建立分段MP3歌曲重複樣型之分類索引結構
Constructing Classified Index Structures for Repeated Patterns of Segmented MP3 Songs

林朝興
電腦工程博士 助教授
南台科技大學資訊管理所
710 台南縣永康市南台街一號
Email: mikelin@mail.stut.edu.tw
06-2533131 ext. 4311

王怡君
資訊管理碩士 研發工程師
東元電機 綜合研究所
台北市南港區園區街3號F棟4F
Email: novel@msrg.mis.stut.edu.tw
02-2655-8255 #2109

摘 要

近年來由於MP3音樂壓縮技術的快速演進，建立一大型音樂資料庫提供隨選音樂(Music-on-Demand, MoD)，已經變的真實可行。直接以外部資料(external information)例如歌名或歌手名稱，搜尋歌曲雖然方便，但對於不知名或不記得曲名的歌曲，往往無法提供有效的查詢。因此，藉由基於內容擷取(content-based retrieval)技術，使用者可以透過麥克風經由直接哼唱(humming)方式，以旋律方式查詢歌曲。隨著音樂資料庫容量的成長，查詢的等待時間也相對增加。有鑑於此，不少學者提出各種方法來加快查詢比對的速度，最常使用的方法是將音樂中重覆片段(repeated patterns)，做爲關鍵特徵(key features)索引來代表整首歌曲，以改善搜尋速度。但由於重覆片段的起迄往往落於樂句之間，不符合一般哼唱習慣，造成比對上的困難。

本研究論文首先提出動態擷取(dynamic retrieval)MP3關鍵樂句(key phrase)的方法。其包含二階段，音樂內涵分段階段包括去除背景音樂及動態分段，資料正規劃階段包括音名轉換及擷取關鍵樂句。此方法以少量的關鍵樂句代表整首歌曲，可有效地減少搜尋比對資料量。此外，本論文另提出Music-Trie索引結構，以建置所有關鍵樂句索引來快速搜尋歌曲。其提供類似字典的直接索引比對，以廉價的儲存空間提供快速的查詢。文中我們並探討Music-Trie索引結構在深度(depth)及廣度(breadth)的優化(optimization)。模擬實驗結果顯示，相較於循序(sequential)及分類循序(classified sequential)索引結構，不管是真實或人造歌曲測試資料集上，Music-Trie在索引的建立及需求空間上效能稍差，但其搜尋時間幾乎是常數(O(1))，不會隨著音樂資料庫中MP3歌曲的增加而增加。

關鍵字：音樂索引結構、Music-Trie、重複樣型、分段、MP3

Abstract

With the rapid advances of the MP3 compression technology recently, building a huge scale of music database to provide Music-on-Demand (MoD) services has become feasible. Although it may be convenient to directly query a song by its external information, such as the name of the singer or the song, for an unknown singer or a forgotten name of a song, this approach cannot be considered as an effective approach. Therefore, by the content-based retrieval technology, users are capable of directly "humming" the melody of a song to microphone as an input to issue a query. However, as the number of MP3 songs in music database grows, the searching time for a particular song is inevitably increased. Many approaches were proposed to expedite such a search process. The most common one is to define repeated patterns in a song as key features for indexing so that the searching time can be effectively shortened. Nevertheless, the start and the end of a repeated pattern often do not fall into the boundaries of phrases. It does not conform to the behavior of normal humming and consequently is difficult to have an effective search.

In this paper, we first propose the approach of dynamic retrieval for MP3 key phrases for building indices. It consists of two phases, the segmentation of music contents phase which includes eliminating background music and dynamic segmentation, and the normalization of representation phase which includes the transformation of notes and the retrieval of key phrases. This approach defines few key phrases to represent a song and can affectively reduce the number of comparison in searching. In addition, we also propose the index structure of Music-Trie to build the indices of key phrases for promptly searching a particular song. It provides direct index comparisons like dictionary by exchanging cheap storage space for prompt search. The optimization of Music-Trie index structure in terms depth and breadth is also discussed in the paper. The experimental results show that compare to sequential and classified sequential index structures, no matter the testing data set is genuine or artificial, Music-Trie performs slightly worse in terms of the time of building indices and the storage requirement of indices. However, with Music-Trie, it is worth noting that the search time for a particular song is almost constant (O(1)) which is independent of the number of MP3 songs in music database.

Keywords：music index structure, Music-Trie, repeated patterns, segmentation, MP3

1. 研究背景

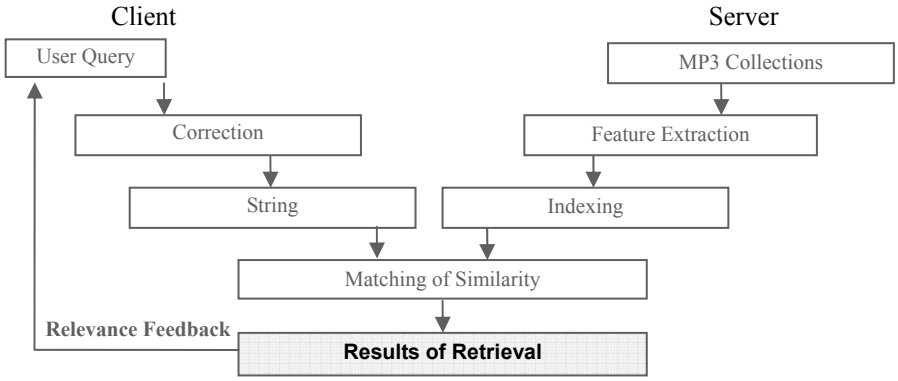
MP3以極高的壓縮比與極小的失真，成爲目前最受歡迎的數位音樂壓縮格式。隨著數位科技的發展與多媒體網站的需求，促使音樂數位化，因此MP3廣泛應用並且大量在網路上流傳。人們利用各大入口網站提供的搜尋引擎，找到想要的MP3歌曲。然而對於不知名或不記得曲名的歌曲，再強大的搜尋引擎也無法提供任何幫助。若能提供一個可以描述歌曲旋律的介面，才能改善且滿足人們的需求。

要提供一個描述歌曲旋律的介面，首先要了解歌曲的特徵。除了歌名、歌手、專輯名稱、年代等文字型態的外部資訊之外，近年來有許多研究以Melody、Pitch、Rhythm、

Interval、Chord and Contour等音樂內容特徵[1,17,4,14,113]，建立爲字串索引來幫助檢索[21,5,19]。

以Content-Based Retrieval(CBR)的方式[21,25,18,6,7,14]，必須考慮取出的特徵值是否符合人類知覺。Y.H. Tseng以五線譜、簡譜等樂理格式提供查詢介面[35]。Wold, E.，T. Blum等人，則是以響度(Loudness)、亮度(Brightness)、頻寬(Bandwidth)及音高(Pitch)等特徵值描述音樂內容[38]。這些方法都清楚描述音樂的內容，然而大部分使用者沒有這些專業知識的背景，查詢變得困難[31]。

後來學者們從使用者的角度出發，提出人類知覺哼唱方式(Query-by-Humming)[8,1,5,6,15,16,38]，讓使用者透過麥克風直接哼唱查詢。然而，大多數的使用者哼唱不易正確，那是因爲在哼唱時，會不自覺的會將歌曲調整成適合自己音域範圍，並且不會完全地將每個音都哼唱出來，使得哼唱的片段與原始真實的音樂有些距離。因此系統在檢索比對時，會因爲使用者因素而找不到使用者哼唱查詢的歌曲。目前有兩個方法解決此問題，一種是提供範例查詢(Query by Example)[25,34,35]，一種是將使用者哼唱的片段先做校正處理後(Correction)，再提供相似度的比對[18,13]。



