國立政治大學資訊科學系
Department of Computer Science
National Chengchi University

碩士論文

Master's Thesis

華語流行音樂之詞式分析與詞曲結構搭配之排比與同步

Lyrics Form Analysis for Chinese Pop Music with Application to Structure Alignment between Lyrics and Melody

研 究 生：范斯越

指導教授：沈錳坤

中華民國一百零一年九月

September 2012

摘要

目前大部分的聽眾主要是透過歌詞與樂曲的搭配來了解音樂所要表達的內容，因此歌詞創作在目前的音樂工業是很重要的一環。一般流行音樂創作是由作曲人與作詞人共同完成，然而有另一種方式是將既有的詩詞做為歌詞，接著重新譜曲的方式產生新的流行音樂。這種創作方式是讓舊有的詞或曲注入新的生命力，得以流傳到現在。因此本研究希望可以為一首旋律推薦適合配唱的歌詞，以對數位音樂達到舊曲新詞的加值應用。本論文包括兩個部分，分別為：(1)自動分析歌詞的詞式，找出每個段落的位置與其段落的標籤；(2)詞曲結構搭配，找出相符合結構的詞與曲，並且同步每個漢字與音符。

本論文的第一部分為詞式分析，首先將歌詞擷取四個面向的特徵值，分別為(1)句字數結構；(2)拼音結構；(3)詞性；(4)聲調音高。第二步驟，利用這四種特徵值分別建立詞行的自相似度矩陣(Self Similarity Matrix)，並且利用這四個特徵的自相似度矩陣產生一個線性組合自相似度矩陣。第三步驟，建立在自相似度矩陣上我們做段落分群以及家族(Family)組合找出最佳的分段方式，最後將找出的分段方式利用我們整理出來的規則讓電腦自動標記段落標籤。第二部分為詞曲結構搭配，首先我們將主旋律的樂句以及歌詞的詞句做第一層粗略的對應，第二步驟，將對應好的樂句與詞句做第二層漢字與音符細部的對應，最後整合兩層對應的成本當做詞曲搭配的分數。

我們以 KKBOX 音樂網站當做歌詞來源，並且請專家標記華語流行歌詞資料庫的詞式。實驗顯示詞式分析的 Pairwise f-score 準確率達到 0.83，標籤回復準確率達到 0.78。詞曲結構搭配中，查詢的歌曲其原本搭配的歌詞，推薦排名皆為第一名。

Abstract

Nowadays, lots of pop music audiences understand the content of music via lyrics and melody collocation. In general, a Chinese pop music is produced by composer and lyricist cooperatively. However, another producing manner is composing new melody with ancient poetry. Therefore, we want to recommend present lyrics for a melody and then achieving value-added application for digital music. This thesis includes two subjects. The first subject is lyrics form analysis. This subject is finding the block of verse, chorus, etc., in lyrics. The second subject is structure alignment between lyrics and melody. We utilize the result of lyrics form analysis and then employ a 2-tier alignment to recommend present lyrics which is suitable for singing.

In lyrics form analysis, the first step, we investigate four types of feature from lyrics: (1) Word Count Structure; (2) Pinyin Structure; (3) Part of Speech Structure; (4) Word Tone Pitch. For the second step, we utilize these four types of feature to construct a SSM(Self Similarity Matrix), and blend these four types of SSM to produce a linear combination SSM. The third step is clustering blocks and finding the best Family combination based on SSM. Finally, a rule-based technique is employed to label blocks of lyrics. For the second subject, the first step is aligning music phrases and lyrics sentences roughly. The second step is aligning a word and a note for corresponding phrase and sentence. Finally, we integrated the cost of two-level alignment regarded as the lyrics and melody collocation score.

We collect lyrics from KKBOX, a music web site, and invite experts label ground truth of lyrics form. The experimental result of lyrics form analysis shows that the proposed method achieves the Pairwise f-score of 0.83, and the Label Recovering Ratio of 0.78. The experiment of structure alignment between lyrics and melody shows that the original lyrics of query melodies are ranked number one.

誌謝

轉眼間在政大經過了三個寒暑，碩士生涯也要告一段落，首先感謝我的父母，提供我穩定的經濟來源，讓我可以恣意的遨遊在音樂研究裡。感謝哥哥送我一台性能優越的筆電，讓我得以順利完成程式以及碩士論文。感謝沈老師在這三年間教導我很多研究的觀念、簡報與 DEMO 的技巧、分享產業界的趨勢、不時的提醒我需要改善的壞習慣以及讓我擔任實驗室大師兄的角色，放心的讓我處裡實驗室的大小事，培養我的責任感。這些的經歷都讓我獲益良多，在未來都能夠成為我達成下一個目標的助力。

我也要感謝實驗室眾多的成員，世宏三年來一步一步的朝他的目標邁進，踏實的做事是值得我學習的地方。戴張的搞笑功力，讓實驗室有時嚴肅的氣氛頓時充滿歡笑，還有他一個人勇敢的在異鄉交換學生，勇於嘗試值得我學習。志傑做事的果斷，不停的學習新的資訊技術，這是我值得學習的地方。柏堯玩音樂的態度，從組團、錄音到 live house 表演，都有適時的抓住機會表現，做好準備等待機會值得我去學習。建成關心他人，體貼他人，讓人感覺很溫暖，關懷他人很值得我去學習。柏聿為人誠懇，不時的從嘉義帶特產與實驗室成員分享，讓大家在做研究時可以適時補充腦力，這種分享的心值得我學習。世通清晰的口條，在我挫折的時候會給我正向的思維，這種正向思考鼓勵他人值得我去學習。還有感謝新加入實驗室的學弟蘇瀚、嘉泰、元翰與楚翔參加我的論文口試，為我加油打氣。

感謝韓助教幫我處理很多行政上的問題，即使我一直麻煩她，她還是不厭其煩的回答。感謝子安不離不棄的陪我度過很多研究難熬的日子，她的溫柔與鼓勵時常又讓我重拾研究的動力。感謝大提琴陪伴我度過好幾個煩惱的夜晚，讓我的煩惱可以消散在其低沉的琴音當中。

還有感謝大學部的學弟妹，博為、睿陵、鉦凱、暐翔、柏皓、執中、FiFi、博軒、Lucy、家霖與俊鋒為實驗室帶來更多歡樂的氣氛。大家一同經歷的酸、甜、苦、辣，是我們培養情感的養分，這三年發生的種種會一直留在我心中，我衷心

的祝福大家在未來能夠大步的往自己理想的目標邁進，開創自己人生的一片天！！

范斯越　謹誌于
政治大學資訊科學所
2012.10.15

國立政治大學 National Chengchi University

目錄

國立政治大學 National Chengchi University

表目錄

國立政治大學 National Chengchi University

圖目錄

國立政治大學 National Chengchi University

第 1 章

前言

1.1. 動機

流行歌曲廣義上的定義為通過大眾媒體傳播、以大眾為接受對象的歌曲[39]，而目前流行音樂的製作包含兩大部分，音樂創作與錄音後製。在流行音樂創作的部分中包含了作曲、作詞以及編曲。作曲人通常會去參考作詞人的歌詞[33]，分析歌詞的主題進而決定樂曲的風格，並且了解歌詞的段落結構以及聲韻，來決定樂曲的拍子、節奏或是調性等等音樂上的特徵，完成一首有主要旋律的樂曲。

相對於作曲，作詞是屬於文學的學科，作詞人也會參考作曲人的樂曲[33]，了解樂曲的長度，決定歌詞的字數，分析樂曲的曲式結構，決定歌詞的結構，聆聽樂曲的主題、情感，決定歌詞的中心思想、意境。以上兩個工作都是一個從無到有的過程，兩者並沒有一個明確的先後關係，因為雙方都會需要互相參考，所以必須要頻繁的交換意見，才能完成一個完整的流行音樂。

現在大部分的聽眾，即使沒有受過專業的音樂訓練，也是可以透過歌詞來了解音樂要表達的主題，並且現在熱門的音樂很少會是以單純的樂器演奏的形式呈現[1]。由此可知，有較多的聽眾可以藉由歌詞跟樂曲搭配的形式來了解音樂，因此歌詞創作在目前的音樂工業是很重要的一環。目前市面上有很多的音樂播放軟體(例如：千千靜聽、Winamp)，有提供動態歌詞的功能。所謂動態歌詞是指當流行歌曲中的演唱者，演唱到某句歌詞的時候，音樂播放軟體會把對應的歌詞同步顯示。這樣的功能可以幫助使用者即時了解目前歌曲演唱的歌詞內容，並且，動態歌詞還可以提供類似在 KTV 裡歌詞引導的功能，使用者可以用此練習演唱

的技巧。

目前有很多華語流行音樂是翻唱自其他國家的歌曲，一般的作法都是保留原曲的主旋律，可能會把伴奏或是樂器的部分做改變，最後填上中文的歌詞。以《古老的大鐘》為例，這首歌被歌手李聖傑翻唱過。大部份的人會認為《古老的大鐘》是源自於日本民謠，但是實際上這首歌的原作者是Henry Clay Work(1832-1884)，一位美國人於 1876 年所寫的一首歌曲，只不過這首歌在 1940 年代隨著戰爭的結束，傳入日本，經過翻譯成為了日本的民謠。這首歌的背景大約是在述說，一個古老的時鐘伴隨作曲者的祖父生活的真實故事，最後在老祖父過世之後，老時鐘也就隨之停擺，似乎是在宣示老時鐘伴隨著老祖父一起離開了，歌詞感人真摯。因此，一首一百多年前的歌曲透過新的翻唱，填上新的歌詞，就可以注入新的生命，讓這首歌曲源遠流傳到現在。因此本研究希望透過歌詞的詞式分析，自動判斷歌詞的主歌、副歌等等的段落，應用在詞曲結構搭配，並且自動同步音樂與歌詞的音符與漢字的對應關係。

1.2. 歌詞推薦系統應用

本研究可以應用在歌詞推薦系統，此系統的目標為輸入一個流行音樂的複音(Polyphonic)MIDI，系統可以自動推薦適合配唱的歌詞，並且每一首推薦的歌詞都有與輸入的音樂做音符與漢字的同步處理，讓使用者了解如何演唱新詞。如圖 1.1 為系統目標的示意圖。

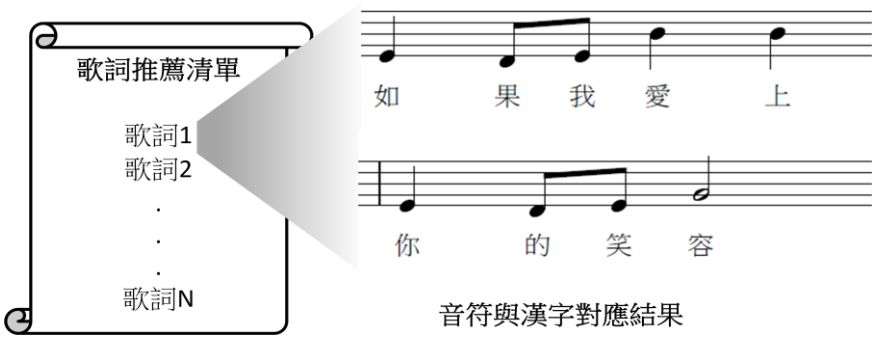


圖 1.1 系統目標示意圖

歌詞推薦系統分成三大部分：(1)歌詞處理(Lyrics Processing)；(2)音樂處理(Music Processing)；(3)對齊與排名(Alignment and Ranking)，系統架構圖如圖 1.2 所示。本研究在此系統涵蓋了(1)與(3)部分(灰色底色)。歌詞處理部分主要分成三個元件：(1)歌詞詞式分析(Lyrics Form Analysis)，將歌詞分段並且標出每個段落的意義（例如主歌、副歌或橋段）；(2)歌詞斷詞(Lyrics Tokenization)，將中文歌詞擷取出有意義的詞彙；(3)歌詞主旋律產生(Lyrics Melody Generation)，利用中文為聲調語言的特性，產生中文流行歌詞聲調對應的旋律。

