Algorithms of Internet Music Search Engine by Humming and Performance Evaluation

DONG Yun-feng[1], QI Bei [2]

1. Computing Center, Shandong Institute of Light Industry, 250353,Jinan,China

2. Network Center, Shandong Institute of Light Industry, 250353,Jinan,China

1. dyf@sdili.edu.cn, 2. qb@sdili.edu.cn

Abstract: This paper have made a study of algorithms that commonly used in internet search engine by humming, and evaluated the performance of algorithms in the large-scale music search system by humming. In order to support study of various approximate match algorithms' compare, an algorithm-independent test platform based on network was developed to evaluate the performance of algorithms. The basic algorithms include Suffix-tree, Hidden Markov Models (HHM), approximation melody algorithm, Dynamic Time Wrapping (DTW) and similarity matching algorithm. The author have carried out large amount of experiments on the algorithms of approximation melody algorithm, Dynamic Time Wrapping (DTW) and similarity matching algorithm, analyzed and evaluated them. The experiment results indicate that, similarity matching algorithm is fit to internet music search engine by humming.

Keywords: query by humming; large-scale music warehouse; Dynamic Time Wrapping; approximation melody algorithm; similarity matching algorithm

网络音乐哼唱检索引擎中的算法及性能评测

董云峰 [1]，亓蓓 [2]

1. 山东轻工业学院计算中心，济南，中国，250353

2. 山东轻工业学院网络中心，济南，中国，250353

1. dyf@sdili.edu.cn, 2. qb@sdili.edu.cn

【摘要】本文研究了网络音乐哼唱检索引擎中的常用算法，并对算法在大型音乐哼唱检索系统中的性能进行了评测。为了支持对于多种近似匹配算法的比较研究，开发了一个独立于算法的网络测试平台用于测试多种算法的性能。所研究的基本算法包括后缀树方法、隐马尔科夫模型（HMM）方法、近似旋律算法、动态时间规整（DTW）方法和相似度匹配算法。对其中的近似旋律算法、动态时间规整（DTW）方法和相似度匹配算法进行了大量实验，分析评价了它们的性能。实验结果表明，相似度匹配算法是适用于网络音乐哼唱检索引擎的查询算法。

【关键词】哼唱检索；大型乐曲库；动态时间规整；近似旋律算法；相似度匹配算法

1 引言

近几年来，网络音乐检索引擎开始发生变革，一种新的通过用户哼唱输入检索条件检索歌曲的网络检索引擎开始出现。它以通过用户哼唱的旋律找出用户所要求的音乐文件，这种方式对于网络用户而言更直观，更贴近音乐的本质。这种哼唱检索是一种基于内容的音乐检索系统（content—based music retrieval），允许用户通过哼唱的形式来寻找所需的歌曲。用户只要能回忆起其中的片断旋律，并用麦克风哼唱出来，检索系统就能找到所要的歌曲。

对于网络音乐检索而言，乐曲库的容量是相当庞大的，因此对网络音乐进行哼唱检索应当把研究重心放在大型乐曲库的检索上。现在的大型乐曲库的哼唱检索方法主要分为严格匹配和近似匹配两种算法，它们适用于不同的环境。但是很少有人将这些算法放在具有大数据量的环境中，从命中率和反应时间两方面全面评测它们的性能。针对于网络音乐检索引擎，人们希望的是输入一段可能包含一些错误的哼唱，系统在尽可能短的时间

里返回它想查询的较为准确的结果。有的论文以查全率和查准率为标准，对已经有的一些算法进行了性能比较[1]。但是针对于网络搜索引擎，作者认为应以查询命中率和查询的响应时间为评价标准，即在最短时间内命中率最高这样一个标准，在统一的测试平台对以下几种大型音乐库哼唱检索算法的性能进行评测。

2 适用于大型音乐库哼唱检索的近似匹配算法

目前多种基于内容的音乐检索算法中，最具代表性的有：基于后缀树的索引方法，动态时间规整（DTW）方法，隐马尔科夫模型（HMM）方法，近似旋律匹配算法等，还有作者设计并实现的相似度匹配法。