圖一：檢索模式

圖一表示目前在哼唱查詢上的理想化檢索模式。將搜集音樂資料(WAVE、MIDI、MP3)針對內容擷取特徵值，透過字串方式記錄並建置爲該音樂資料的關鍵索引(Indexing)。當使用者輸入哼唱的片段後，首先必須將使用者哼唱片段進行校正處理，再與資料庫中的索引做相似度比對(Matching of Similarity)。最後將得到檢索結果並回饋給使用者[37]。

除了考慮使用者哼唱正確性的問題外，所有的資料庫都會面臨成長帶來的負面影響：檢索比對歌曲量增加，查詢的等待時間將相對增加[22,13,10,11]。有許多學者提出減少比對的方法，包括Classification、Non-Trivial Repeating Pattern and Segmentation等方式。在眾多分類的方法中，以音樂曲風(Music genres)[20,36,38]與歌手(Artists)[23]的分類最常見。根據使用者哼唱內容來檢索某一分類(Classification)[23,33,38]，可以有效減少比對次數來提升查詢的效率。但音樂曲風多變，或者混合兩種以上的編曲，就無法明確的分類。

叢集(Clustering)可以解決這個問題，將歌曲的特徵值以多維度方式呈現[28]，再計算鄰近歌曲的歐幾里德距離。然而，若低沉聲音的使用者哼唱由音色清亮的歌手所演唱的歌曲，會造成系統判斷錯誤而增加檢索的等待時間。因此分類或叢集法大都應用在歌

曲推薦[3]，系統根據使用者點選的歌曲或存取記錄，分析並且推薦與使用者興趣相近的歌曲叢集。

只比對歌曲中的重覆片段可以加快查詢。作曲者爲了讓歌曲更受歡迎與更容易哼唱，往往刻意強調且重覆的旋律。許多音樂理論和人類知覺上的研究學者也一致認爲，讓消費者印象深刻的旋律並非整首歌，而是樂曲中一再重覆的片段[1,17]。因此只記錄歌曲中非不重要重覆片段來代表整首歌可以降低比對資料量。但音頻資料是連續且龐大的資料集合，無論是Suffix Tree[32,19]、Correlative Matrix[10,29,30]、RP-Tree[22,11]等等方法，或是Tseng, Y. H. [35]提出Data mining中類似Apriori的方法，在找尋重覆片段上的時間複雜度都十分高昂。

分段方式可縮短音樂資料的長度，改善在找尋非不重要重覆片段上的缺點。因此部分學者主張分段來簡化分析MP3內容索引的複雜度[2,36]。然而目前的技術只能採用固定時間長度分段[2,28,36]，這樣方式有一個重大的缺失，會使完整樂句被分割在不同的段落中，造成歌曲結構的誤判。Kuo and Yao[24,26]取出歌曲中有規則的人聲的部分，再採用先斷音(phoneme segmentation)再重組樂句(phrased)的方式改善固定分段的缺失，但這樣的方式將受限於樂句較工整的歌曲。Yu[27]提出的改善方法，就是將歌曲消減背景音樂以降低分段時的干擾。消減背景音樂能適合任何有人聲的歌曲，但是分段後的音樂存在許多重覆的片段，而且當歌曲數量快速成長，比對的時間依舊增加。

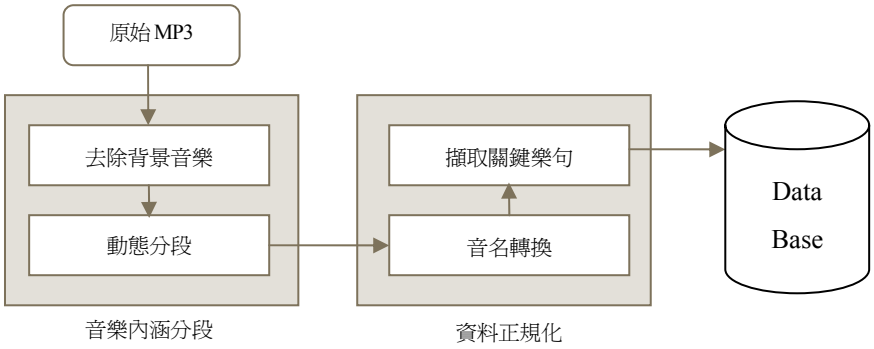
綜合以上學者的研究，我們得知在多媒體音樂資料庫的哼唱查詢上，普遍受到二個因素的影響：一個是使用者哼唱內容與真實歌曲的差距，另一個是隨著資料庫的成長，比對資料量增加，使查詢等待時間相對延長。因此，本研究結合分段、重覆片段與分類等方法致力改善。首先，將歌曲動態分段來降低比對與去除重覆片段時的複雜度，改善固定分段的缺失，並適用任何長短句型的歌曲。分段後的樂句只記錄音的變化，改善使用者哼唱的不正確。接下將樂句去除重覆後，建置在我們提出的Music-Trie索引結構中，查詢時間與索引數量無關，提供快速穩定的檢索。

在本文的其他章節組織如下。在第二節中我們提出動態擷取MP3關鍵樂句的方法。第三節中我們討論Music-Trie索引結構和資料處理。第四節討論Music-Trie索引樹深度與廣度的優化。第五節說明實驗環境及分析實驗結果，討論Music-Trie的效能。最後，第六節爲結論及未來研究。

2. 動態擷取 MP3 關鍵樂句

我們提出一個針對MP3音樂內容爲基礎的分段方法，取名爲動態擷取(dynamic retrieval)MP3關鍵樂句(key phrases)。主要目的要找出代表歌曲的關鍵樂句，並加以建置在資料庫中提供查詢與比對。歌曲中的樂句有許多是重覆出現的，這些重覆的樂句不需要重覆的記錄與比對。爲了去除這些重覆的樂句，我們利用動態分段來降低複雜度。

爲了達到動態分段，首先必需去除背景音樂，讓電腦能夠從音頻內容自動判斷分段的地方，改善傳統以固定分段的缺失。分段後的樂句經由音名轉換和擷取關鍵樂句兩步驟，將音頻資料轉換成字串方式存在資料庫中。整個系統流程如圖二，分爲音樂內涵分段、資料正規化兩大部分，詳述如下：



圖二：動態擷取 MP3 關鍵樂句系統流程圖

2.1 音樂內涵分段

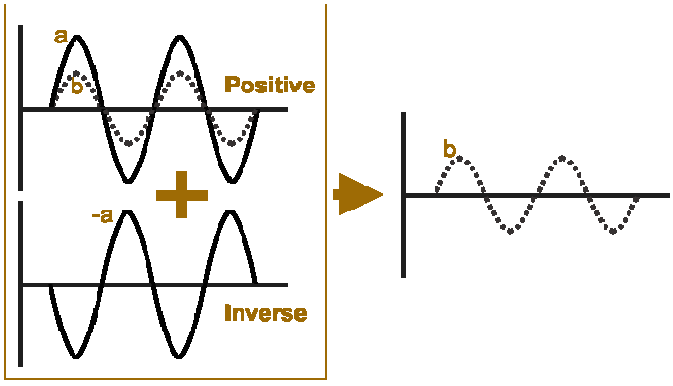
作曲者爲了讓歌曲更受歡迎，會刻意重覆歌曲中某些旋律片段，來加深聽眾的印象。因此，音樂中重覆的片段是音樂結構中普遍性的特徵，也是整首歌中人們最熟悉並且可能哼唱的片段。利用這個特性，去除歌曲中重覆片段可節省儲存空間，也可以減少比對資料，幫助減少查詢的等待時間。

