序列的个数应该相等

②计算相似度

为了便于衡量，相似度应该取为一个百分比，所以在这里我们建立以下公式来衡量此时两者之间的相似度 S_1：

$$S_1=(K*m-D)/(K*m)$$

其中，K 为一个常数，这里我们将其设为 6

（3） 筛选模板片段

①判断是否为可能的匹配片段

并非每个片段都能够进入第二层进行匹配，有一些明显不符的片段将会被排除掉，只留下少数可能匹配的片段。

通过对模板音程差序列的扫描，分别计算出每一种搭配之间的相似度，然后与一个阈值进行比较，若不小于这个阈值，则这个片段可以通过第一层匹配进入到第二层匹配，否则即被排除。在这里，由于受上述计算相似度公式的影响，将阈值 $THRESHOLD$ 设置为 0.8。

②确定满足条件的所有模板片段

当通过上述过程之后，就会得到一个带有所有可能与哼唱序列匹配的模板子序列的集合，第一层匹配结束。将这个集合和哼唱序列送入第二层匹配，进行最终的匹配。

4 第二层匹配建模

（1）DTW 算法简介

在孤立词语音识别中，最为简单有效的方法是采用 DTW（Dynamic Time Warping，动态时间归整）算法，该算法基于动态规划（DP）的思想，解决了发音长短不一的模板匹配问题，是语音识别中出现较早、较为经典的一种算法。用于孤立词识别，HMM 算法在训

练阶段需要提供大量的语音数据，通过反复计算才能得到模型参数，而 DTW 算法的训练中几乎不需要额外的计算。所以在孤立词语音识别中，DTW 算法仍然得到广泛的应用。

在这里，我们采用 DTW 算法用于比较哼唱音程差序列和筛选过的模板音程差序列集合中的每一个元素。DTW 算法的基本思想就是，我们需要在两个序列之间找到一条非线性的匹配路径，即并不像第一层匹配那样两组序列的对应的值做欧式距离求和，而是允许序列进行压缩和扩张，求出这种非线性的路径中，到达两组序列各自终点的最短路径，即为最佳匹配路径，对应的总距离即为两组序列之间的距离。由于 DTW 算法是 DP 算法，相应的状态转移方程如下所示：

$$D(i,j)=\begin{cases}\min(D(i,j-1),D(i-1,j),D(i-1,j-1))+d(i,j) & i,j>1\\ d(i,j) & i=1\,||\,j=1\end{cases}$$

其中，$D(i,j)$ 是分别以 i,j 作为终点的两组序列的距离，$d(i,j)$ 是两组序列的第 i 和第 j 个数之间的欧式距离

（2）对 DTW 算法的一点改进

在我们这个问题里面，由于要比较的是两个完全相等的序列，而且通过第一层匹配之后，可以认为两组序列之间的线性关系较大，也就是说，我们的第一层匹配是建立在线性关系之上的，即两组序列的相似度是线性的相似度。但是，由于 DTW 算法可以处理非线性的关系，要想使 DTW 算法在这个问题中发挥应有的作用，需要对其进行一点改进。

因此，根据 DTW 算法的特点，我们对其进行以下改进：首先调用最原始的 DTW 算法，计算出 $D(m,m)$，加到总距离 D 中。从 $D(m,m)$ 开始回溯，寻找到达 (m,m) 的最佳匹配路径，此时必然有至少一个序列回溯到第 1 个值，另一个序列回溯到第 k 个值。若 k=1，则此时两组序列是彼此对应的数做匹配，则匹配结束。若 $k=t,t\neq 1$，则重新计算 $D(t-1,t-1)$ 并加到总距离 D 中，并重复以上过程，直至匹配结束。在这个过程中，记录下总的匹配点数 n 和总的距离 D

（3）计算第二层匹配相似度

根据计算公式：

$$S_2=(K*n-D)/(K*n)$$

其中，S_2 为第二层相似度

（4）从模板子序列集合中找出与哼唱序列的最大相似度

模板子序列集合中的每一个片段分别与哼唱序列根据 DTW 算法做匹配，取其中的最大相似度的片段，认为其是这首歌中与哼唱序列相匹配的片段，这个最大相似度作为与哼唱序列的第二层匹配的相似度 S_2，这个片段对应的第一层匹配的相似度为 S_1。

5　总相似度计算

根据以上过程可以得到，第一层匹配的相似度 S_1，第二层匹配的相似度 S_2，则可以计算出哼唱序列和模板序列之间的总相似度：

$$S = \omega_1 * S_1 + \omega_2 * S_2$$

其中，ω_1, ω_2 分别为两阶段的权重，这里均设为 0.5

四　实现

根据模型，我们可以得到数据库里的每首歌与哼唱频率序列之间的相似度，然后根据这个相似度从大到小进行排序，即可以找到我们所需要的歌曲。

两层匹配模型的实现代码具体见附录。

五　问题与不足

1　通过两层匹配模型，可以很快的排除许多与哼唱序列不相似的模板子序列，提高了整体匹配的速度。但是，哼唱序列由于人唱或者前期处理等原因，极有可能会有不同程度的误差。当出现个别频率值误差较大，但其他值匹配程度很高的情况时，第一层匹配很有可能将其排除。所以，第一层排除的片段，无法保证一定不是真正相匹配的片段，从而造成匹配的失败。

2　在建立计算相似度的公式时，虽然将衡量距离转换成衡量百分比，更符合相似度特点，但公式选择的不当，导致了整体匹配的相似度偏高，即无论是真正匹配的片段，还是本不相匹配的片段，相似度的值均较高，不太符合对相似概念的直观感受。