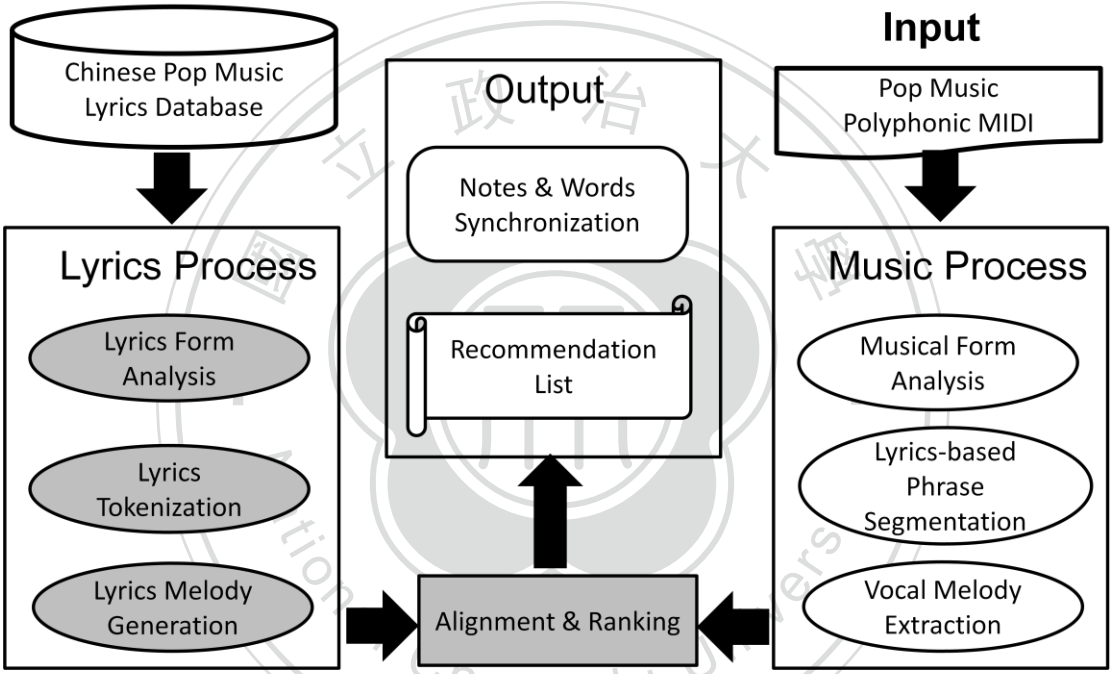


圖 1.2 歌詞推薦系統架構圖

音樂處理目的是要抓取跟歌詞相對應的特徵，分成三個元件：(1)音樂結構分析(Musical Form Analysis)，將抓出的主旋律做分段，並且標出每個段落的意義（例如主歌、副歌或橋段）；(2)基於歌詞的樂句分段(Lyrics-based Phrase Segmentation) 將主旋律找出樂句的分段點，而樂句的分段是依據其歌詞中一句歌詞的分段點；(3)人聲旋律擷取(Melody Extraction)，抓取使用者輸入的複音 MIDI 的人聲旋律。

最後為詞曲的排名與同步。我們先將歌詞中詞句的結構與音樂的樂句結構做

第一層的最佳化的對應，接下來在做漢字與音符第二層最佳化的對應。在第二層的對應中，考慮了詞式與曲式的搭配、中文歌詞產生的旋律與音樂主旋律的搭配以及歌詞唱音節奏與主旋律節奏的搭配。最後計算出詞曲搭配的分數，再根據搭配分數得到歌詞推薦清單，而在產生推薦清單的同時也完成了歌詞中的每個漢字與音樂音符的對應關係，如此便達到推薦歌詞系統的目的。

1.3. 論文架構

本論文之架構如下，第一章介紹研究動機、研究目的與歌詞推薦應用實例；接下來，第二章探討相關研究，分成四大方向：(1)音樂結構分析之相關方法；(2)歌詞與旋律關係研究；(3)流行音樂與歌詞自動同步；(4)由歌詞產生旋律與由旋律產生歌詞，其中(1)與詞式分析相關，(2)、(3)與(4)則與詞曲結構搭配相關。第三章介紹詞式分析研究方法，主要分成四大步驟：(1)行與行相似度計算；(2)自相似度矩陣建立；(3)重複樣式探勘；(4)詞式標記。第四章介紹詞曲結構搭配的研究方法，主要分成兩大步驟：(1)詞句與樂句結構排比；(2)漢字與音符同步；第五章介紹實作方法、建立環境、詞式評估方法，詞式分析實驗結果以及詞曲結構搭配實驗結果。第六章為結論與未來工作。

國立政治大學 National Chengchi University

第 2 章

相關研究

關於詞式分析與詞曲搭配目前還沒有相同目標的研究，因此以下整理出四個與本研究最相近的研究議題：(1)音樂結構分析；(2)歌詞與旋律關係研究；(3)流行音樂與歌詞自動同步；(4)由歌詞產生旋律與由旋律產生歌詞。

2.1. 音樂結構分析

現在的流行音樂中常常分成幾個部分，分別為主歌、副歌、前奏、間奏與尾奏。音樂家會將這些部分依照一定的邏輯排列，目的為了達到他想表達的內容，這樣的排列我們稱為曲式。其中主歌或副歌是最基本的曲式單元[34]，主歌是傳達一首歌曲的背景故事，是整首歌曲的骨幹。副歌往往是令聽眾最印象深刻的片段，通常也是歌曲的高潮，並且會相對於主歌重複多次，用以加深聽眾的記憶。因此有學者希望可以透過電腦自動分析出音樂的曲式結構，如此可以應用在 Indexing 或是 Audio Thumbnailing 等等的領域。Foote[3]是第一個利用 SSM(自相似度矩陣)來視覺化音樂訊號的學者，隨後的研究大部分都是建立在 SSM 上來做後續的音樂結構分析。

音樂結構分析的第一步通常為音樂的特徵值擷取，擷取的方式是利用 10-100 毫秒的 Frame 慢慢的移動覆蓋整個音樂訊號，並且對每個 Frame 的訊號內容進行特徵值擷取，最後一個音樂訊號就會形成一條特徵值序列。抓取的特徵值通常為高階的特徵，像是 MFCC(Mel-scale Frequency Cepstral Coefficient)、Chroma 與 Rhythmogram 等等，其對應的意義依序為音色、音高或節奏，每種特徵值通常表示為一個向量的形式。有了一序列的音樂特徵值 $x_1,x_2,\ldots x_N$，接下來再對特徵序列的元素兩兩計算相似度(或距離)產生一個 N*N 的 $SSM(i, j) = sim(x_i, x_j)$,

$\forall i, j \in \{1,2,..N\}$，sim 為兩個特徵元素的相似度函數。

有學者[23]整理了訊號基底(Audio-Based)的音樂結構分析的研究，他們將到目前為止的音樂結構分析方法分成三大部分：(1)Novelty-based(新奇性)；(2)Homogeneity-based(同質性)；(3)Repetition-based(重複性)。

2.1.1. Novelty-based

主歌與副歌之間往往會有一個變化，可能是音量、節奏或是樂器等等的面向，如此是為了可以吸引聽眾，讓音樂不單調。因此 Foote[4]提出偵測 Novelty 的方式來找出音樂中段落的分段點。他先以 MFCC 特徵為基礎，建立一個 N*N 的 SSM 接著利用一個 M*M 的 Checkerboard Kernel 矩陣，他將此 Checkerboard Kernel 矩陣延著 SSM 的主對角線掃描，計算每一個時間點的 Novelty，得到一個 Novelty 的隨時間的分佈，接著再做峰值擷取(Peak Picking)。Novelty 分數越高表示越可能是一個分段點。

2.1.2. Homogeneity-based

這個方向的目標是將輸入的音樂訊號視為一連串的狀態(state)，並且將同質性的狀態分為同一個族群。Cooper 等人[2]利用先前提到的偵測 Novelty 的方法，找出音樂訊號的分段點，接下來將這些段落利用 Kullback-Leibler Divergence 計算段落兩兩之間的距離產生段落的 SSM。通常特徵序列 SSM 的大小為幾千乘幾千，轉換成段落的 SSM 後，大小降為幾十乘幾十，如此可以減少計算量。接著對段落 SSM 做 SVD(Singular Value Decomposition)，可以得到 K 個特徵值(Singular Value)，這也意味著可以將目前切割出的段落分成 K 群，最後再將每個段落依據其最大可能性選擇其屬於的族群，達到段落分群，產生出音樂結構。

2.1.3. Repetition-based

重複是音樂創作很重要的元素，如此可以加深聽眾的印象。在 SSM 上若有重複樣式產生，則會出現一條很明顯的斜線樣式，例如圖 2.1 為曲式 ABABB 產生 SSM 後的示意圖，其中黑色斜線樣式代表者兩段落的特徵值序列的相似度很高。可以看到 A_1 與 A_2 之間有條斜線樣式，A_1B_1 與 A_2B_2 之間也有條斜線樣式。

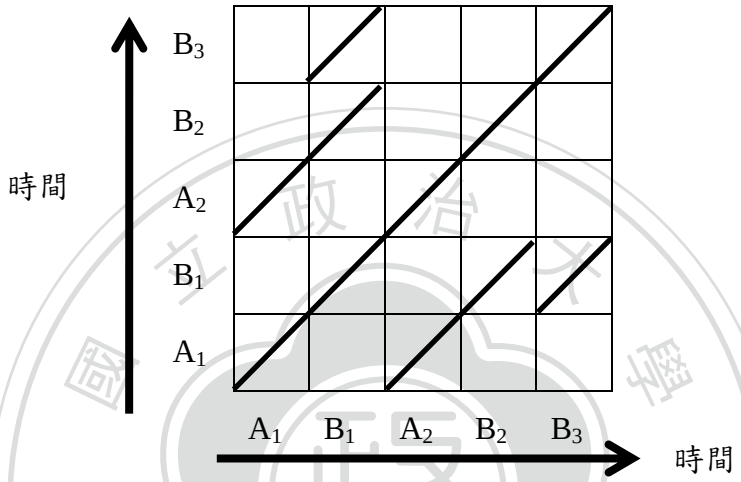


圖 2.1 SSM 上的斜線樣式

圖 2.1 為一個理想情況，實際上 SSM 上的斜線樣式產生是取決於抓取的音樂特徵值，因此 Muller 等人[19]希望可以強化斜線樣式的呈現並且去除雜訊，將由音樂特徵序列產生的 SSM 考慮 Contextual Information 產生一個強化的(Enhanced)SSM。而 Peeters[24]則將由音樂特徵序列產生的 SSM 考慮遞移關係，也就是當 A 與 B 像，B 與 C 像，則可以推論 A 與 C 也會像，這種性質產生一個 Higher-order 的 SSM。這兩種方法都可以達到強化斜線樣式的目的。

在 SSM 上有了清楚的斜線樣式後，接下來便是自動找出取出這些斜線樣式。Mueller 等人[18]利用貪婪(Greedy)的方式並且設定門檻值，找出斜線樣式，接下來將取出的斜線樣式考慮遞移關係，找出音樂的重複結構。

2.2. 歌詞與旋律關係研究

流行音樂的歌詞與旋律之間是否有種特定的關係存在？Nichols 等人[20]首先研究英文流行歌詞與旋律的關係，他們認為作詞人在作詞時會把歌詞中特定的特徵對應到音樂的特徵，例如作曲家會傾向於把樂曲的節奏與說話的節奏作對應。他們把歌曲的樂譜取出三個特徵：(1)標記每個音符是屬於哪一類的拍點(例如 Downbeat 是屬於一小節的第一拍，Half-beat 是屬於一小節的第三拍)；(2)標註每個音符的音高是否是一個高峰(Peak)；(3)計算每個音符的持續時間(Duration)相對於歌曲中平均音符的持續時間，如果大於一表示比平均音符持續時間來的長。歌詞也取出三個特徵：(1)標記歌詞中每個音節是屬於非重音、次重音或是主要重音；(2)標記歌詞中每個單字是否是屬於非重要的字(Stopword)；(3)標記歌詞中每個音節所屬於的母音種類。接下來把歌詞與旋律的特徵做交叉統計發現，歌詞中音節的主要重音會比較容易對應到旋律中的重拍、音高的高峰以及持續時間長的音符；非重要的字則比較容易對應到非重拍的拍點以及歌曲中低音的音符；母音中的雙母音會較容易對應到持續時間長的音符。

徐富美等人[35]則是檢測中文的歌詞與旋律是否搭配的和諧。他們提出五個歌詞與旋律搭配和諧的關係：(1)歌詞中單一的字，聲調高低起伏與旋律的高低起伏應配合；(2)聲調是陰平的字，宜用單音的音符，如果是二聲或是四聲，宜用兩個音來搭配；(3)歌詞中兩兩字之間與旋律對應的相鄰音高，第一聲 ≥ 第四聲 ≥ 第二聲 ≥ 第三聲，例如回家一詞，搭配的旋律應該是由低到高；(4)相鄰音高的限制是以一個樂句為單位，避免讓旋律太死板，以擴大音域來突破聲調的限制；(5)輕聲虛字如「阿」「哪」等等…可以自由對應旋律的音高，並不受自虛字聲調的限制。他們利用以上五個規則來判斷使用者輸入的歌詞與既有的樂譜搭配的和諧程度。