音頻資料是連續且龐大的資料集合，找尋重覆片段時，普遍受到音樂資料長度影響找尋效能[22,32,10,11,19,35,29,30]。我們利用動態分段來縮減音樂資料長度並降低複雜度，以歌手唱歌換氣間做爲分段的地方。直接從 MP3 解碼中無法找出歌手換氣的地方，因爲 MP3 將所有聲音輸出爲單一頻道，人的耳朵雖然聽得出哪裡是斷句的地方，但電腦無法判斷。我們分析歌手換氣的時間點，沒有歌手的聲音但存在背景音樂。因此，若去除背景音樂，那麼換氣時間點將會沒有任何能量存在，就能提供電腦自動化斷句依據。

去除背景音樂，我們使用波形反相疊加原理。如圖三，實線a是一個正向的波，實線-a波是實線a波的反向波，波形相同但相位相反。當兩者疊加後將會互相抵消，剩下虛線的b波。利用這個物理特性，我們將左右聲道之一反相後疊加到另外一個聲道，兩聲道中聲場較窄的人聲被抵消了，剩下背景音樂。然而背景音樂是連續不存在能量中斷處，無法提供電腦自動化斷句依據。因此，再使用波形反相疊加原理，將背景音樂反相疊加到原始MP3中，就會得到去除背景音樂的人聲資料。透過去除背景音樂模組，將消除無人聲的前奏、間奏與尾奏等片段。

接下來，動態分段模組從MP3解碼過程中IMDCT（Inverse Modified Discrete Cosine Transform）中計算每個框架的多相位濾波器係數向量。再加總得到框架能量，計算出能量係數。透過斷句模組將樂句分割提供電腦自動化斷句的依據。利用過濾背景音樂與歌手換氣處分段，突破一般採用固定時間分段的限制，適用於任何長短句型的歌曲。

範例歌曲經由音樂內涵分段後，資料量降爲原始 MP3 音樂的 63%；此外，實驗分析百首真實歌曲平均約縮減四分之一的資料量[37]。

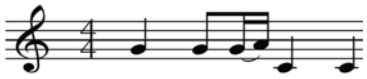


圖三：波形反相疊加

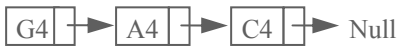
2.2 資料正規化

完成音樂內涵分段後，必需將已分段的音頻片段透過音名轉換模組，轉換成字串形式方便記錄與比對。首先透過ISO/IEC 11172-3所定義的Frequencies、critical band rates and absolute threshold[12]，將聲音能量局部最大值轉換成頻率。接下來再依據1975年國際標準化組織(IS0)定義，將取得的頻率轉換成標準八度音程(an octave)（鋼琴上從最低音數上來的第四個A（A4，440Hz）），頻率以接近爲主。此外，取樣時間以該歌曲中所有音符時間長度的最大公因數，才能完整記錄旋律變化。以圖四旋律片段爲例，16分音符爲所有音符中最大取樣單位，也就是每 *1/4* 拍必需取樣一次，才能完整記錄旋律輪廓。轉換成音名字串的樂句集合爲G4G4G4G4G4G4G4A4C4C4C4C4C4C4C4C4。

由於使用者沒有經過專業歌手訓練，進行哼唱查詢時拍子會忽快忽慢不容易正確，因此系統在相似度比對（Matching of Similarity）的正確性降低。若能忽略使用者哼唱與歌曲雙方的速度，只保留樂句變化情況，將可提高使用者哼唱比對的相似度。將圖四音名字串的樂句集合，記錄成{(G4),(A4),(C4)}，不但減少Node的個數與儲存空間，同時調整樂句速度，表示爲圖五。



圖四：16 分音符爲最大取樣時間



圖五：圖四表示 List

分段與轉換後相似樂句字串中，存在許多重覆樂句，透過擷取關鍵樂句模組去除重覆樂句。首先我們將已分段後的樂句依序編號，分別爲T_1~T_n，n爲歌曲分段後的句數。接著定義一個相鄰串列（Adjacency List）記錄歌曲中不重覆的樂句。相鄰串列裡的每個樂句串列K_j，都有一個標頭節點，包含音名個數與重覆出現次數。將該首歌曲的所有樂句依序放入相鄰串列中，去除重覆的演算法如下：

- ●若相鄰串列爲空集合，表示相鄰串列中無任何樂句，新增 T_1 到相鄰串列中。
- ●若相鄰串列不爲空集合，T_i 的音名個數與相鄰串列中 K_j 的音名個數無相同者，新增 T_i 到相鄰串列中。
- ●若相鄰串列不爲空集合，T_i 的音名個數與相鄰串列中 K_j 的音名個數有相同者，比較串列內容。

A. 若 T_i 與相鄰串列中任一串列 K_j 完全相同，捨棄 T_i，K_j 標頭節點中次數加 1。

B. 若 T_i 與相鄰串列中任一串列 K_j 無相同，新增 T_i 到相鄰串列中。

以範例歌曲爲例，經由音樂內涵式分段後將歌曲分割爲80句相似樂句，去除重覆後的關鍵樂句只有13句[37]。根據百首真實歌曲實驗，平均每首歌曲的關鍵索引大約有18.25句。另外，假設範例歌曲去除背景音樂後轉換成音名字串的資料量爲100%，而關鍵樂句爲原始音樂字串的6%，百首真實歌曲的關鍵樂句爲其原始音樂字串10-15%[37]。

3. Music-Trie 索引結構(Index Structure)

經由動態分段和擷取轉換後產生的關鍵樂句，將被建置在資料庫中。然而，資料庫查詢時間，與資料的數量有關。被建置的歌曲數量快速成長，比對的關鍵樂句相對增加，查詢的等待時間會延長。因此，我們提出Music-Trie索引結構[9]改善資料庫查詢的缺失，提供與歌曲數量無關的快速查詢。

我們將歌曲轉換和去除重覆後的關鍵樂句集合repeat[max_AL]，依序建置在Music-Trie索引樹上。Music-Trie索引結構適合不等長的關鍵樂句，並且保留其音名字組的順序關係。首先，我們定義Music-Trie索引結構：

1. 分支數 $M \geq 2$ 的階層樹。
2. 包含兩種節點，分別爲分支節點和葉節點。
3. 將所有頻率情況建置爲一個對照表，提供搜尋時，直接找到下一層的對應路徑。

對照表將所有頻率情況依大小給予順序編號，其中0代表空白字元，1~52是音名字組所有可能情況（A0=1,B0=2,...C8=52），依頻率大小排列組合而成，頻率的範圍以樂器可以發出的頻率爲極限。樂器中以鋼琴發出的頻率範圍最廣，範圍大約從27.5～4186Hz，轉換爲標準八度音程，一共52個情況，被依序記錄在對照表中1到52個位置，與分支節點相對。

分支節點是一個一維整數陣列，陣列內容是用來存放下一個節點的位址，陣列範圍從0~52。葉節點是用來儲存關鍵樂句與其歌曲編號。以C++資料結構方式宣告，使用union方式實現Music-Trie索引結構，如圖六。接下來我們針對Music-Trie索引結構在新增、查詢與刪除等資料處理上，分別在3.1、3.2與3.3節裡詳盡的描述。

```
/* Music-Trie Index Structure */

typedef char ELEM;
typedef ELEM *Attr;
const int base=52;
const char start='A0';
enum nodetype{leafnode, internal};
enum symbol {A0=1,B0,C1,D1,E1,F1,G1,A1,B1,C2,D2,E2,F2,G2,A2,B2,C3,D3,E3,F3,G3,A3,B3,
             C4,D4,E4,F4,G4,A4,B4,C5,D5,E5,F5,G5,A5,B5,C6,D6,E6,F6,G6,A6,B6,C7,D7,E7,
             F7,G7,A7,B7,C8}          //Musical Alphabet
class node {                          //Generic Node Class
   public:
      nodetype mytype;
      char symbol;
      int flag;                       // 1 代表是葉節點,0 代表分支節點
   union {
      struct{                         //Structure For Internal Node
            node *ptr[base+1];
            int layer;                //In Layer
      }intl;
      struct{
            ELEM *data;               //歌曲編號
            Attr attribute;           //葉節點內容
      }leaf;
   };
};
```