2.1 后缀树方法

文献[2]中提到对含有 4000 余首民歌，每首乐曲的长度为 210 个字符（远小于实际乐曲的长度）的音乐数据库，建立后缀树索引，则索引占用的内存空间为 3.3G，而且扫描一次约需 42s。这种方法对于有更多歌曲的系统基本是无法建立有效的索引，因此不能用于大型哼唱检索系统。

2.2 隐马尔科夫模型方法

HMM 方法具有很好的容错性。但是，文献[3]中给出的测试结果，对数据库中的每条记录，HMM 方法需要计算 2s，因此，对于 10,000 首歌的数据库则需要 20,000 s，即 5.56 h，这样长的检索时间，用户是不能忍受的，因此对于大型哼唱检索系统显然也是不适合的。

2.3 动态时间规整方法

基于动态时间归整匹配的 DTW 算法从目前来看，可能是一个最为小巧的音频识别的算法。其系统开销小，识别速度快，在对付少量单一音频控制系统中是一个非常有效的算法。

2.4 近似旋律匹配算法

现有的一些基于内容的音乐检索系统多采用近似字符串匹配算法，如 DP 法。这种方法在容错方面有其明显缺陷，针对该缺陷以及旋律的一些特性，改进近似字符串匹配算法为近似旋律匹配算法。

根据音律具有的节奏性特征体的方法是：先把两段旋律（即两个音符序列）在时间轴上线性延展到相同的长度，并在一定的误差范围内对齐发声时刻接近的音符，考察旋律在节奏上的相似性。之后，继续比较两段等长旋律在每个时间点上音高频率的距离。采用音高差的形式表示旋律，保证了用户可以用任意的音调哼唱。最后，综合考虑节奏和音高两方面的相似程度，给出匹配评价。

2.5 相似度匹配算法

相似度匹配方法是作者设计的用于大型音乐检索系统的快速检索方法，其基本思想是：计算离散特征音频相邻帧的前后音差，记录每次音调变化超过给定阈值的时间，每段时间是不定长的。对于每一个音乐文件进行相似度计算。算法如下：

第一步：计算哼唱信号片段基频周期值的两个对应的基频周期值变化数组。设 M[i]为存放哼唱信号片段基频周期值的数组，其中 M[i]=p_i 0<i≤N_m，N_m 为所求出的哼唱信号片段中的音调个数。设数组 $M_{F1}[i]$，其中 0<i<N_m，用来表示当前音调的基频周期值同前一音调的基频周期值之间的变化关系，其对应于音乐文件特征提取的旋律特征二。可根据下式求得：

$$M_{F2}[i] = \begin{cases} 0 & i=1 \\ M[i]-M[i-1] & 1<i\le N \end{cases} \quad (1)$$

设数组 $M_{F2}[i]$，其中 0<i<N_m，用来表示当前音调的基频周期值同哼唱信号片段中第一个音调的基频周期值之间的变化关系，其对应于音乐文件特征提取的旋律特征三。可根据下式求得：

$$M_{F3}[i] = \begin{cases} 0 & i=1 \\ M[i]-M[1] & 1<i\le N_m \end{cases} \quad (2)$$

第二步：提取音乐文件的第一个窗口的特征矩阵，如果窗口长度同哼唱信号片段的音调个数相同，则进行如下计算：设 $W_1[i][j]$为窗口特征矩阵，其中 0<i≤3，0<j≤20。 N=$W_1[1][1]$为窗口长度。N_m 为所求出的哼唱信号片段中的音调个数，此时 N=N_m。相似度值（similarity）用 S_t 表示：

$$S_t = \frac{\sum_{i=2}^{N} S_{F1}(i)}{10*(N-1)}*0.7+\frac{\sum_{i=2}^{N} S_{F2}(i)}{20*(N-1)}*0.3 \quad (3)$$

其中

978-1-935068-31-0 © 2010 SciRes.