2.3. 流行音樂與歌詞自動同步

再另一方面目前動態歌詞與 KTV 裡歌詞演唱節奏的引導，都還是利用人工來標記歌詞與音樂對應的時間點。圖 2.2 為動態歌詞用的時間標記的檔案格式(LRC)。

[mm:ss.xx] 第一行歌詞

[mm:ss.xx] 第二行歌詞

…

[mm:ss.xx] 最後一行歌詞

圖 2.2 動態歌詞檔案格式

[mm:ss.xx]為時間標記的格式，mm 為分鐘數，ss 為秒數，mm 為百分之一秒

因此有學者研究如何自動化的同步使用者輸入的數位訊號(例如：MP3 或是 WAV 檔案)流行歌曲與此流行歌曲的歌詞。當然歌詞與歌曲同步的基本單位會有不同的大小，例如是歌詞中每一個段落與歌曲同步的時間點[11]；歌詞中每一句與歌曲同步的時間點(如 LRC 標記的方式)[5][9][15][29][30]；歌詞中每一個字與歌曲同步的時間點[14][16](如 KTV 的歌詞引導系統)；或是歌詞中每一個音節與歌曲同步的時間點[8]。Lee 等人[11]提出英文流行音樂與歌詞段落同步的研究。系統分為兩個步驟，首先利用 SSM [3] (Self-Similarity Matrix)來找出流行音樂的結構，例如前奏-主歌-副歌-主歌-副歌-尾奏。接下來利用動態規劃(Dynamic Programming)計算流行音樂段落與利用人工標記好的歌詞段落，彼此間最小的對應成本，並以此當作段落同步的時間點。

Wang 等人[9][29]提出英文流行音樂與單句歌詞同步的研究，他們是第一個提出流行音樂語歌詞同步的議題。系統首先找出歌曲的結構，他們定義了前奏、主歌、副歌、間奏與尾奏五種結構，如此可以確定歌詞在歌曲段落中出現的粗略位置。他們先偵測歌曲的拍點(Beat)，進而推算出 IBI(Inter-Beat Interval)。接下

來將 IBI 當作一個計算單位，計算每個 IBI 可能的和絃，再計算歌曲可能所屬的調性，接下來再修正剛剛和弦偵測的結果，最後利用修正過後的和弦邊界(Boundary)找出歌曲的小節結構；另一方面，利用副歌偵測(Chorus Detection)演算法找出副歌片段所為在的時間區間。有了副歌片段的時間區間以及小節結構，利用 Forward/Backward 搜尋法找出剩下的歌曲結構片段(主歌、間奏以及尾奏)出現的時間區間，例如主歌出現在第 6 小節到第 20 小節。

有了歌詞的粗略位置後，接下來要找出每段歌詞裡每句歌詞精確的同步時間點。樂曲的部分以 MM-HMM(Multi-Model Hidden Markov Model)為分類模型，預測每一個 IBI 是屬於人聲類別還是非人聲類別。歌詞的部分利用監督學習(Supervised Learning)的方式訓練好每種音素(Phoneme)在不同發音時間長度下的分佈情況，計算出每種音素平均的發音時間，再把每句歌詞中的所有音素平均發音時間作加總，作為每句歌詞演唱的時間長度。有了人聲的段落位置以及每句歌詞演唱的時間長度，接下來把兩個部分用一對一對應(One-to-one Mapping)的方式找出單句歌詞的同步時間區間。在同步時有以下三種情況，群聚(Grouping)，人聲出現的段落數比歌詞的句數還多，如圖 2.3(a)；拆解(Partitioning)，人聲出現的段落數比歌詞的句數還少，如圖 2.3(b)；直接對應，人聲出現的段落數跟歌詞的句數相同時，如圖 2.3(c)。

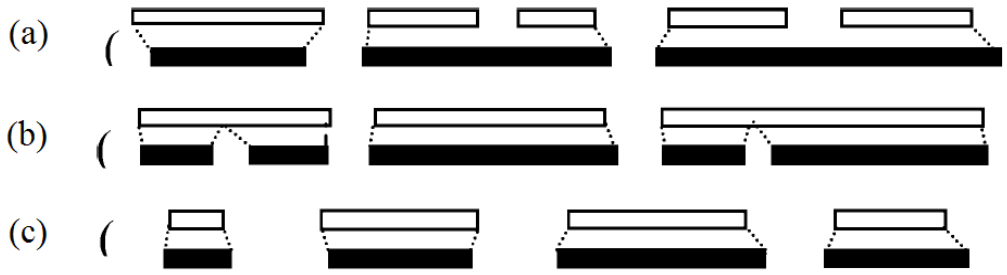


圖 2.3 人聲段落與歌詞句的對應情況，白色方框為人聲段落，黑色為單句歌詞
(a)群聚 (b)拆解 (c)直接對應

此篇研究的歌曲只有考慮主歌-副歌-主歌-副歌-間奏-尾奏的歌曲結構，這種

結構只有涵蓋 40%的流行歌曲，並且假設歌曲都是 4/4 拍，所以受限於歌曲的結構以及拍號(Time Signature)的型態。此外，在歌詞演唱的時間長度評估上，用句子中每個音素平均時間加總的方式會有很大的問題，因為只有在說話時，每種音素才會有固定的持續時間，但歌曲中每個音素發音長度是根據樂曲中音符長度來決定。

Wong 等人[30]隨後提出廣東流行歌曲與單句歌詞同步研究。在音訊處理方面，先把有人聲部分的訊號加強，再把此訊號利用 Onset 偵測，找出歌曲裡每個音符發音的時間點。接下來以 MLP(Multiple-Layer Perceptron)為分類模型，把 Onset 時間點分成人聲類別或是非人聲類別。並且利用 YIN 演算法找出歌曲的基頻(Fundamental Frequency 或 f0)，如圖 2.4 是歌曲的基頻隨著 Onset 時間點的分佈情形。

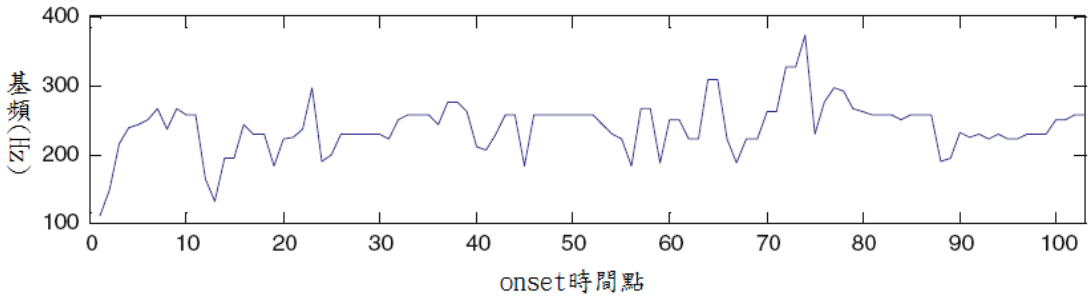


圖 2.4 YIN 音高追蹤演算法偵測出的歌曲基頻分佈

在歌詞處理方面，廣東話是屬於聲調語言(Tone Language)的特性。所謂聲調語言是指發同一個語音的時候，會因為不同的發音長度或是聲調的高低會有不同的意思，因此廣東話的歌詞基本上就隱含了樂曲旋律的基本輪廓。廣東話有六種聲調，他們把六種聲調分成三種類別，一、二聲屬於高音，三、五聲屬於中音，四、六聲屬於低音，用這些規則抓取歌詞所代表的旋律特徵。最後利用**DTW**(Dynamic Time Warping)演算法求出人聲片段的歌曲旋律特徵與歌詞旋律特徵最佳化的時間對應關係。如圖 2.5 為只有考慮相對音高特徵的同步示意圖。

圖 2.5(b)中歌詞的第 10 到 14 個字最佳化對應到圖 2.5(a)中第 27 到 61 的 Onset 時間點。此研究用 Bottom-up 的方式來做同步對應，所以歌曲中 Onset 時間點的人聲分類如果準確率不高的話，歌詞在做 DTW 同步時對應到錯誤的時間點機率就會增加。並且他們把 Onset 偵測的 Frame 大小設定在 50 毫秒，如果遇到輸入的流行音樂太長的情況，用 DTW 演算法做同步時會有計算量太大的問題，因此他們系統限定輸入的音樂片段只能在 20 秒左右。

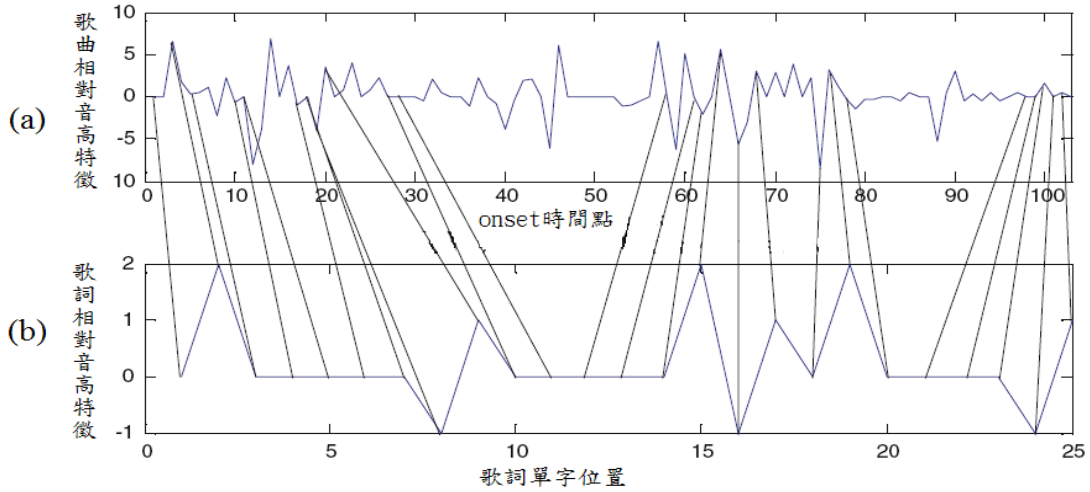


圖 2.5 單存考慮相對音高特徵的歌詞與 Onset 時間點同步示意圖

(a)音樂訊號的相對音高特徵隨 Onset 時間點的分佈情形

(b)歌詞的相對音高特徵隨歌詞單字位置的分佈情形

Maddage 等人[14]提出英文流行音樂與單字同步研究。此篇研究最大的不同是同步的時間解析度是一個單字，其最關鍵的想法是利用 TTS(Text-To-Speech)把歌詞轉換成聲音的訊號，再把流行音樂的訊號跟歌詞合成的訊號做同步。首先他們以兩個 GMM 為分類模型，分別為歌曲訊號中每個 Frame 預測是屬於人聲類別還是非人聲類別，其中以 OSCC(Octave Scale Cepstral Coefficient)為分類的特徵。接下來利用 DTW 演算法為歌詞合成的訊號以及歌曲訊號作每個單字的時間對齊，其中以 LFCC(Log Frequency Cepstral Coefficient)為計算兩兩 Frame 相似度的特徵。

以上提到的都是屬於訊號基底(Signal-based)的作法，因為他們在為歌詞與流行音樂同步時並不需要做額外的訓練(Training)，而是直接分析歌詞與流行音樂相互對應的特徵。相對於訊號基底的方法，另一方面也有模型基底(Model-based)的方法[5][8][15][16]。此方法是利用大量人工標記好的歌曲以及歌詞同步時間點，讓電腦自動學習它們之間的關聯，建立出關聯模型，再用此模型同步使用者輸入的歌曲和歌詞。因此，如果有大量的訓練資料的情況下，利用模型基底的方法會有較好的效果，反之如果只有少量的訓練資料用訊號基底的方法會有較出色的效果[14]。