圖六：使用 c++，建立 Music-Trie 索引結構

3.1 在 Music-Trie 索引樹中新增插入（Insert）關鍵樂句

Music-Trie索引結構爲樹狀結構，必需產生一個根節點root，提供所有相似樂句插入、刪除與查詢。完成根節點的建置後，欲插入的關鍵樂句必需先透過對照表，由左而右依序轉換成相對的編號。指標σ沿著轉換後的音名字組編號i和順序j來決定分支路徑與階層δ_j。插入規則如下：

1. 若σ→intl.ptr[i] == NULL，表示該關鍵樂句不存在 Music-Trie 索引樹中。產生一個葉節點，將關鍵樂句與歌曲編號放入葉節點中，最後將 intl.ptr[i]指到該葉節點。
2. 若σ→intl.ptr[i] != NULL，且σ→intl.ptr [i]→flag == 0，表示下一個節點爲分支節點。改變指標σ指到下一個分支節點，$\sigma = \sigma$→intl.ptr[i]。再由下一個音名字組編號決定下一層陣列位置。
3. 若σ→intl.ptr[i] != NULL，且σ→intl.ptr[i]→flag == 1，表示下一個節點爲葉節點。兩種可能情況如下：
   a. 若插入關鍵樂句與該葉節點相等，只需更新該葉節點的歌曲編號。
   b. 若插入關鍵樂句與該葉節點不相等，由左而右比較關鍵樂句和葉節點差異；相等時新增不存在的分支節點並建立連結，直到不同爲止。新增開始分歧的分支節點，同時建立連結。重新尋找關鍵樂句與該葉節點插入的位置。
4. 若σ→intl.ptr[i] != NULL，且沒有下一個音名字組編號提供路徑，將該關鍵樂句最後加上一個空白字元。重新尋找關鍵樂句插入的位置。

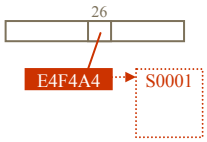
假設連續插入關鍵樂句，依序爲"E4F4A4"、"E4F4A4"、"E4F4G4"、"E4F4"。首先新增一個根節點root，欲插入的關鍵樂句必需透過對照表轉換。

第一個插入的關鍵樂句爲"E4F4A4"，轉換成音名字組編號爲(26,27,29)。指標σ→intl.ptr[26] == NULL，符合規則1。產生一個葉節點，放入該關鍵樂句的內容與歌曲編號，然後建立連結，σ→intl.ptr[26]→E4F4A4，如圖七。

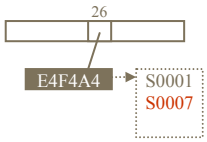
第二個插入的關鍵樂句爲"E4F4A4"，轉換成音名字組編號爲(26,27,29)。指標σ→intl.ptr[26] != NULL，且σ→intl.ptr[i]→flag == 1。比對已存在的葉節點，符合規則3.a。只需更新歌曲編號，如圖八，S0001與S0007的歌曲中都有相同的關鍵樂句。

第三個插入的關鍵樂句爲"E4F4G4"，轉換成音名字組編號爲(26,27,28)。指標σ→intl.ptr[26] != NULL，且σ→intl.ptr[26]→flag == 1。比對已存在的葉節點，符合規則3.b。關鍵樂句與葉節點存在｛26,27｝重覆，建立不存在的δ_2與開始分歧的δ_3，最後將原始的葉節點與關鍵樂句加入δ_3，如圖九。

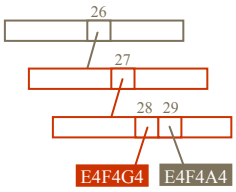
第四個插入的關鍵樂句爲"E4F4"，轉換成音名字組編號爲(26,27)。指標σ→intl.ptr[26] != NULL，且σ→intl.ptr[26]→flag == 0，符合規則2。更改指標σ→intl.ptr[26]→intl.ptr[27] != NULL，然而已經沒有下一組音名字組編號提供分支路徑，而且σ→intl.ptr[26]→intl.ptr[27]不能同時指到分支節點與關鍵樂句"E4F4"，符合規則4。因此將x4最後加上一個空白字元，x4依據對照表內容被轉換成{26,27,0}。最後關鍵樂句被加入σ→intl.ptr[26] →intl.ptr[27]→intl.ptr[0]→"E4F4"，如圖十。



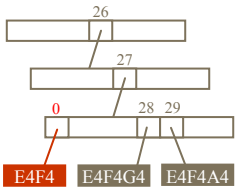
圖七：插入 E4F4A4



圖八：插入 E4F4A4



圖九：插入 E4F4G4



圖十：插入 E4F4

3.2 在 Music-Trie 索引樹中查詢（Query）關鍵樂句

Music-Trie索引結構利用對照表的方式，在每一層搜尋時提供直接定位方式快速找到下一層的對應路徑。將所有頻率情況依大小記錄在對照表中，使得在每一層搜尋時不必再循序地搜尋。當使用者進行哼唱查詢時，首先將哼唱片段的音頻內容轉換成字串，然後在字串的內容最後加上一個空白字元。再依照Music-Trie索引結構中定義的對照表內容，由左而右依序轉換成音名字組的相對編號。

指標σ沿著音名字組編號i和順序j來決定分支路徑與階層δ_j。根據以下的規則，找出與哼唱片段符合的歌曲。查詢規則如下：

1. 若σ→intl.ptr[i] != NULL，且σ→intl.ptr [i]→flag == 0，表示下一個節點爲分支節點。改變指標σ指到下一個分支節點，σ = σ→intl.ptr[i]。再由下一個音名字組編號決定下一層陣列位置。
2. 若σ→intl.ptr[i] != NULL，且σ→intl.ptr [i]→flag == 1，表示下一個節點爲葉節點。兩種可能情況如下：
   a. 若插入關鍵樂句與該葉節點相等，傳回與使用者哼唱片段相似地歌曲編號。

b. 若插入關鍵樂句與該葉節點不相等，由關鍵樂句下一個音名字組編號決定下一層陣列位置。

3. 若σ→intl.ptr[i] == NULL，且沒有下一個音名字組編號提供路徑，表示使用者哼唱的相似樂句不存在 Music-Trie 索引樹中。

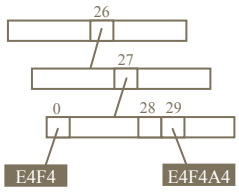
3.3 在 Music-Trie 索引樹中刪除（Delete）關鍵樂句

欲刪除的關鍵樂句，必需先找到該關鍵樂句在Music-Trie索引樹中的位置，並透過對照表內容轉換成音名字組的相對編號。由左而右依序沿著音名字組編號i和順序j來決定分支路徑與階層δ_j，直到找到葉節點*Leaf*。刪除規則如下：