$$S_{F1}(i)=\begin{cases}0, & W_1[2][i]=0 \ \ |M_{F2}[i]|>10\\ 10, & W_1[2][i]=0 \ \ 4\le|M_{F2}[i]|\le 10\\ 10, & |W_1[2][i]|=1 \ \ |M_{F2}[i]|>10\\ 20, & |W_1[2][i]|=1 \ \ 4\le|M_{F2}[i]|\le 10\\ 10, & |W_1[2][i]|=1 \ \ |M_{F2}[i]|<4\\ 0, & W_1[2][i]>1 \ \ |M_{F2}[i]|>10\\ 20, & W_1[2][i]>1 \ \ |M_{F2}[i]|<-10\\ 10, & W_1[2][i]>1 \ \ 4\le|M_{F2}[i]|\le 10\\ 0, & W_1[2][i]>1 \ \ |M_{F2}[i]|<4\\ 20, & W_1[2][i]<-1 \ \ |M_{F2}[i]|<4\\ 0, & W_1[2][i]<-1 \ \ |M_{F2}[i]|<-10\\ 10, & W_1[2][i]<-1 \ \ 4\le|M_{F2}[i]|\le 10\\ 0, & W_1[2][i]<-1 \ \ |M_{F2}[i]|<4\end{cases} \quad (4)$$

$$S_{F2}(i)=\begin{cases}0, & W_1[3][i]=0 \ \ |M_{F3}[i]|>10\\ 10, & W_1[3][i]=0 \ \ 4\le|M_{F3}[i]|\le 10\\ 10, & |W_1[3][i]|=1 \ \ |M_{F3}[i]|>10\\ 20, & |W_1[3][i]|=1 \ \ 4\le|M_{F3}[i]|\le 10\\ 10, & |W_1[3][i]|=1 \ \ |M_{F3}[i]|<4\\ 0, & W_1[3][i]>1 \ \ |M_{F3}[i]|>10\\ 20, & W_1[3][i]>1 \ \ |M_{F3}[i]|<-10\\ 10, & W_1[3][i]>1 \ \ 4\le|M_{F3}[i]|\le 10\\ 0, & W_1[3][i]>1 \ \ |M_{F3}[i]|<4\\ 20, & W_1[3][i]<-1 \ \ |M_{F3}[i]|<4\\ 0, & W_1[3][i]<-1 \ \ |M_{F3}[i]|<-10\\ 10, & W_1[3][i]<-1 \ \ 4\le|M_{F3}[i]|\le 10\\ 0, & W_1[3][i]<-1 \ \ |M_{F3}[i]|<4\end{cases} \quad (5)$$

其中 $2\leqslant i\leqslant N$。

在这一步骤中计算离散特征音频帧的相邻单调变化值，记相邻两次音调变化幅度与给定阈值相等的周期，每个周期的长度是不确定的，对每一个周期进行模糊匹配。虽然用户对音调把握的不是很准确，可是人们往往对相邻近的 2 个音调的变化哼唱的比较准确，所以在相似度计算中旋律特征一所对应的系数选取较高，为 0.7。

如果窗口长度大于哼唱信号片段的音调个数，表明完全不匹配，则 S_t=0；如果窗口长度小于哼唱信号片段的音调个数，表明该哼唱信号片段为两句或两句以上的歌唱片段。

这种情况则进行如下判断计算：

提取下一窗口特征矩阵 W_2。设 N_1=W_1[1][1]为第一个窗口的长度，N_2=W_2[1][1]为第二个窗口的长度，N_m 为所求的哼唱信号片段中的音调个数。如果 $N_1+N_2>N_m$，则表明完全不匹配，S_t=0。如果 $N_1+N_2=N_m$，则进行如下计算：

$$M_{F3}[i]\begin{cases}M_{F3}[i] & 1<i\le N_1\\ M_{F3}[i]+M[1]-M[N_1+1] & N_1<i\end{cases} \quad (6)$$

$$S_t=\frac{\sum_{i=2}^{N_1}S_{F1}(i)+\sum_{i=2}^{N_2}S'_{F1}(i)}{10*(N_1+N_2-2)}*0.7+\frac{\sum_{i=2}^{N_1}S_{F2}(i)+\sum_{i=2}^{N_2}S'_{F2}(i)}{10*(N_1+N_2-2)}*0.3 \quad (7)$$