Iskandar 等人[8]就是利用模型基底的方式解決英文流行音樂與音節的同步，其最主要的貢獻是在同步的時候加入了音樂知識的考量。他們導入語音辨識(Speech Recognition)常用的方法，訓練一個 HMM(Hidden Markov Model)來建立歌詞中的音素與歌曲訊號特徵的關聯，其中歌曲訊號的特徵為 MFCC。接下來再利用訓練好的 HMM 用 Viterbi 演算法，求出流行音樂與音節最佳的同步時間點。在同步時還結合了三個音樂的知識：(1)在音樂中音符的長度並不是一個任意的連續時間長度，而是跟歌曲節拍有關係的時間長度，例如一首節拍是 60，拍子是 4/4 拍的歌曲中，一個 4 分音符持續的時間就會是 1 秒，一個 8 分音符持續的時間會是 0.5 秒，在這樣的歌曲設定下比較不會出現一個持續時間為 0.68 秒的音符；(2)不同的音符長度會有不同的出現機率，例如 8 分音符跟 4 分音符會比 9/16 和 13/16 音符常出現；(3)在歌曲中不同的片段會有不同的音符分佈，例如副歌中出現的音符往往會比在主歌中出現的音符來的長。實驗結果當同步容錯範圍為 1/4 小節時，同步的錯誤率為 18.7%。

2.4. 由歌詞產生旋律與由旋律產生歌詞

由樂曲自動產生歌詞此類的議題，往往都是先研究要產生歌詞的語言特性，來做為其歌詞產生的規則或是限制條件。Tra-la-lyrics 系統[21]產生的是葡萄牙語歌詞，其利用的原理為詞語中的重音節(Stressed Syllable)要與樂曲的重拍對應。因此，他們先分析好音樂中重拍音符出現的位置，並且也計算好詞彙資料庫中，每個單字重音節出現的音節位置，之後再從詞彙資料庫利用三種選字的方法，分別是隨機選字、文句樣版(Sentence Template)選字以及文法選字，來讓產生的歌詞達到有押韻或是有符合文法規則。也有印度的學者[27]是以坦米爾語為歌詞產生的語言，此語言通行於印度南部、斯里蘭卡東北部，這個名稱是來自梵文。其語言的特性為一個字母會對應到一個音素(Phoneme)，因此他們可以把所有字母轉換成 KNM 表示法，K 表示短母音，N 表示長母音，M 表示子音。例如 thA-ma-rai(蓮花)則表示為 N-K-N。其研究關鍵的想法是旋律產生的 K-N-M 樣式(pattern)要與產生歌詞的 K-N-M 樣式互相匹配。因此他們先把輸入的旋律序列利用 CRF(Conditional Random Field)把標記成 K-N-M 樣式序列。接下來利用 Dijkstra 最短路演算法從語料庫中找出與旋律的 K-N-M 樣式序列最相似的歌詞 K-N-M 樣式序列。但是他們的語料庫只有利用 Bi-gram 方式學習一個字與下一個字產生的機率，所以產生出來的歌詞沒有符合文法，並且歌詞中每一句也沒有意義上的承接。因此相同的研究者[26]利用 Knowledge-based 的方式改良上述產生歌詞時的缺點。

相反的，也有學者研究從歌詞自動產生樂曲的議題。Orpheus 系統[6][7]是為使用者輸入的日文歌詞產生歌曲。而日文是屬於高音重音語言 (Pitch Accent Language)，此語言的特性為重音的音節音高要比較高，因此不同的重音位置及使音節一樣，意義上就會不同，例如 ka'mi 為神的意思而 kami'為紙的意思。系統首先找出產生旋律可能的節奏型態，接下來再以人聲唱歌的音高範圍、和弦進程、低音伴奏以及日語的高音重音特性為限制條件，計算最大可能性(Maximum

Likelihood)所產生的旋律序列，進而完成一首歌曲。實驗結果在 1378 人中，有70.8%的人認為產生出來的歌是好聽、有吸引力的。隨後相同的學者又開發出了OrpheusBB 系統[10]，其中多加入讓使用者對於系統所產生的歌曲進行細部的修改並且系統會自動去維持音樂的一致性，例如使用者把系統產生的旋律做調整，可能會造成旋律與和弦有不和諧(Disharmony)的情況，因此系統會自動修正為符合旋律的和弦進程。在[25]研究中，則是利用 Orpheus 來做為中文語言的學習的工具。因為中文是跟廣東話一樣都是屬於聲調語言，而這種特性對很多非中文為母語的人學習起來是很困難的，因此他們把 Orpheus 系統原本是以日文特性為條件限制改成以中文特性為條件限制，讓系統產生的歌曲幫助外國人學習中文發音的聲調。

國立政治大學 National Chengchi University

第 3 章

詞式分析

一首流行音樂是由歌詞與樂曲構成的，所以歌詞與樂曲可說是一體兩面，因此我們採用音樂結構分析的架構，來進行歌詞的詞式分析，希望能夠找出歌詞主歌或副歌等等段落的排列方式。詞式分析分成四大步驟，依序為(1)計算行於行之間相似度 (2)自相似度矩陣建立；(3)重複樣式探勘；(4)詞式段落標記。

3.1. 問題描述

主歌	1. 孤獨站在這舞台 聽到掌聲響起來 我的心中有無限感慨 2. 多少青春不在 多少情懷已更改 我還擁有你的愛 3. 好像初次的舞台 聽到第一聲喝采 我的眼淚忍不住掉下來 4. 經過多少失敗 經過多少等待 告訴自己要忍耐
副歌	5. 掌聲響起來我心更明白 你的愛將與我同在 6. 掌聲響起來我心更明白 歌聲交會你我的愛
主歌	7. 孤獨站在這舞台 聽到掌聲響起來 我的心中有無限感慨 8. 多少青春不在 多少情懷已更改 我還擁有你的愛 9. 好像初次的舞台 聽到第一聲喝采 我的眼淚忍不住掉下來 10. 經過多少失敗 經過多少等待 告訴自己要忍耐
副歌	11. 掌聲響起來我心更明白 你的愛將與我同在 12. 掌聲響起來我心更明白 歌聲交會你我的愛
副歌	13. 掌聲響起來我心更明白 你的愛將與我同在 14. 掌聲響起來我心更明白 歌聲交會你我的愛

圖 3.1 鳳飛飛《掌聲響起》歌詞

一般來說歌詞對應的音樂只有人聲的音樂片段，而沒有其它前奏、間奏等等的音樂片段。音樂有其曲式，像是二段體、反覆體[37]等等，其中最常見的就是

二段體，此段體主要包含兩種片段，主歌與副歌。這兩種片段也是流行歌曲中人聲演唱的部分，因此歌詞也有其對應的結構。主歌是歌詞最主要的內容，也是故事的主幹，副歌通常會與主歌呈現情緒上的反差，達到歌曲的高潮也是歌曲的記憶點。有些歌詞還會包含橋段、前段與尾聲的片段。圖 3.1 為鳳飛飛《掌聲響起》的歌詞，左方的標號為行號，總共 14 行歌詞。其中段落 1 到 4 行與段落 7 到 10 行的內容為主歌片段。段落 5 到 6 行、段落 11 到 12 行與段落 13 到 14 行為副歌片段。此歌曲是一個標準的二段體，因為歌詞有包含主歌與副歌片段，並且主歌重覆兩次，副歌重覆三次。可以發現相同片段的歌詞內容會相仿，例如主歌中第 1 行與第 7 行，第 2 行與第 8 行等等，內容完全一模一樣，副歌的第 5、11 與 13 行內容也是完全一樣。圖 3.2 為趙傳《我很醜，可是我很溫柔》中副歌歌詞，其中第 5 行與第 13 行內容一模一樣，而第 6 行與第 14 行內容有些不同。

…

副歌

5. 我很醜 可是我很溫柔
6. 外表冷漠 內心狂熱 那就是我
7. 我很醜 可是我有音樂和啤酒
8. 一點卑微 一點懦弱 可是從不退縮

…

副歌

13. 我很醜 可是我很溫柔
14. 白天黯淡 夜晚不朽 那就是我
15. 我很醜 可是我有音樂和啤酒
16. 有時激昂 有時低首 非常善於等候

…

圖 3.2 《我很醜，可是我很溫柔》副歌歌詞

一首歌詞中的各種片段，依一定的邏輯排序，我們稱為**詞式**。因此本章節的目標為透過電腦自動分析出每種片段位於的行號區間以及標記上片段所屬的標籤。

3.2. 計算行與行之間相似度

我們針對歌詞的每一行分別從四個角度做特徵值擷取，分別是：(1)句字數結構序列；(2)拼音結構序列；(3)詞性結構序列；(4)聲調音高序列，以下分別依序詳述。

3.2.1. 句字數結構序列

每一行歌詞通常會由數個句子構成，句子之間用空白隔開，例如趙傳《我很醜，可是我很溫柔》中的第一行歌詞為：

每一個晚上　在夢的曠野　我是驕傲的巨人

可以看到此行有包含兩個空白符號，所以是由三個句子構成。每個句子由數個漢字所組成。一般來說主歌與主歌的之間相對應詞行的句數、空白數以及每一句的字數往往會差不多，例如趙傳《我很醜，可是我很溫柔》中主歌 1 的第一行與主歌 2 的第一行分別如下：

主歌 1 第一行：　每一個晚上　在夢的曠野　我是驕傲的巨人

主歌 2 第一行：　每一個早晨　在都市的邊緣　我是孤獨的假面

若將每一行根據每一句歌詞的字數轉換成一個序列，則主歌 1 第一行表示為序列<5, _, 5, _, 7>，主歌二的第一行表示為序列<5, _, 6, _, 7>，底線代表空白符號。可以看出只差在第二句的字數不同。而如果是主歌與副歌的詞行內容則有較大的差異，例如趙傳《我很醜，可是我很溫柔》的主歌 1 第一行與副歌 1 的第一行：

主歌 1 第一行：　每一個晚上　在夢的曠野　我是驕傲的巨人

副歌 1 第一行：　我很醜　可是我很溫柔

將主歌 1 第一行表示為序列<5, 5, 7>，副歌 1 第一行表示為序列<3, 5>，發現句字數以及句數都有差異。因此總共 N 行歌詞的 $L = l_1,l_2,...,l_N$，中的任一行 l_i 表示成句字數的序列 $Sen_i = <s_1, s_2,...,s_M>$， $s_j \in$ {句字數, _ }，底線代表空白符號。

3.2.2. 拼音結構序列

我們發現主歌與主歌相對應的詞行，雖然字面上的漢字不一樣，但是有時候發音會有相似的情況，例如五月天《擁抱》裡主歌 1 第一行與主歌 2 第一行的內容分別如下：

主歌 1 第一行：　脫下長日的假面　奔向夢幻的疆界

主歌 2 第一行：　隱藏自己的疲倦　表達自己的狼狽

兩行的歌詞的總字數、空白數都一樣，並且可以看到有些字詞的發音有些相似，像是「下」與「藏」,「面」與「倦」,「奔」與「表」,「疆」與「狼」。根據**拼音相似度**的想法我們發現有學者[38]在研究自動判斷形聲字的注音，其中他們有請語言學家一同訂定注音的拼音相似度。

注音分為拼音及聲調兩部分，其中拼音又分成聲母與韻母兩部分。例如「我」字的拼音為「ㄨㄛ」,「ㄨ」為聲母,「ㄛ」為韻母。兩個字之間的拼音相似度 p 分別是將聲母對聲母、韻母對韻母的相似度相加除以二，$0 \le p \le 1$。例如「空」與「回」，拼音分別是「ㄎㄨㄥ」與「ㄏㄨㄟ」,「ㄎ」與「ㄏ」聲母的相似度為 0.9,「ㄨㄥ」與「ㄨㄟ」韻母的相似度為 0.5，兩字的拼音相似度為 0.9 + 0.5 / 2=0.7。

我們利用此方法看回前面五月天《擁抱》的兩行歌詞中相似的字詞，這些字詞的拼音相似度分別是：「下」與「藏」為 0.7，「面」與「倦」為 0.4，「奔」與「表」為 0.5，「疆」與「狼」為 0.4，可以看到這些字詞的拼音的確存在一定的相似度。

有學者[40]發現目前流行音樂的歌詞的作詞者，大部分還是創作有押韻的歌詞，例如黃鶯鶯《哭砂》的主歌 2 中所有句子押的韻腳集合為「ㄛ」與「ㄜ」，而副歌中所有句子押的韻腳集合為「ㄧ」。然而現今流行歌曲之唱法，並不特別要求咬字發音的精確，常常有**合韻**的現象。此學者整理了七個合韻的韻腳：(1)「ㄧ」與「ㄩ」; (2)「ㄣ」與「ㄥ」; (3)「ㄢ」與「ㄤ」; (4)「ㄛ」與「ㄜ」; (5)