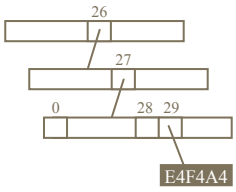
1. 若σ→intl.ptr[i]→*Leaf*，σ→intl.ptr[i]= NULL。
2. 若該分支節點以下只包含一個葉節點，將該葉節點向上移動一層，刪除該分支節點。

欲刪除圖十葉節點E4F4G4，透過對照表轉換成{26,27,28,0}。接下來由左而右依序沿著音名字組的相對編號所決定的分支路徑搜尋。在δ_3找到該葉節點intl.ptr[28]→E4F4G4，將intl.ptr[28]→NULL刪除該葉節點。intl.ptr[28]所在分支節點包含二個以上的葉節點，完成刪除E4F4G4動作，如圖十一。

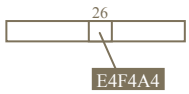
若刪除圖十一葉節點E4F4，除了將δ_3中intl.ptr[0]→NULL（圖十二），δ_3與δ_2都沒有符合≥2個葉節點原則，因此將葉節點E4F4A4往上移動到δ_1，刪除δ_3與δ_2（圖十三）。



圖十一：刪除 E4F4G4



圖十二：刪除 E4F4



圖十三：完成刪除 E4F4

4. 優化(Optimization) of Music-Trie

在Music-Trie索引樹中搜尋時受階層數影響，階層數與關鍵樂句中變化的情況成正比，並且小於等於關鍵樂句變化情況。4.1節以實例分析每句樂句的變化情況，Music-Trie索引樹的深度控制在一個常數範圍。4.2節考慮Music-Trie的廣度，廣度受頻率範圍影響，以實例說明歌曲頻率範圍，減少分支節點陣列大小，降低空間的使用。

4.1 Music-Trie Index Structure 深度(depth)優化

一般流行音樂存在休止符提供歌手換氣，同時也讓歌手藉著換氣間所形成的樂句，去詮釋歌曲所呈現的意境，其作用如同文章中的逗號一樣，是作者語文呼吸一部分。樂句太長或變化太大會讓人無法呼吸。我們分析「HitFM 2002年度百首單曲」，大致分爲兩種情況，一種是樂句很長，但變化很少；另一種是樂句變化很多，但樂句不長。樂句的變化會影響Music-Trie的搜尋效能，以下我們以實驗分析歌曲樂句的變化情況。

以「HitFM 2002 年度百首單曲」爲實驗資料來源。該百首單曲由 Hit Fm 聯播網在 2002 年年底所舉辦之「網友票選 2002 年 TOP 100 單曲」活動所產生，其中包括中文、英文、日文、韓文與男女對唱。圖十四將該百首單曲依歌手換氣處分段後的樂句，呈現

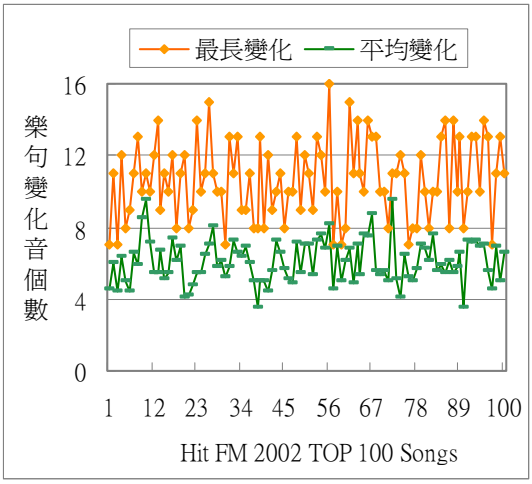
每首歌曲中樂句平均與最多變化情況。橫軸爲百首單曲，縱軸爲樂句變化個數。圖十四的第一首歌曲爲例，該首歌曲樂句的變化情況平均爲 4.52，而樂句最多變化情況爲 7[37]。

圖十四也顯示該百首單曲中變化情況最多爲16，影響Music-Trie搜尋時間被控制在16層內。樂句和樂句之間的相似度愈大，則需要愈多的分支節點來提供分歧。最差情況爲兩句以上樂句變化爲16的樂句，由左而右比較，有15個音皆相同，Music-Trie索引樹的深度將爲16層。我們將百首單曲的關鍵樂句全部建置在Music-Trie索引樹中，實驗結果階層數爲12，小於樂句最多變化情況[37]。

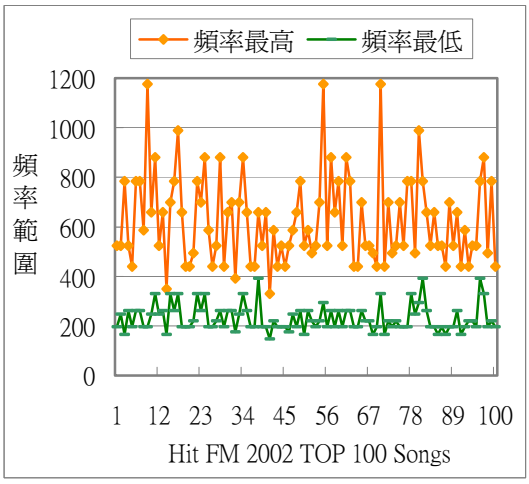
4.2 Music-Trie Index Structure 廣度(breadth)優化

Music-Trie索引結構以增加分支節點的方式來提供穩定快速的搜尋回饋，相對付出較高昂的空間代價。分支節點的陣列大小，以鋼琴發出頻率範圍，扣除半音後，以52個全音和一個空白共53種情況組成而成。要有效節省使用空間，必需要能縮減分支節點的陣列大小，也就是頻率的範圍。實驗以"HitFM 2002年度百首單曲"分析真實歌曲的頻率範圍。

圖十五將該百首單曲去除背景音樂並依照歌手換氣處分段後，呈現人聲頻率範圍，介於147~1175Hz之間，大約是D3~D6共22種情況。由此我們可以將分支節點陣列大小由原始53(52+1)縮減至23(22+1)，縮減50%。優勢在於我們巧妙的利用過濾背景音樂的方式，不但提供電腦做自動化斷句，而且沒有背景音樂的人聲頻率範圍更窄，被擷取出來的相似樂句都可以完全被建立在優化後的Music-Trie索引結構中。



圖十四：百首單曲平均樂句變化與最長變化



圖十五：百首單曲中人聲音域範圍

5. 模擬實驗及分析

我們以百首真實歌曲與十萬首人造歌曲所產生的關鍵相似樂句，分別建置在循序索引結構、分類循序索引結構與Music-Trie索引結構中。比較這三個方法的查詢索引等待時間、建置索引的時間與建置索引結構所需的儲存空間等三個因素上的優劣。執行環境在Solaris作業系平台、8 顆 UltraSPARC III 1050MHZ CPU、16G RAM。

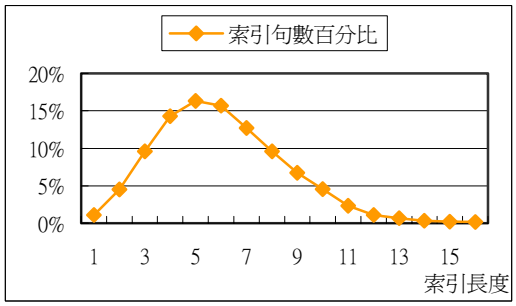
關鍵索引的資料來源，我們將百首真實歌曲過濾背景音樂，分段與去除重覆平均每首歌曲約有18.85句關鍵索引，去除歌曲間重覆的相似樂句後，約有1600句相似樂句。

真實歌曲樂句變化情況呈現常態分配，如圖十六，依照比例配置亂數產生十萬首人造歌曲，共200萬筆人造相似樂句。去除歌曲間重覆相似樂句後約160萬句關鍵索引，如Table 1所示。