其中 $S'_{F1}(i)$，$S'_{F2}(i)$分别由式（8）和式（9）计算求得：

$$S'_{F1}(i)=\begin{cases}0, & W_1[2][i]=0 \ \ |M_{F2}[i+N_1]|>10\\ 10, & W_1[2][i]=0 \ \ 4\le|M_{F2}[i+N_1]|\le 10\\ 10, & |W_1[2][i]|=1 \ \ |M_{F2}[i+N_1]|>10\\ 20, & |W_1[2][i]|=1 \ \ 4\le|M_{F2}[i+N_1]|\le 10\\ 10, & |W_1[2][i]|=1 \ \ |M_{F2}[i+N_1]|<4\\ 0, & W_1[2][i]>1 \ \ |M_{F2}[i+N_1]|>10\\ 20, & W_1[2][i]>1 \ \ |M_{F2}[i+N_1]|<-10\\ 10, & W_1[2][i]>1 \ \ 4\le|M_{F2}[i+N_1]|\le 10\\ 0, & W_1[2][i]>1 \ \ |M_{F2}[i+N_1]|<4\\ 20, & W_1[2][i]<-1 \ \ |M_{F2}[i+N_1]|<4\\ 0, & W_1[2][i]<-1 \ \ |M_{F2}[i+N_1]|<-10\\ 10, & W_1[2][i]<-1 \ \ 4\le|M_{F2}[i+N_1]|\le 10\\ 0, & W_1[2][i]<-1 \ \ |M_{F2}[i+N_1]|<4\end{cases} \quad (8)$$

$$S'_{F2}(i)=\begin{cases}0,\ W_1[3][i]=0\ |M_{F3}[i+N_1]|>10\\10,\ W_1[3][i]=0\ 4\le|M_{F3}[i+N_1]|\le10\\10,|W_1[3][i]|=1\ |M_{F3}[i+N_1]|>10\\20,|W_1[3][i]|=1\ 4\le|M_{F3}[i+N_1]|\le10\\10,|W_1[3][i]|=1\ |M_{F3}[i+N_1]|<4\\0,\ W_1[3][i]>1\ |M_{F3}[i+N_1]|>10\\20,\ W_1[3][i]>1\ |M_{F3}[i+N_1]|<-10\\10,\ W_1[3][i]>1\ 4\le|M_{F3}[i+N_1]|\le10\\0,\ W_1[3][i]>1\ |M_{F3}[i+N_1]|<4\\20,\ W_1[3][i]<-1\ |M_{F3}[i+N_1]|<4\\0,\ W_1[3][i]<-1\ |M_{F3}[i+N_1]|<-10\\10,\ W_1[3][i]<-1\ 4\le|M_{F3}[i+N_1]|\le10\\0,\ W_1[3][i]<-1\ |M_{F3}[i+N_1]|<4\end{cases}\quad(9)$$

如果 $N_1+N_2<N_m$，则继续提取下一个窗口的特征矩阵，取其窗口长度，进行窗长比较，进而根据比较结果为 S_t 赋值。

第三步：将计算得到的 S_t 值赋给 S_{max}，S_{max} 用来记录整个音乐文件所有窗口中的最大相似度。

第四步：按顺序提取下一个窗口的特征矩阵。与第一步算法相同，根据所提取的窗口长度和哼唱信号片段窗口长度的比较情况，分别计算 S_t 值。

第五步：如果 $S_t>S_{max}$，则 S_t 值赋给 S_{max}。如按顺序还有窗口的特征矩阵未曾提取，则回到第四步。如果该音乐文件所有窗口的特征矩阵已经提取完毕，S_{max} 即为该音乐文件对应于哼唱信号片段的相似度值。

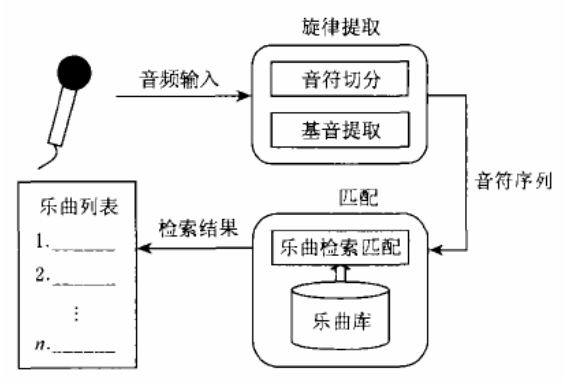
所有音乐文件计算完其对应于哼唱信号片段的 S_t 后，按照 S_t 的大小对音乐文件排序，返回相似度最高的几条音乐文件信息作为检索结果。