「ㄊ」與「ㄡ」; (6)「ㄛ」與「ㄡ」; (7)「ㄝ」與「ㄟ」。其中第一、四、五項的合韻的韻腳在學者[38]定義的拼音相似度為零，因此我們將此三項的相似度調整為 0.1 用以表示有相似度存在。因此我們將歌詞 $L = l_1,l_2,...,l_N$，其中任一行 l_i 表示成拼音結構序列 $Pinyin_i$，$Pinyin_i = <p_1, p_2,...p_M>$，$p_j \in$ {拼音, _ }，$1 \leq j \leq M$，底線代表空白符號。例如 P_i=<ㄨㄟ, ㄏㄊ, _ , ㄎㄋ, ㄨㄛ, ㄒㄧㄠ>。

3.2.3. 詞性結構序列

對聯一般會講求對丈，也就是上下聯對應的字、詞需用同類詞性，例如名詞對名詞，形容詞對形容詞等等。而在歌詞中我們發現有有這樣的現象存在，例如趙傳《我很醜，可是我很溫柔》中主歌 1 的第一行與主歌 2 的第一行分別如下：

主歌 1 第一行： 每一個晚上 在夢的曠野 我是驕傲的巨人

主歌 2 第一行： 每一個早晨 在都市的邊緣 我是孤獨的假面

可以發現「晚上」對「早晨」是名詞對名詞，「夢」對「都市」是名詞對名詞，「曠野」對「邊緣」是名詞對名詞，「驕傲」對「孤獨」是形容詞對形容詞，「巨人」對「假面」是名詞對名詞。剛好此兩行也是一個互相對應的詞行，因此是一個可以尋找詞式的線索。因此我們將歌詞 $L = l_1,l_2,..,l_N$，其中任一行 l_i 表示成詞性結構序列 Pos_i，$Pos_i = <p_1,p_2,...p_M>$，$p_j \in$ {詞性, _ }，$1 \leq j \leq M$，底線代表空白符號。例如 T_i=<名詞, 動詞, _, 形容詞, 受詞>。

3.2.4. 聲調音高序列

學者楊蔭瀏[32]有提到字調與音樂關聯性，其中提到：『因為我國漢族語言文字中的平仄、四聲，它們本身就已包含著音樂上的旋律因素。每一個自各有高低升降的傾向；連接若干字構成歌句之時，前後單字互相制約，又蘊蓄著對樂句進行的一種大致上的要求。』由此可知漢字聲調與音樂旋律有互相影響的關聯，其中的關聯包含正字與倒字的關係。正字關係代表歌詞的聲調走勢與音樂的旋律有互相搭配，因此歌手在唱歌的時候可以正確的表達歌詞的意義。倒字關係

則是當歌詞的聲調走勢與音樂的旋律有牴觸時，歌手在唱歌的時候可能會傳達不正確的歌詞。例如圖 3.3(a)是一個旋律與聲調有正字關係的例子，旋律是 Do, Re, Mi, Do，歌詞是「兩隻老虎」，這就是我們一般兒歌的唱法。而圖 3.3(b)是一個旋律與聲調有倒字關係的例子，旋律是 So, Re, Mi, Do，歌詞依然是「兩隻老虎」，可是實際唱的時候，會聽起來像「亮紙老虎」，無法傳達出歌詞正確的意義。

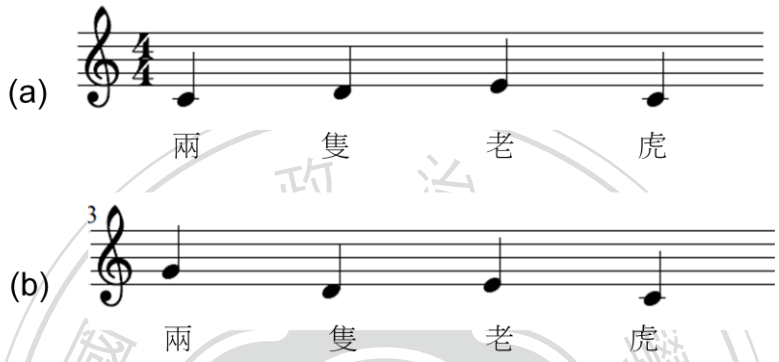


圖 3.3 聲調與旋律搭配的例子 (a)正字關係例子 (b) 倒字關係例子

因此我們利用學者徐富美等人[35]研究出預防倒字關係的旋律與聲調搭配的規則，如表 3.1，T1 與 T2 分別表示第一個漢字的聲調與下一個漢字的聲調，表格中每一格代表前後聲調所要搭配的音高走勢，其中任意包含高→低、水平與低→高。例如「回家」一詞，分別是二聲與一聲，搭配的旋律應該是由低到高。

表 3.1 聲調與音高走勢搭配規則

T1 \ T2	第一聲	第二聲	第三聲	第四聲	輕聲
第一聲	任意	高→低	高→低	高→低	任意
第二聲	低→高	任意	高→低	低→高	任意
第三聲	低→高	低→高	任意	低→高	任意
第四聲	低→高	高→低	高→低	任意	任意
輕聲	任意	任意	任意	任意	任意

接下來為了要訂定出確切的音高數值，我們參考[25]所定義的四聲實際的音高走勢，並且我們重新修改至符合表 3.1 避免倒字關係的規則，四聲所對應的實際音高走勢如圖 3.4，如果是輕聲，則我們將其音高走勢設定與前一個漢字的聲調一樣。

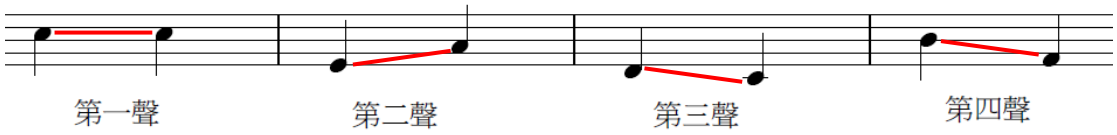


圖 3.4 四聲對應的實際音高走勢

因此我們根據聲調與旋律的特性，將歌詞 $L = l_1,l_2,...,l_N$，其中任一行 l_i 表示成聲調音高序列 $Tone_i$，$Tone_i = <t_1,t_2,...t_M>$，$t_j \in${音高}，$1 \le j \le M$。例如「我的吉他」的聲調序列為<3,輕聲, 2, 1> ，則聲調音高序列 $Tone_i$=<Re,Do^1, Re, Do^1, Mi, La, Do^2, Do^2> (Do^1 代表中央 Do，Do^2 代表高八度的 Do)。

3.3. 自相似度矩陣(Self-Similarity Matrix)建立

我們利用自相似度矩陣(Self-Similarity Matrix)的方式來去找出歌詞的重複樣式。自相似度矩陣的概念是參考自[4]的想法，Foote 當時用自相似度矩陣來做音樂的視覺化(Visualization)的研究，隨後有很多的研究是建立在自相似度矩陣上來做音樂結構分析[17]的研究。一個自相似度矩陣的建立，是將一個序列(Sequence)中的每個元素與其他元素做兩兩的相似度計算後，所形成的矩陣。如圖 3.5 為序列 $<e_1, e_2,..., e_N>$ 所形成的自相似度矩陣示意圖，矩陣中的每個元素(Element)代表由 $Sim(e_i, e_j)$(或是距離)函數計算得到的 e_i 與 e_j 相似度，其中 $1 \le i, j \le N$。

我們將歌詞 L 轉換成一個詞行的自相似度矩陣，$L = l_1,l_2,...,l_N$。矩陣中每個元素表示 l_i 行特徵與 l_j 行特徵的相似度 $Sim(l_i, l_j)$，$1 \le i, j \le N$。l_i 特徵值可能是上個小節中四種特徵值中的任一種。

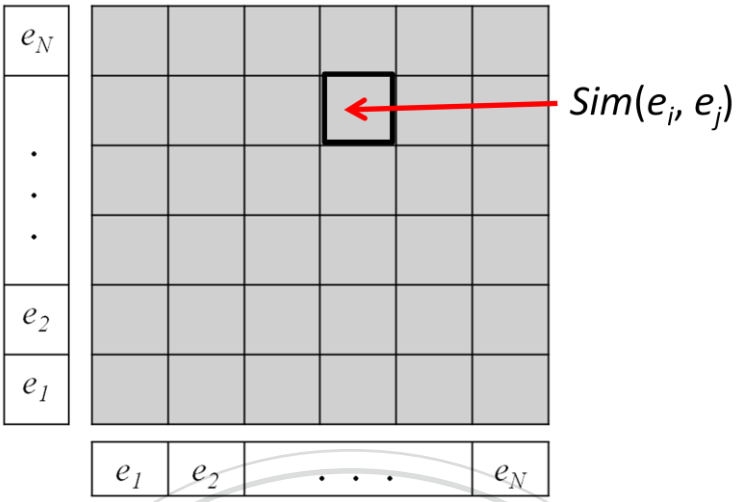


圖 3.5 自相似度矩陣示意圖

接下來我們定義詞行的 $Sim(l_i, l_j)$(或是距離)函數，我們利用了兩種方法：(1)詞行結構排比演算法；(2)DTW(Dynamic Time Warping)演算法。方法一是用來計算其中三種詞行特徵值：(1)句字數結構序列；(2)拼音結構序列；(3)詞性結構序列。方法二是用來計算詞行的聲調音高序列特徵值。

3.3.1. 詞行結構排比演算法

我們將兩行歌詞利用序列排比(Sequence Alignment)為基礎的方式尋找最佳的特徵序列對應分數，來當做兩行歌詞間的相似度。我們希望元素較少的詞行特徵序列，其所有元素都能夠與元素較多的詞行特徵序列做對應，因此特徵序列元素較多的詞行必定會對應到 Gap 因而產生懲罰分數(Penalty)，如此可以突顯出兩詞行結構上的差異。例如兩行歌詞，以拼音結構序列特徵為例分別是序列 $Pinyin_1$ = <A, B, C, D, E> 總字數 5，序列 $Pinyin_2$ = <F, G, H>總字數 3，因為兩行歌詞差了 2 個字，所以 $Pinyin_1$ 中的某兩個元素必定會對應到 Gap。在排比的過程中，我們希望考量到詞行與詞行之間 **Gap 分佈**與**空白結構**的特性，這兩種特性的相似度的高低關係，我們以下分別來探討。

[特性 1] 連續 Gap 與非連續 Gap

某些歌曲為求變化，即使相同是主歌的片段，在音樂上會以增加音符的變化，因此歌詞也會跟著增加。例如五月天《突然好想你》中的兩句歌詞分別如下：

主歌 1 第一行第一句： 最怕空氣突然安靜

主歌 2 第一行第一句： 我們像一首最美麗的歌曲

可以發現主歌 2 第一行第一句比主歌 1 第一行第一句總字數多了三個字，我們去聽音樂的旋律可以知道隨著音符增加的字詞是「最美麗」這三個字，而這三個字也就是一個詞彙，因此我們認為一般增加的音符往往都會**群聚**在一起。根據這樣的想法，我們參考生物資訊裡的同盟線性評分法(Affine Gap Penalty)的概念，讓行與行之間排比時，連續 Gap 的情況要比非連續 Gap 的情況相似度來的高。例如現在有兩行歌詞特徵序列，分別是序列 seq_1=<A, B, C>元素各數為 3，序列 seq_2 = <1, 2, 3, 4, 5>元素各數為 5。圖 3.6 為兩序列排比時兩種 Gap 分佈的示意圖。圖 3.6(a)為非連續 Gap 的情況，序列 seq_2 的元素 2, 4 對應到 Gap。圖 3.6(b)為連續 Gap 的情況，序列 seq2 的元素 2, 3 對應到 Gap。則圖 3.6(b)排比方式的相似度要比圖 3.6 (a)排比方式的相似度高。

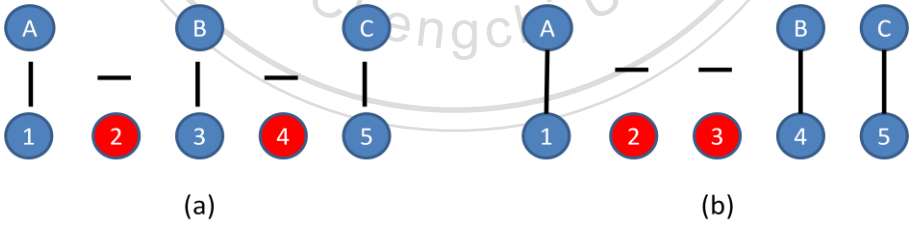


圖 3.6 序列排比的兩種 Gap 分佈示意圖

(a) 為非連續 Gap 情況，(b)連續 Gap 情況

[特性 2] 空白對應 Gap 與非空白對應 Gap

一行歌詞中會有包含數個句子，每個句子是用空白符號做區隔。一般來說主歌與副歌的歌詞的空白數目會有明顯的差異，所以我們認為空白 Gap 的懲罰分數要比非空白 Gap 的懲罰分數高。例如有兩行歌詞特徵序列，分別是序列 seq_1=<A, B, C>總元素為 3，序列 seq_2=<1, 2, _, 3> 特徵元素為 3(不包含空白)，如圖 3.7(a)為空白對應 Gap 的排比情況。現在將序列 seq_1 改成<A, B, _, C>，將 seq_2 改成<1, 2, _, 3, 4>，最後排比的情況為圖 3.7(b)，seq2 的元素 4 對應 Gap。則我們預期圖 3.7(b)排比情況的相似度要比圖 3.7(a)排比情況的相似度高。