比較的方法包括循序索引結構、分類循序索引結構與Music-Trie索引結構。循序索引結構是將關鍵索引全部建置同一分類中，查詢時針對該分類做線性搜尋。分類循序索引結構，則是將關鍵索引依索引長度細分爲16類，每一類中的關鍵索引長度都相同，查詢依據樂句長度，到相對應的分類中線性搜尋。

表一：真實與人造歌曲相關資訊

模式 / 相關資訊	真實歌曲	人造歌曲
歌曲數量	100	100,000
平均索引量	18.85	20
樂句數量	1,885	2,000,000
去除重覆樂句數量	1,600	1580,082

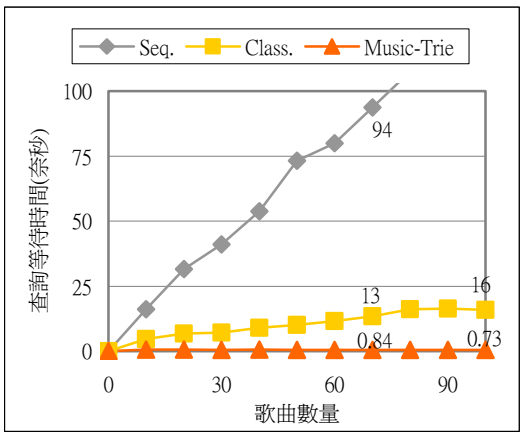


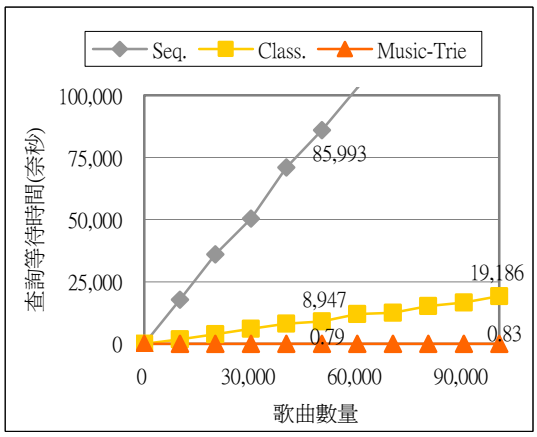
圖十六：百首單曲樂句變化情況與相對百分比

5.1 查詢索引的等待時間

圖十七與圖十八分別從真實1600句關鍵索引與人造160萬句關鍵索引中，分別隨機選擇100句索引，計算在三個方法中連續查詢100句索引所需花費的查詢時間。除了Music-Trie索引結構外，循序索引結構與分類循序索引結構的查詢等待時間隨著歌曲數量的增加而成長。

循序索引結構提供循序的比對，隨著關鍵索引數量的增加，耗費相當多的等待時間。分類循序索引結構將全部關鍵索引依長度分成16類，依照查詢索引的長度只針對單一分類比對來加速查詢，花費的查詢等待時間大約是循序索引的十分之一。Music-Trie索引結構無論關鍵索引量的多寡，查詢等待時間與歌曲數量無關，平均查詢時間爲0.81奈秒。（註：無論在百首歌曲或十萬首歌曲，Music-Trie索引結構的查詢時間都介於0.75~0.85奈秒之間，平均查詢時間則是分別將各歌曲數量下的查詢時間加總後平均）。



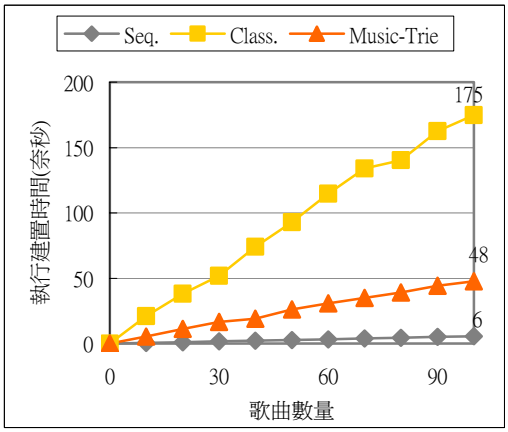


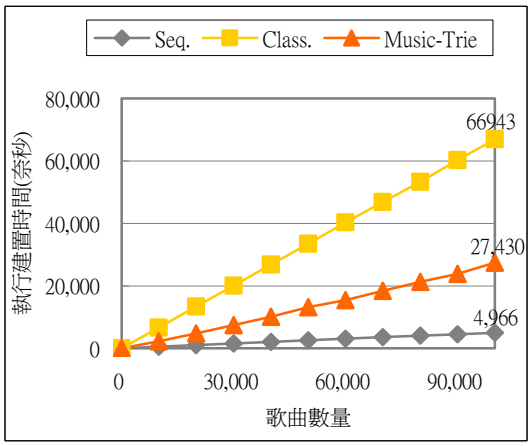
圖十七：100 個連續查詢在真實關鍵索引中的等待時間

圖十八：100 個連續查詢在人造關鍵索引中等待時間

5.2 建置索引的執行時間

圖十九與圖廿分別爲建置真實1600句關鍵索引與人造160萬句關鍵索引，相對花費的執行時間。循序索引結構沒有提供任何加速查詢的方法，相對花費最少的執行建置索引時間，平均建置一個索引的執行時間爲0.003奈秒。其次是Music-Trie索引結構，以增加分支節點方式提供穩定直接的查詢回饋，平均執行時間爲0.017奈秒。最差爲分類索引結構，比循序索引結構多一個分類的步驟，平均執行時間爲0.042奈秒。（註：平均建置索引時間＝建置索引的總時間／實際建置索引數量）



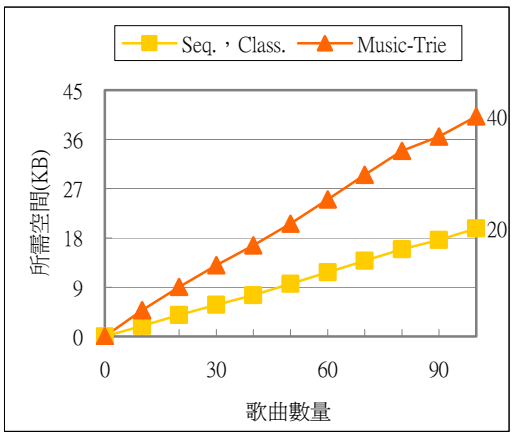


圖十九：建置真實 1600 句關鍵索引所花費的執行時間

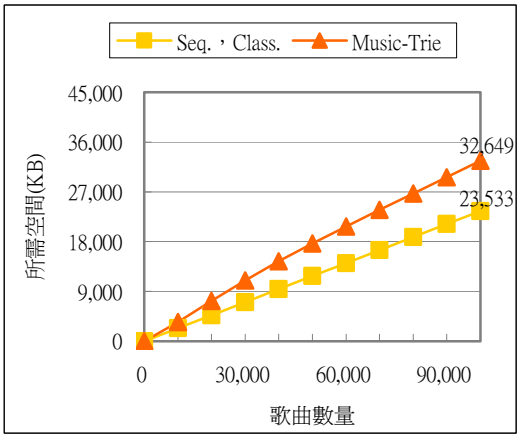
圖廿：建置人造 160 萬句關鍵索引所花費的執行時間

5.3 建置索引結構花費儲存空間

圖廿一與圖廿二分別爲建置百首真實歌曲與十萬首人造歌曲的關鍵索引所需花費的儲存空間。循序索引結構與分類索引結構都沒有增加額外空間來幫助加速查詢，因此這兩者建置索引花費儲存空間的需求相同。而Music-Trie索引結構以增加分支節點的方式來提昇查詢速度，因此所需儲存空間比另外兩個方法高。接下來我們接進一步探討，每個索引平均分擔多少儲存空間。