为适用于大型音乐数据库，选择了近似旋律匹配、DTW 方法和相似度匹配算法作实验进行性能比较。

3 性能评测实验

3.1 哼唱音乐检索测试平台

为了评测不同查询算法的性能，构建了一个音乐信息检索测试平台，采用 VC++6.0 开发环境进行程序开发，SQL-server 6.0 作为后台的数据库开发和管理工具，哼唱检索系统的工作流程如图 1：



Flag 1. Working Process of Internet Search Engine's Querying by Humming Music

图 1.网络哼唱检索系统工作流程

在对不同的音乐信息检索算法进行比较时，只需替换检索算法部分即可，这样就保证了对不同算法的测试均能在统一的测试数据集上完成。

3.2 乐曲库

采用从网络上免费下载的乐曲，构建测试乐曲库。由于从 MIDI 格式的文件中提取乐曲的音高、音长特征值序列非常便利，故本文所构建的音乐数据库中的乐曲均采用 MIDI 格式的文件。此外，下载的乐曲一般长度在 400~1000 个音符之间，为了使有些耗时的算法也能参与比较，将下载得到的每首乐曲作了裁剪，只裁成长度为 120 个音符的片断，将最后得到的 76,000 个 MIDI 片断构成测试集。

3.3 音频特征表示

在查询算法中，为了能消除用户哼唱在调式方面的错误，一般的检索算法采用乐曲的相对特征表示，即采用乐曲音符的音高差、音长比作为乐曲特征值序列。本文采取旋律轮廓的表示方法，即把音高差距量化为若干个等级（比如 U，D，S）：对于音长比，通常按照大于 1，小于 1，等于 1 分为 3 级，分别用 L、S、Q 表示；对于音高差，通常按照大于 0，小于 0，等于 0 分为 3 级，分别用 U、D、S 表示。若乐曲段的相对特征表示为（[-5，1.5），（2，1），（1，0.6）],则以 UDS，LSE 序列编码的三级旋律特征表示为（[D，L），（U，Q），（U，S）]。使用更高的精度会有助于检索的成功率，并缩短所需的查询时间，同时也能容纳用户的哼唱在音高差和音长比上的一般错误作者分析了 118 首流行音乐中音符音高差的分布，对用

978-1-935068-31-0 © 2010 SciRes.

户哼唱错误的分析中得知，用户的与旋律轮廓相同的错误中，90%以上只与正确的音高差相差 1~2 个半音。因此，将音高差分为 5 级更为合理，使其既能更精确地表示用户的哼唱，又能在多数情况下包容用户与旋律轮廓方向相同的错误。将旋律轮廓按照 5 级音高差，并不对音长做更细化的分级。

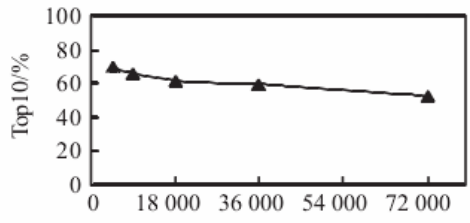
4 实验结果

4.1 三种算法的查询命中率比较

在 3.1 节讨论的测试平台上，对近似旋律匹配算法，DTW 和相似度方法分别作了数据量为 18,000 首、36,000 首、54,000 首和 72,000 首乐曲的 5 种情况下的性能测试实验。测试结果如下：

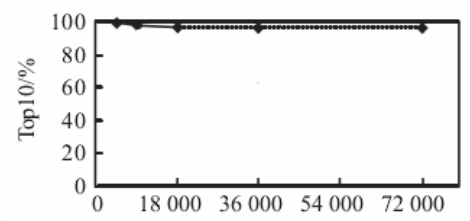
我们通过 10 位测试人员，7 男 3 女，录制了 62 段录音进行实验。10 位测试人员均无音乐专业背景，以“lailai”、“lala”和“dada”声哼唱乐曲中的任意一段旋律，长度从 30 到 120 个音符不等。实验查询命中率如下图所示