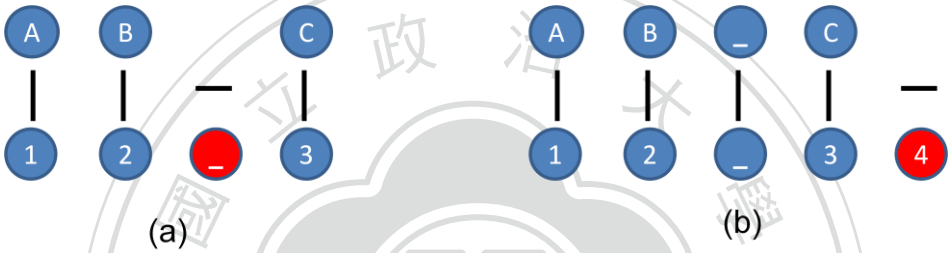


圖 3.7 空白對應 Gap 的排比情況

(a)空白對應 Gap 的排比情況 (b)特徵元素對應 Gap 的排比情況

為了要符合上述的兩個特性，我們將元素之間可能互相對應的關係整理成表 3.2。為了一般性，假定 Seq1 的特徵元素個數(不包含空白符號)比 Seq2 的特徵元素個數少，表格中打勾表示在排比過程中此兩種類型的元素可能會互相對應到，而打叉表示此兩種類型的元素完全不可能對應。例如為了要符合特性二，空白符號只可能對應到 Gap 或是另一個序列的空白符號，而空白符號不可能對應到特徵元素，因為沒有對應上的意義存在。

表 3.2 特徵元素對應關係表

Seq1 \ Seq2	Gap	_	特徵元素
Gap	✗	✔	✔
_	✔	✔	✗
特徵元素	✗	✗	✔

現在給定一個歌詞 L 總共 N 行，$L = l_1,l_2,\ldots,l_N$，其中任一行 l_i 表示成特徵序列 $F_i \in$ {句字數結構序列, 拼音結構序列, 詞性結構序列}，$1 \le i \le N$。現在將歌詞 L 中任兩詞行的特徵序列 F_i 與 F_j，且 F_i 與 F_j 屬於相同的特徵序列，$1 \le i, j \le N$ 作詞行結構排比計算。為了一般性，設序列 F_i 的特徵元素個數(不包含空白)比序列 F_j 的特徵元素個數少，則兩序列的相似度 $Sim\,(F_i, F_j)$定義為(3.1)：

$$Sim(F_i, F_j) = \frac{S(|F_i|, |F_j|)}{|F_i|} \tag{3.1}$$

其中 $|F_i|$為特徵序列長度(包含空白符號)，S 為一個求兩序列排比最佳相似度的函數，並且除以$|F_i|$做正規化。函數 S 的一般性遞迴關係式定義為(3.2)與(3.3)：

$$S(a,b) = \begin{cases} max\begin{cases} S(a-1,b-1) + c(F_i[a], F_j[b]) \\ G(a,b) \end{cases}, & if\ F_i[a] \wedge F_j[b] \in f \\ S(a-1,b-1), & if\ l_i[a] = \_ \wedge l_j[b] = \_ \\ S(a-1,b) + P_s, & if\ l_i[a] = \_ \wedge l_j[b] \in f \\ S(a,b-1) + P_s, & if\ l_i[a] \in f \wedge l_j[b] = \_ \end{cases} \tag{3.2}$$

$$G(a,b) = max\begin{cases} S(a,b-1) + P_o \\ G(a,b-1) + P_e \end{cases} \tag{3.3}$$

其中(3.2)的 $F_i[a]$代表特徵序列 F_i 的第 a 個元素，$F_j[b]$代表特徵序列 F_j 的第 b 個元素，c($F_i[a]$, $F_j[b]$)代表 $F_i[a]$與 $F_j[b]$兩元素的元素成本分數。P_s 表示空白對應 Gap 的懲罰分數。(3.3)中 G 代表序列 F_j 最後一個元素對應 Gap 的情況下，其最佳成本分數。P_o 表示第一個 Gap 產生的懲罰分數，P_e 表示第二個 Gap(含)之後產生的懲罰分數。函數 S 與函數 G 的初始關係式定義為(3.4)~(3.7)：

$$S(0,0) = 0 \tag{3.4}$$

$$S(a,0) = -\infty, \forall a \in \{1,2,\ldots,|F_i|\} \tag{3.5}$$

$$S(0,b) = P_o + (b-1) * P_e, \forall b \in \{1,2,\ldots,|F_i|\} \tag{3.6}$$

$$\mathrm{G(a,c)} = -\infty, \forall c \in \{b | F_j[b] = \_\}, \forall a = \in \{0,1,\ldots,|F_i|\} \tag{3.7}$$

其中(3.7)的 c 表示在特徵序列 F_j 中所有出現空白的位置索引。若演算法中的懲罰分數滿足$|\boldsymbol{P}_s| > |\boldsymbol{P}_o| > |\boldsymbol{P}_e|$，則這樣的參數設定就已經將之前探討的特性 1 與特性 2 考慮到詞行結構排比演算法中。因為特徵序列其中的特徵元素有三種可能，接下來我們針對不同的特徵元素定義(3.2)中特徵元素之間的成本函數 c。

■ **句字數特徵元素**

輸入兩個句字數特徵元素 $s_i,s_j \in N^+$，則兩個句字數元素的成本為：

$$c(s_i, s_j) = |s_i - s_j| \tag{3.8}$$

由於(3.8)的成本是屬於距離的計算方式，因此最後計算完成的矩陣會是一個自距離矩陣(Self-Distance Matrix)，為了一致性我們將 SDM 經由轉換成一個 SSM，轉換方法如(3.9)：

$$SSM(i,j) = 1 - \frac{SDM(i,j)}{max(SDM)}, \forall i,j \in \{1,2,\dots,N\} \tag{3.9}$$

其中 *max(SDM)*表示在 SDM 中的最大值。

■ **拼音特徵元素**

輸入兩個拼音特徵元素 $p_i,p_j \in$ {拼音}，關於拼音之間相似度的定義，我們參考學者張嘉惠[38]等人有請語言學家定義了拼音之間的相似度，我們為了考慮歌詞通押的特性，因此有將原本定義的相似度做調整，將有通押的韻母，相似度設定為 0.1。聲母與韻母相似度都介於 0 到 1 之間。兩個拼音特徵的成本為(3.10)：

$$c(p_i, p_j) = \frac{\text{聲母相似度} + \text{韻母相似度}}{2} \tag{3.10}$$

如果 p_i 與 p_j 剛好位於句子的最後一個拼音，則 $c(p_i, p_j) = 1$，如此是為了加重韻腳考量的權重。

■ **詞性特徵元素**

輸入兩個詞性特徵元素 $t_i,t_j \in$ {詞性}，則兩個詞性元素的成本為(3.11)：

$$c(t_i,t_j) = \begin{cases} 1, & if\ t_i = t_j \\ 0, & otherwise \end{cases} \qquad (3.11)$$

表 3.3 為一個詞行結構排比回溯的例子，這裡以詞性結構序列為例，輸入的兩條特徵序列 T_1,T_2，T_1=<主詞, 動詞, 受詞>、T_2=<主詞, 動詞, _, 受詞>，參數設定 P_s=-1，P_o=-0.5，P_e=-0.25。表 3.3(a)為式(3.2)中 S 函數利用填表法(Tabular Method)求解的累積成本矩陣(Accumulated Cost Matrix) S_M。表 3.3(b)為式(3.3)中 G 函數的累積成本矩陣 G_M (此矩陣為的意義為特徵元素較多之序列結尾對應 Gap 的累積成本)。表 3.3(a)與表 3.3(b)中灰色格子代表回溯的座標，紅色箭頭為回溯的順序，S_M(3, 5)為回溯起點。兩條序列的相似度為 1.5 / 3 = 0.5。最後排比的結果以(T1 元素, T2 元素)配對表示為一序列<(主詞, 主詞), (動詞, 動詞), (Gap, _), (Gap, 動詞), (受詞, 受詞)>。

表 3.3 詞行結構回溯過程

	T_2	**主詞**	**動詞**	**_**	**動詞**	**受詞**
T_1	0	-0.5	-0.75	-1	-1.25	-1.5
主詞	-∞	1	0.5	-0.5	-1	-1.25
動詞	-∞	-∞	2	1	0.5	0.25
受詞	-∞	-∞	-∞	-∞	1	1.5

(a)

	T_2	**主詞**	**動詞**	**_**	**動詞**	**受詞**
	-∞	-0.5	-0.75	-∞	-1.25	-1.5
Gap	-∞	-∞	0.5	-∞	-1	-1.25
Gap	-∞	-∞	-∞	-∞	0.5	0.25
Gap	-∞	-∞	-∞	-∞	-∞	0.5

(b)

最佳化的排比路徑回朔演算法如圖 3.8。

Algorithm Optimal Alignment Path

Input: Table S, Table G

Output: Path P /* 序列 Path 儲存 l_i 與 l_j 的元素對應序號 Tuple */

```
Procedure PathBackTracking()
    A←|l_i|, B←|l_j|
    P←TrackingS(A, B)
    return P

Procedure TrackTableS(A, B)
    if A = 0 and B = 0 then
        return P ← < ∅ >
    if l_i[A] = _ and l_j[B] = _ then
        P ← TrackTableS(A – 1, B – 1)
        return P.append( (A, B) )
    else if l_i[A] = _ then
        P ← TrackTableS(A – 1, B)
        return P.append( (A, -1) )   /* -1 代表對應到 Gap */
    else if l_j[B] = _ then
        P ← TrackTableS(A, B – 1)
        return P.append( (-1, B) )
    else
        if TableS[A, B] = TableG[A, B] then
            TrackTableG(A, B)
        else
            P ←TrackTableS(A – 1, B – 1)
            return P.append( (A, B) )

Procedure TrackTableG(A, B)
    if   TableG[A, B] = TableS[A, B – 1] + P_o then
            P ←TrackTableS(A, B – 1)
    else
            P ←TrackTableG(A, B – 1)
    return P.append( (-1, B) )
```

圖 3.8 最佳化的排比路徑回朔演算法

3.3.2. DTW (Dynamic Time Warping)演算法

DTW 是用來比對兩個與時間相關(Time Dependent)序列之間的距離，此方法是一個非線性的距離計算方法，最早是出現在 Spoken Word Recognition 問題的研究上[28]。由於旋律也是與時間相關的序列，並且有些旋律即使經過改編，我們人還是聽得出來原旋律，而 DTW 的非線性比對的特性就很適合比對兩條旋律線的相似程度。例如圖 3.9(a)為原旋律，圖 3.9(b)為改編後的旋律，兩旋律人聽起來有點像，實際上用 DTW 計算兩旋律隨時間對應關係如紅線，有找出圖 3.9(b)多餘音符的對應。因此我們將 DTW 應用在計算兩條聲調音高序列的距離。