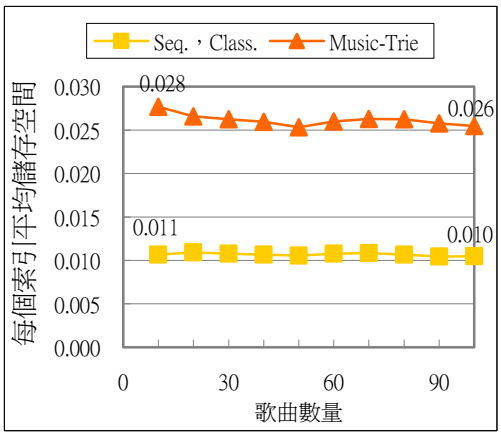
圖廿一：建置百首真實歌曲的關鍵索引所需花費的儲存空間



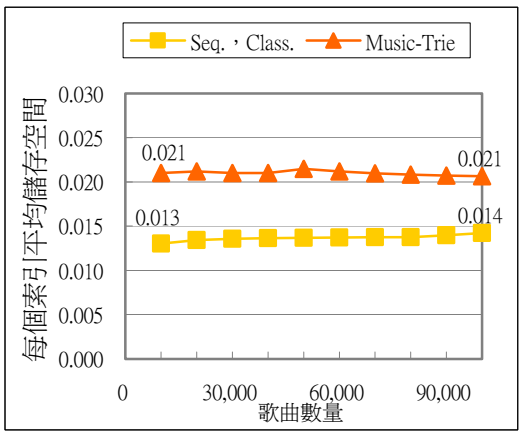
圖廿二：建置十萬首人造歌曲的關鍵索引所需花費的儲存空間

圖廿三與圖廿四分別探討建置真實1600句關鍵索引與人造160萬句關鍵索引，平均每個關鍵索引分擔多少儲存空間。在圖廿三，建置在Music-Trie索引中每個關鍵索引平均分攤空間爲0.026KB。循序索引與分類索引的關鍵索引平均分攤空間爲0.011KB。(註：每個關鍵索引平均分攤空間＝索引所需總儲存空間／實際索引數量)

隨著加入的關鍵索引增加，如圖廿四，Music-Trie索引結構中每個關鍵索引平均分攤空間將會減少。因爲當加入的關鍵索引增加，較多的關鍵索引共用的一個分支節點。因此每個關鍵索引額外分攤的分支節點的空間將減少，平均分攤空間爲0.021KB。循序索引與分類索引平均分攤空間會漸增，因爲較短的關鍵索引會有重覆出現而被去除，但較長關鍵索引卻很少重覆而被加入索引結構中。



圖廿三：在百首真實歌曲中，每個索引平均所需的儲存空間



圖廿四：在十萬首人造歌曲中，每個索引平均所需的儲存空間

5.4 綜合比較分析

Table 2列出循序索引、分類循序索引與Music-Trie索引結構三個方法的時間與空間複雜度比較。假設總索引數量爲N，每個關鍵索引的長度爲2m，最多由m組音名字組組成，每音名字組最多n種可能。其中m與n爲常數，分別在4.1與4.2節實驗分析的結果，m

等於每句樂句最多變化情況16，n等於優化後的頻率範圍23。

循序索引結構搜尋時間與索引資料量相關，時間複雜度爲O(N)；空間需求爲總索引數量和每個關鍵索引長度的乘積，也就是2Nm，空間複雜度爲O(Nm)。分類索引結構搜尋時只比對單一分類，搜尋時間爲O(N/m)；空間複雜度和循序索引結構相同，爲O(Nm)。Music-Trie索引結構搜尋時間與樹的深度相關，樹的深度小於等於最多樂句變化情況，故時間複雜度爲O(1)；空間需求爲葉節點空間加上分支節點的空間，分支節點空間爲分支節點個數與分支節點大小的乘積，也就是$2Nm+n^{m-1}\times n$，空間複雜度主要受分支節點影響，爲$O(n^m)$。三種索引結構的關係，時間複雜度Music-Trie＜Classification＜Sequence，空間複雜度Music-Trie＞Classification＝Sequence。

表二：時間與空間複雜度比較

比較 / 方法	時間複雜度	空間複雜度
Sequence	O(N)	O(Nm)
Classification	O(N/m)	O(Nm)
Music-Trie	O(1)	$O(n^m)$

6. 結論

本文以MP3歌曲的內容爲檢索基礎，提出動態擷取MP3關鍵樂句和Music-Trie索引結構兩個方法。在動態擷取MP3關鍵樂句方面，我們採用過濾背景音樂方式幫助動態分段，再擷取關鍵樂句建置在資料庫中。動態分段降低去除重覆片段時的複雜度，且適用於任何樂句型態的歌曲。被擷取出來的關鍵樂句，忽略樂句節拍的速度來幫助且校正使用者哼唱上速度的錯誤。

在資料庫方面以Music-Trie索引結構方式建置所有的關鍵樂句。Music-Trie索引結構以增加分支節點的方式，提供快速直接的查詢。實驗以1600句真實關鍵索引與160萬句人造關鍵索引，分析Music-Trie索引結構的效能。實驗結果顯示，索引資料量的多寡不影響查詢時間，查詢時間的複雜度爲O(1)。

相較於其他以 MP3 Content-Based Retrieval 爲議題的文獻，本文具備了以下的優勢與貢獻：

- 以樂句爲單位的分段法適用任何長短句型的歌曲。
- 關鍵索引忽略樂句速度，改善使用者哼哼唱速度的不一致。
- Music-Trie 索引結構查詢的時間複雜度爲 O(1)。

本文提供使用者哼唱速度上的校正處理，在未來研究工作將針對非專業使用者在哼唱時會調整成適合自己的音域範圍，而形成比對上的誤差方面的問題，做更深入的研究探討。

Reference

[1] Bakhmutova, V., Gusev, V.D. and Titkova, T.N. “The Search for Adaptations in Song Melodies”, *Computer Music Journal, Vol. 21, No. 1*, pp. 58-67, Spring 1997.

[2] Chen, Arbee L.P., Chang, M., Chen, J., Hsu, J.L., Hsu C.H. and Spot Y.S., “Query by Music Segments: An Efficient Approach for Song Retrieval”, *In Proc. Of IEEE Int'l Conf. on*

Multimedia and Expro, New York, pp. 873-876, 2000.

[3] Chen, Hung-Chen and Chen, Arbee L. P., "A Music Recommendation System Based on Music Data Grouping and User Interests", *In Proc. of ACM International Conference on Information and Knowledge Management CIKM'01*, pp. 231-238, Nov. 2001.

[4] Chen, J. C. C. and Chen, A. L. P., "Query by Rhythm: An approach for song retrieval in music databases", *In Proceedings of the 8th International Workshop on Research Issues in Data Engineering, Orlando, Florida, USA*, pp. 139-146, February 1998.

[5] Chou, T.C., Chen, A.L.P. and Liu, C.C., "Music Databases:Indexing Techniques and Implementation", *In Proc. of IEEE Intl. Workshop on Multimedia Data Base Management* System, pp. 46-53, August 1996.

[6] Foote, J. "An overview of audio information retrieval", *ACM Press & Springer-Verlag,* Multimedia *Systems*, vol. 7, no. 1, pp. 2-10, 1999.