从图 2、图 3 和图 4 中可以看出，3 种算法中，DTW 算法前 3 位与前 10 位的查询命中率都是最高；且当数据库中所含的乐曲数增多时，DTW 算法的查询命中率下降最慢，在数据量为 18,000 与数据量在 72,000 时查询的命中率相差很少，而数据量在 36,000 与数据量为 72,000 时几乎是一样的。说明 DTW 算法的扩展性在 3 种算法当中是最好的。



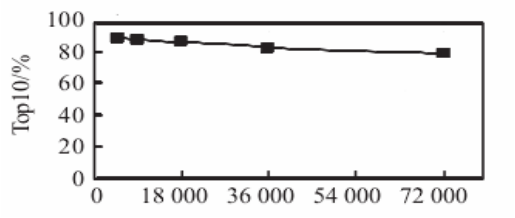
Flag 2. Approximation Melody Algorithm

图 2.近似旋律匹配算法



Flag 3. DWT Algorithm

图 3.DTW 算法



Flag 4. Similarity Matching Algorithm

图 4.相似度匹配算法

4.2 三种算法的查询相应时间比较

衡量音乐信息检索算法性能的另一个重要指标是查询速度，对 3 种算法在不同规模的数据库上进行测试时所需的查询时间进行了记录。结果发现，算法在查询速度上要大大优于其余两种算法。以在大小为 72,000 首乐曲的数据库上所做的测试为例，相似度匹配算法平均每次查询所需的时间仅为 10ms。而且，随着数据量的增加，相似度匹配算法查询所需时间的增加比其余两种方法缓慢得多。相比之下，近似旋律匹配算法和动态时间规整算法平均每次查询所需的时间分别为 96s 和 125s。考虑到设计评测数据集时，将每首歌的音符数只截取到 120 个音符，在实际情况下近似旋律匹配算法和动态时间规整算法平均每次查询所需的时间应是实验时间的 3 倍以上。由于网络用户通常希望在哼唱结束后较短的时间内就得到查询结果，因此综合考虑查询命中率和查询速度两个指标，作者认为相似度匹配算法是三者中最优的，最适合网络音乐哼唱检索引擎。

5 结论

实验结果表明，动态时间规整算法在查询命中率与扩展性两个方面都具有最好的性能，当音乐数据库的数据量增大时，动态时间规整算法甚至可以保证查询的前 10 位命中率趋于稳定。但是，由于动态时间规整算法的查询速度最慢，因此在实用性方面，尤其是大数据量条件下，网络音乐搜索应用这个方面受到了限制，毕竟网络用户希望在最短的时间内得到可用的信息；近似旋律匹配算法则在查询命中率与查询速度两个方面都不能令人满意；相似度匹配算法查询速度快，查询命中率较高，虽然扩展性不是很好，特别是当乐曲库的数据量增大时，查询前十位命中率下降的比较明显，但是都基本能保持在 80%以上。对于网络音乐哼唱检索，这个命中率已经可以接受，而且其查询速度快的特点也会增加网络用户的认可度。因此，

作者认为相似度匹配算法是适合于网络音乐哼唱检索的一种算法。

作为相似度匹配算法而言，作者实现的系统还有一个缺陷，就是只建立了查询输入错误模型，并没有根据此模型构造查询输入集。而这一模型也是有局限性，不同年龄段、不同职业的人没有参加测试，哼唱的歌曲的类别非常少，今后将扩大参加哼唱输入测试的人员类别和哼唱歌曲的类别，建立更具代表性的用户错误模型，使算法的比较具有更坚实的基础。

References (参考文献)

[1] Jia-Lien Hsu,Arbee LP Chen.The effectiveness study of various music F information retrieval approaches[C]//Proceedings of the Eleventh International Conference on Information and Knowledge Management

[2] Roger B Dannenberg,William P Birmingham.The MUSART testbed forquery-by-humming evaluation[C]//Proceedings of 4th International Symposium on Music Information Retrieval.Baltimore,Maryland,USA:The Johns Hopkins University,2003:268.

[3] Hoos H.Rentz K.Gorg M.GUIDO/MIR. An Experimental Musical Information Retrieval System based on GUIDO Music Notation[EB/OL].[2005-05-28]http://www.midi.org/about-midi/specshome.shtml.

978-1-935068-31-0 © 2010 SciRes.