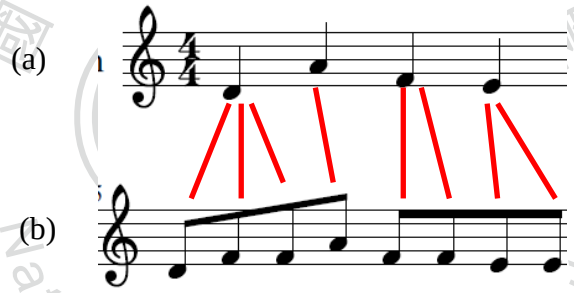


圖 3.9 音高旋律用 DTW 比對的例子

現在給定一個歌詞 L 總共 N 行，L = $l_1,l_2,\ldots,l_N$，其中任一行 l_i 表示成聲調音高特徵序列 T_i，$1 \le i \le N$。現在將歌詞 L 中任兩詞行的特徵序列 T_i 與 T_j，$1 \le i, j \le N$ 作 DTW 計算。兩條聲調音高特徵序列的距離 $Dist(T_i, T_j)$為：

$$Dist(T_i, T_j) = \frac{D(|T_i|, |T_j|)}{|T_i| + |T_j|} \qquad (3.12)$$

其中 $|T_i|, |T_j|$為各自音高特徵序列長度，D 為一個求兩序列 T_i, T_j 之間最短 DTW 距離的函數，最後除以兩序列長度的加總做正規化。

在 DTW 演算法中我們還需要設定 Step Size 類型，由於我們認為歌詞中若是兩行對丈的詞行，字數不會差距到兩倍，因此對於字數差距到兩倍的詞行，我們

直接認為此兩行歌詞的距離為 1。

因此，我們選擇的 Step Size 類型如圖 3.10，如此可以確保兩行詞行的字數差距超過兩倍時，DTW 距離必定會形成無限大。

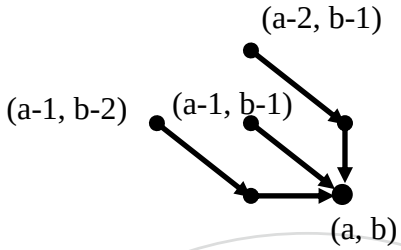


圖 3.10 DTW Step Size 類型

在這樣的 Step Size 類型之下，兩序列 T_i,T_j 間最短 DTW 距離函數 D 一般性的遞迴關係如(3.13)、初始化關係如(3.14)~(3.16)：

$$D(a,b) = min\begin{cases} D(a-1,b-2) + w_1 * \left(c\left(T_i[a], T_j[b-1]\right) + T_i[a], T_j[b]\right) \\ D(a-1,b-1) + w_2 * T_i[a], T_j[b]) \\ \mathrm{D(a-2,b-1)} + w_3 * \left(c\left(T_i[a-1], T_j[b]\right) + T_i[a], T_j[b]\right) \end{cases} \tag{3.13}$$

$$D(-1,-1) = 0 \tag{3.14}$$

$$D(a,b) = \infty, \forall a \in \{-1,-2\} \text{ and } \forall \mathrm{b} \in \left[1, \left|\mathrm{T_j}\right|\right] \tag{3.15}$$

$$D(a,b) = \infty, \forall a \in [1, |\mathrm{T_i}|] \text{ and } \forall \mathrm{b} \in \{-1,-2\} \tag{3.16}$$

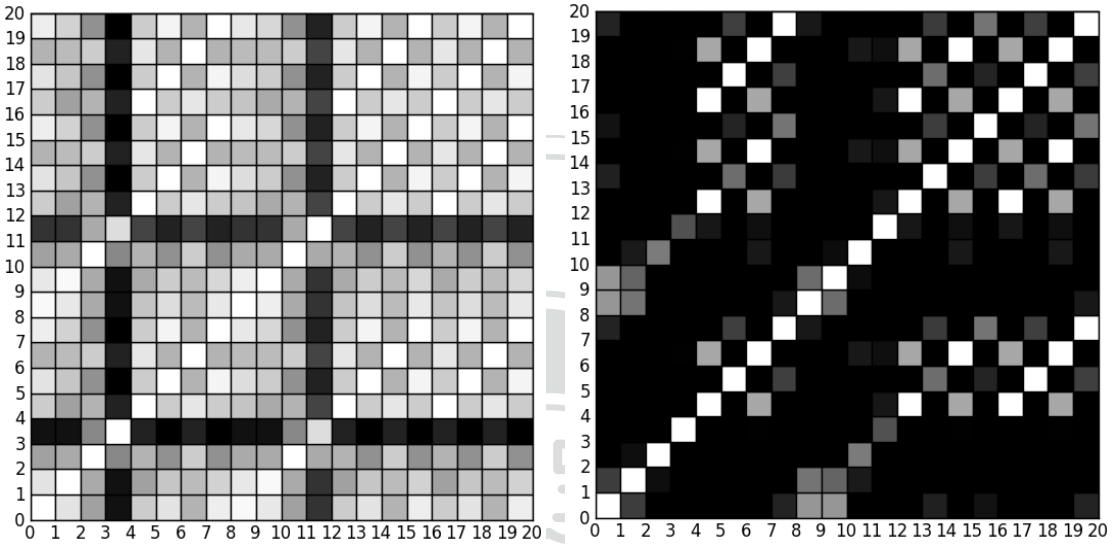
(3.13)中 $T_i[a]$代表聲調音高序列 T_i 的第 a 個元素，$T_j[b]$代表聲調音高序列 T_j 的第 b 個元素。w_1,w_2,w_3 代表不同路徑的權重，越長的路徑權重通常越高，如此可以顯示出總字數的差異性。c 為兩音高的成本函數其定義為(3.17)：

$$c(t_i, t_j) = |t_i - t_j| \tag{3.17}$$

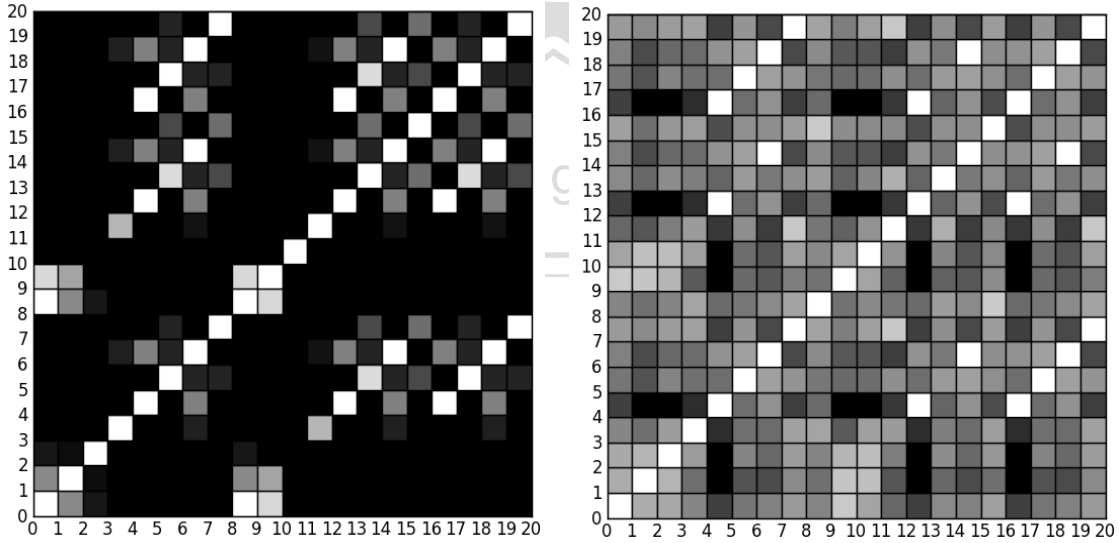
由於式(3.17)的定義為距離，因此最後計算完成的矩陣會是一個自距離矩陣，為了一致性我們將 SDM 同樣經由式(3.9)轉換成一個自相似度矩陣。

3.3.3. 線性組合 SSM

圖 3.11 為趙傳《我很醜，可是我很溫柔》歌詞轉成四種特徵序列，經由詞行結構排比或 DTW 計算而產生的四種 SSM。歌詞總共 20 行，所以產生的四種 SSM 大小為 20 乘 20。SSM 中顏色越白代表兩行之間的相似度越高，反之越黑代表相似度越低，相似度最大值為 1，最小值為 0。



(a) 句字數結構序列特徵 SSM　　(b) 拼音結構序列特徵 SSM



(c) 詞性結構序列特徵 SSM　　(d) 聲調音高序列特徵 SSM

圖 3.11《我很醜，可是我很溫柔》四種特徵值 SSM

由於我們定義的元素相似度(或距離)是對稱型的函數，因此四種特徵的 SSM 必定是一個對稱矩陣(Symmetric Matrix)。趙傳《我很醜，可是我很溫柔》歌詞的第 1 行與第 9 行內容如下：

第 1 行：每一個晚上 在夢的曠野 我是驕傲的巨人

第 9 行：每一個早晨 在都市的邊緣 我是孤獨的假面

圖 3.11(c) 詞性結構序列特徵 SSM(1, 9)為全白，相似度為 1，其他三種特徵的 SSM(1,9)只有部分相似度(灰色)。而第 3 行與第 11 行的內容如下：

第 3 行：在鋼筋水泥的叢林裡 在呼來喚去的生涯裡

第 11 行：在一望無際的舞台上 在不被瞭解的另一面

圖 3.11(c) 詞性結構序列特徵 SSM(3, 11)為黑色，相似度為 0，而其他三種特徵的 SSM(3, 11)則有呈現部分相似度，因此不同特徵可以找出各自獨有的詞行相似度。最後，我們將四種做線性組合如(3.18)：

$$SSM_{hybrid} = \alpha * SSM_{sen} + \beta * SSM_{pinyin} + \gamma * SSM_{pos} + \delta * SSM_{tone} \qquad (3.18)$$

其中 SSM_{hybrid} 代表線性組合 SSM，SSM_{sen} 代表句字數結構序列 SSM，SSM_{pinyin} 代表拼音結構序列 SSM，SSM_{pos} 代表詞性結構序列 SSM，SSM_{tone} 代表聲調音高序列 SSM。圖 3.12《我很醜，可是我很溫柔》歌詞的四種特徵值 SSM 做線性組合的結果，$\alpha = \beta = \gamma = \delta = 0.25$。

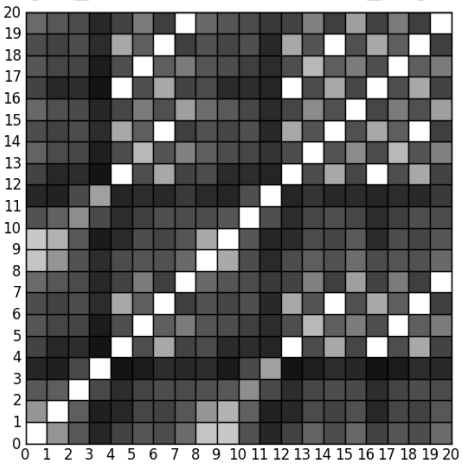


圖 3.12 線性組合 SSM

3.4. 重複樣式探勘

在 3.2 節問題定義我們提到希望透過重複樣式的方式減少計算量，在本節我們希望在線性組合 SSM_{hybrd} 之上做**重複樣式探勘**(Repeating Pattern Mining)來達到段落分群的目的。

[定義 3.1] 段落(Block)

給定一個歌詞 L，一個段落是由連續詞行所構成 b = (s, e)，$1 \le s \le e \le N$，s 代表段落的起始行號，e 代表段落的結束行號，N 為歌詞總行數，段落的長度 len(b) = (e – s + 1)。

[定義 3.2] Path

給定一個歌詞生成的 SSM_{hybrid}，在 SSM_{hybrid} 中任意 $SSM_{hybrid}(i, j)$到 $SSM_{hybrid}(i + L, j + L)$為一條 45 度的斜線樣式，稱為一條 Path p，$0 \le L \le N – \max(i, j)$，N 為歌詞總行數，函數 max(i, j)為在 i, j 中取較大的數值，p 長度 len(p)=L + 1。每條 Path 會對應到兩個段落 $b_1=(i, i+L)$, $b_2 = (j, j + L)$。例如某 SSM_{hybrid} 為一 10*10 的矩陣，Path p 從 $SSM_{hybrid}(2, 3)$到 $SSM_{hybrid}(7, 8)$，len(p)=6，對應到段落(2, 7)與段落(3, 8)。

[定義 3.3] Path 分數(Path Score)

給定一條 Path p = $SSM_{hybrid}(i, j)$到 $SSM_{hybrid}(i + L, j + L)$，其 Path 分數 Ps 定義為(3.19)：

$$Ps(p) = \sum_{\tau=0}^{L} SSM_{hybrid}\,(i + \tau, j + \tau) \qquad (3.19)$$

國立政治大學 National Chengchi University

我們先來探討歌詞詞式與 SSM_{hybrid} 之間的關聯。圖 3.13 左方為葉蒨文《瀟灑走一回》的歌詞右方為其歌詞產生的 SSM_{hybrid}。左方的歌詞總共 17 行，段落(1, 2)與段落(7, 8)同為主歌。段落(3, 6)、段落(9, 12)與段落(13, 16)同為副歌。段落(17, 17)為尾聲。主歌重複兩次，副歌重複三次而尾聲則沒有重複。