[7] Foote, J. "Content-Based Retrieval of Music and Audio", *Multimedia Storage and Archiving systems II, Proc. SPIE*, Vol.3229, pp. 138-147, 1997.

[8] Ghias, A., Logan, J., Chamberlin, D and Smith, B. C., "Query by Humming: Musical Information Retrieval in an Audio database", *In Proc. of ACM Multimedia, San Francisco, California*, pp. 231-236, Nov. 1995.

[9] Horowitz, Ellis, Sartaj Sahni and Anderson-Freed, Susan "Fundamentals of Data Structures in C", *Computer Science Press, New York*, pp. 557-562, 1993.

[10] Hsu, Jia-Lien, and Chen, Arbee L.P., "Efficient Repeating Pattern Finding in Music Databases", *CIKM 1998, Bethesda, Maryland, United States*, pp. 281-288, Nov. 1998.

[11] Hsu, Jia-Lien, Liu, C.C. and Chen, Arbee L.P., "Discovering Non-Trivial Repeating Patterns in Music Data, Multimedia", *IEEE Transactions on Multimedia*, Vol. 3, No. 3, pp. 311-325, Sept. 2001.

[12] ISO/IEC 11172-3, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s-Part 3 : Audio", *ISO/IEC JTC1/SC 29*, May 20, 1993.

[13] Jang, J.-S. Roger and Gao, Ming-Yang, "A Query-by-Singing System based on Dynamic Programming", *International Workshop on Intelligent Systems Resolutions (the 8th Bellman Continuum), Hsinchu, Taiwan*, pp. 85-89, Dec. 2000.

[14] Jang, J.-S. Roger, Lee, Hrong-Ru and Yeh Chia-Hui, "Query by Tapping: A New Paradigm for Content-based Music Retrieval from Acoustic Input", *The Second IEEE Pacific-Rim Conference on Multimedia, Beijing, China*, October 2001.

[15] Kosugi, N., Nishihara, Y., Kon'ya, S., Yamamuro, M. and Kushima, K. "Music Retrieval by Humming", *In Proceedings of IEEE PACRIM'99*, pp. 404-407, 1999.

[16] Kosugi, N., Nishihara, Y., Sakata, T., Yamamuro, M. and Kushima, K., "A practical query-by-humming system for a large music database", *In Proc. ACM Int. Conf. on Multimedia, Los Angeles, CA*, pp. 333-342, October 2000.

[17] Krumhansl, C.L. "Cognitive Foundations of Musical Pitch", *Oxford University Press, New York*, pp. 180-185, 1990.

[18] Lee, I-Yang, Jang, J.-S. Roger and Hsu, Wen-Hao, "Content-based Music Retrieval from Acoustic Input", *12th IPPR Conference on Computer Vision, Graphics, and Image Processing, Taiwan*, pp. 325-330, August 1999.

[19] Lee, Wegin and Chen, Arbee L.P., "Efficient Multi-Feature Index Structures for Music Data Retrieval", *In Proc. of SPIE Conference on Storage and Retrieval for Image and Video Databases*, pp. 177-188, 2000.

[20] Lin, Chang-Rong, Liu, Ning-Han, Wu, Yi-Hung, and Chen, Arbee L.P., "Music Classification Using Significant Repeating Patterns", *International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, pp. 506-518, March 2004.

[21] Liu, C.C., Hsu, J.L. and Chen, Arbee L.P., "An Approximate String Matching Algorithm for Content-Based Music Data Retrieval", *In Proc. of IEEE Int'l Conf. on Multimedia Computing and Systems, Florence, Italy*, pp. 451-456, June 1999.

[22] Liu, C.C., Hsu, J.L. and Chen, Arbee L.P., "Efficient Theme and Non-Trivial Repeating Pattern Discovering in Music Databases", *15th International Conference on Data Engineering, Sydney, Australia*, pp. 14-21, March 1999.

[23] Liu, Chih-Chin and Huang Chuan-Sung, "A Singer Identification Technique for Content- Based Classification of MP3 Music Objects", *International Conference on Information and Knowledge Management (CIKM 2002), USA*, pp. 438-445, 2002.

[24] Liu, Chih-Chin and Kou Wei-Yi, "Content-Based Segmentation of MP3 Music Objects", *In Proc. of the Workshop on the 21st Century Digital Life and Internet Technologies, Tainan, Taiwan*, 2001.

[25] Liu, Chih-Chin and Tsai, Po-Jun, "Content-based retrieval of MP3 music objects", *Proceedings of the tenth international conference on Information and knowledge management, October 05-10, 2001, Atlanta, Georgia, USA*, pp. 506-511, 2001.

[26] Liu, Chih-Chin and Yao, Pang-Chia, "Automatic Summarization Of Mp3 Music Objects", *International Conference on Speech, Acoustics, and Signal Processing (ICASSP 2004) , Canada, May, 2004*.

[27] Liu, Chih-Chin and Yu, Hung-Min, "Effective Segmentation And Retrieval Of Mp3 Music Objects", *In Proc. of Third International Workshop on Content-Based Multimedia Indexing (CBMI 2003), France*, 2003.

[28] Logan, B. and Chu, S., "Music summarization using key phrases", *In Proceedings of IEEE International Conference on Audio, Speech and Signal Processing, Orlando, USA*, vol. 2, pp. 749-752, 2000.

[29] Lo, Yu-lung, Yu, Ho-cheng and Fan, Mei-chin, "Efficient Non-trivial Repeating Pattern Discovering in Music Databases," *Tamsui Oxford Journal of Mathematical Sciences*, Volume 17, No. 2, pp. 163-187,.Nov. 2001.

[30] Lo, Yu-lung, Yu, Ho-cheng and Fan, Mei-chin, "FastPET: A Fast Non-trivial Repeating Pattern Extracting Technique for Music Data," *2001 National Computer Symposium - Multimedia, Computer Graphics, and Image Processing, Taipei*, pp. D043-052, Dec. 2001.

[31] Martin, K. D. and Kim, Y. E., "2pMU9. Musical instrument identification: A pattern-recognition approach", *In the 136th meeting of the acoustical society of America, Norfolk, VA.*, October 1998.

[32] McCreight , Edward M., "A Space-Economical Suffix Tree Construction Algorithm", *Journal of the ACM (JACM)*, v.23 n.2, pp. 262-272, April 1976.

[33] Melih, K. and Gonzalez, R. "Audio Retrieval Using Perceptually Based Structures", *In Proc. of IEEE International Conference on Multimedia Computing and System, Austin, Texas, USA*, pp. 338-347, 1998.

[34] Pfeiffer, S., Fischer, S. and Effelsberg, W,. "Automatic audio content analysis", *In Proc. of ACM Multimedia Conference, Boston, Massachusetts, United States*, pp. 21-30, 1996.

[35] Tseng, Y.H., "Content-Based Retrieval for Music Collections", *In Proc. of ACM SIGIR'99, Berkley, CA, USA*, pp. 176-182, 1999.

[36] Tzanetakis, G., and Cook, P., "A Framework for Audio Analysis Based on Classification and Temporal Segmentation", *In Proc. EUROMICRO Conf., Milan, Italy*, Vol. 2, pp. 61-67, 1999.

[37] Wang, Yi-Chun, "Constructing Classified Index Structures for Repeated Patterns of Segmented MP3 Songs", Southern Taiwan University of Technology, Taiwan, 2004.

[38] Wold, E., Blum, T., Keislar, E. and Wheaton, J., "Content-Based Classification, Search, and Retrieval of Audio", *IEEE Multimedia*, Vol. 3, No. 3, pp. 27-36, Fall 1996.