右方的 SSM_{hybrid} 可以看到有很多全白的 Path 存在，例如 p = SSM_{hybrid}(1, 7)到 SSM_{hybrid}(6, 12)，len(p) = 6，Ps(p) = 1，也代表段落(1,6)與段落(7, 12)的相似度為 1。由此可以推論在 SSM_{hybrid} 對角線上一條任意長度的 Path p_n，其 Ps(p_n)必定為 1，因為相同段落的相似度必定為 1。

因此，主歌為段落(1, 2)與段落(7, 8)構成，可以發現反應在 p1 = SSM_{hybrid}(1, 7)到 SSM_{hybrid}(2, 8)，len(p1)=2。副歌為段落(3, 6)、段落(9, 12)與段落(13, 16)構成，反應在 p2 = SSM_{hybrid}(3, 9)到 SSM_{hybrid}(6, 12)、p3 = SSM_{hybrid}(3, 13)到 SSM_{hybrid}(6, 16)與 p4 = SSM_{hybrid}(9, 13)到 SSM_{hybrid}(12, 16)，有三條長度為 4 的 Path(藍色圈)。在這個例子當中，相同主歌或副歌內，兩兩段落的相似度皆為 1。

1. 天地悠悠 過客匆匆 潮起又潮落
2. 恩恩怨怨 生死白頭 幾人能看透
3. 紅塵呀滾滾 痴痴呀情深 聚散終有時
4. 留一半清醒 留一半醉 至少夢裡有你追隨
5. 我拿青春賭明天 你用真情換此生
6. 歲月不知人間 多少的憂傷 何不瀟灑走一回
7. 天地悠悠 過客匆匆 潮起又潮落
8. 恩恩怨怨 生死白頭 幾人能看透
9. 紅塵呀滾滾 痴痴呀情深 聚散終有時
10. 留一半清醒 留一半醉 至少夢裡有你追隨
11. 我拿青春賭明天 你用真情換此生
12. 歲月不知人間 多少的憂傷 何不瀟灑走一回
13. 紅塵呀滾滾 痴痴呀情深 聚散終有時
14. 留一半清醒 留一半醉 至少夢裡有你追隨
15. 我拿青春賭明天 你用真情換此生
16. 歲月不知人間 多少的憂傷 何不瀟灑走一回
17. 歲月不知人間 多少的憂傷 何不瀟灑走一回

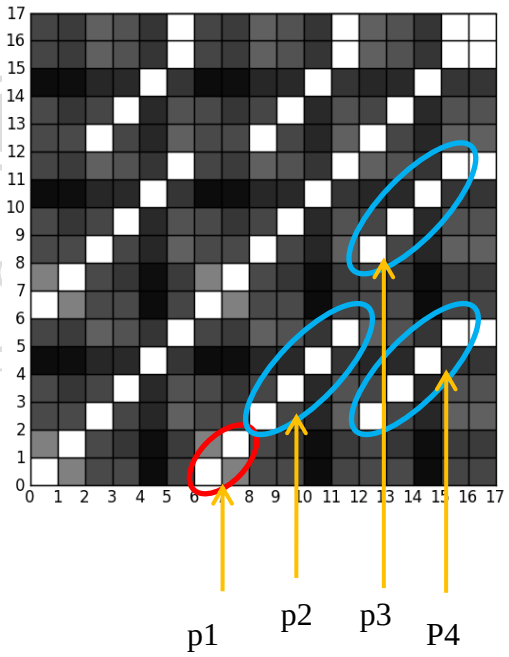


圖 3.13《瀟灑走一回》歌詞原文與其 SSM_{hybrid}

圖 3.14 為趙傳《我很醜，可是我很溫柔》歌詞產生的 SSM_{hybrid}，在其歌詞的詞式，主歌為段落(1, 4)與段落(9, 12)構成，副歌為段落(5, 8)、段落(13, 16)與段落(17, 20)構成。p2 = SSM_{hybrid}(1, 9)到 SSM_{hybrid}(4, 12)，Ps(p2) < 1，這表示即使段落(1, 4)與段落(9 ,12)同為主歌，但兩段落可能也只有**部份相似度**(灰色)。即使如此，p3 的整體的顏色還是較 p2 深，這表示 Ps(p2) > Ps(p3)。相同的 p1 = SSM_{hybrid}(5, 13)到 SSM_{hybrid}(8, 16)，段落(5, 8)與段落(13, 16)雖然同為副歌，可是歌詞中第 6 行與第 14 行、第 8 行與第 16 行也是只有部分相似度存在，相似度呈現一種**跳躍式的樣式**。圖 3.15 為此兩段落的歌詞對照。

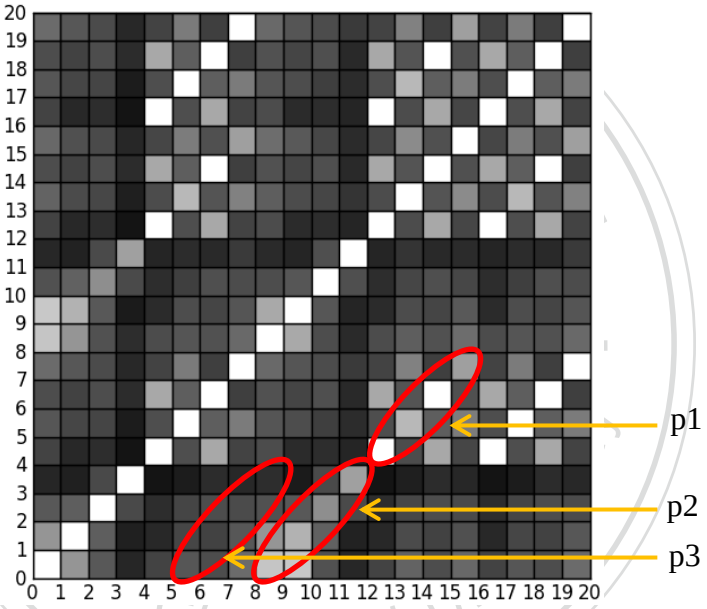


圖 3.14 《我很醜，可是我很溫柔》SSM_{hybrid}

5.	我很醜 可是我很溫柔	13.	我很醜 可是我很溫柔
6.	外表冷漠 內心狂熱 那就是我	14.	白天黯淡 夜晚不朽 那就是我
7.	我很醜 可是我有音樂和啤酒	15.	我很醜 可是我有音樂和啤酒
8.	一點卑微 一點懦弱 可是從不退縮	16.	有時激昂 有時低首 非常善於等候

圖 3.15《我很醜，可是我很溫柔》段落(5, 8)與段落(13, 16)歌詞

辛曉琪《領悟》歌詞的副歌為段落(15, 22)、段落(32, 38)與段落(39, 46)構成。雖然同為副歌，可是段落(32, 38)長度為 7 行，而另外兩個段落長度為 8 行，有**長度上的不同**。圖 3.16 為此三段落的歌詞內容，段落(15,22)與段落(39, 46)比段落多了(32, 38)「別再為愛受苦」。

段落(15,22)與段落(39, 46)內容

啊 多麼痛的領悟
你曾是我的全部
只是我回首來時路的每一步
都走的好孤獨
啊 多麼痛的領悟
你曾是我的全部 只願你掙脫情的枷鎖
愛的束縛 任意追逐
別再為愛受苦

段落(32, 38)內容

啊 多麼痛的領悟
你曾是我的全部
只是我回首來時路的每一步
都走的好孤獨
啊 多麼痛的領悟
你曾是我的全部 只願你掙脫情的枷鎖
愛的束縛 任意追逐

圖 3.16《領悟》的副歌內容

[定義 3.4] 樣式段落(Pattern Block)

給定一歌詞 L，樣式段落為一任意長度的段落 $B_p=(s_p, e_p)$，$1 \le s_p \le e_p \le N$，N 為歌詞總行數。樣式段落長度 $len(B_p) = (e_p – s_p + 1)$。例如某歌詞為 10 行，$B_p=(1, 10)$，$len(B_p)=10$。樣式段落 $B_p=(1, 1)$ ，$len(B_p) = 1$。

[定義 3.5] 實體 Path (Instance Path)

給定一個 SSM_{hybrid} 與樣式段落 $B_p = (s_p, e_p)$，對於任意一條 Path $p_n = SSM_{hybrid}(i, j)$到 $SSM_{hybrid}(i + L, j + L)$滿足 $s_p \le i \le e_p$ 且 $e_p < j \le N$ 且 $0 \le L \le e_p – \max(i, j)$，則此 Path 稱為樣式段落 B_p 的一條實體 Path。

[定義 3.6] 實體 Path 矩陣(IPM)

給定一個 SSM_{hybrid} 與樣式段落 $B_p = (s_p, e_p)$，其所有實體 Path 可以出現範圍矩陣為 SSM_{hybrid} 的一個子矩陣(submatrix)IPM = $SSM_{hybrid}(s_p \sim e_p, e_p \sim N)$，N 為歌詞總行數。則 IPM 稱為段落 B_p 的實體 Path 矩陣。

[定義 3.7] 實體段落(Instance Block)

給定一個樣式段落 $B_p = (s_p, e_p)$與一條 B_p 的實體 Path p，則此 p 對應的兩個段落，其中一段落 $I = (s_I, e_I)$，滿足 $s_I > e_p$，則此段落 I 稱為樣式段落 B_P 的實體段落。

圖 3.17 為樣式段落 B_p=(5, 8)其實體 Path 矩陣、實體 Path 與實體段落說明的例子。圖 3.17 最左方紅框為一個樣式段落 B_p=(5, 8)，右方在對角線下的矩形紅框為 B_p 的實體 Path 矩陣，因此 Path1 到 Path4 皆為 B_p 的實體 Path，而 Path 5、Path6 不為 B_p 的實體 Path。Path 1 與 Path 2 的實體段落皆為(10, 11)，長度為 2。Path 3 的實體段落為(17, 20)，長度為 4。Path 4 的實體段落為(14, 14)，長度為 1。

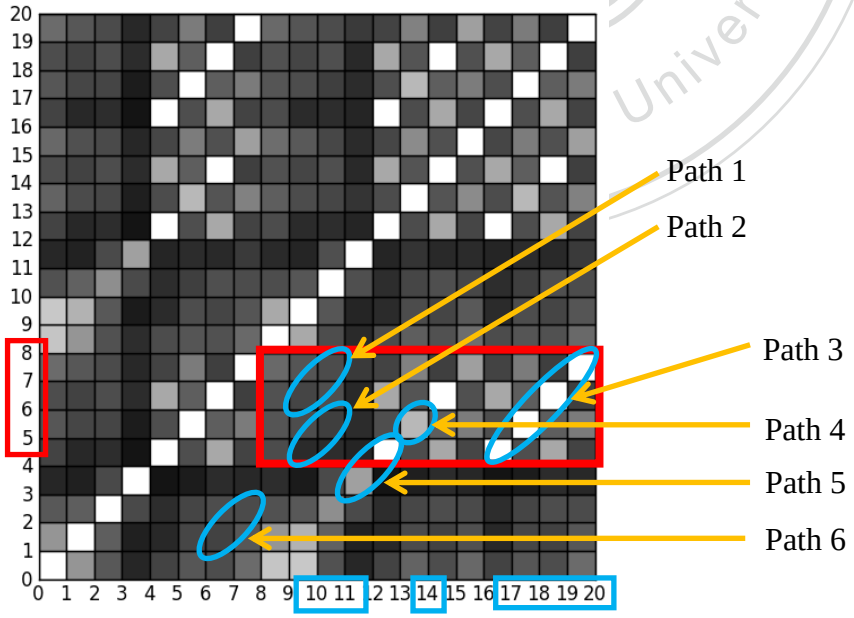


圖 3.17 樣式段落(5, 8)的實體段落例子

樣式段落 $B_p=(s_p, e_p)$與其實體 Path 矩陣 $IPM=SSM_{hybrid}(s_p\sim e_p, e_p\sim N)$，N 為歌詞總行數。我們希望對 B_p 找到其多條的實體 Path ，$p_1, p_2,\ldots,p_M$，這些實體 Path 分數總和最大，$\max(\sum_{i=1}^{M} Ps(p_i))$，並且這些實體 Path 滿足兩個限制：(1)對應的實體段落 $I_1, I_2, \ldots, I_M$，兩兩實體段落的段落範圍**不能有重疊**；(2) 對應的實體段落 $I_1, I_2, \ldots, I_M$，$\sum_{j=1}^{M} len(I_j) = N - e_p$。我們參考[31]，設計了一個 Instance Path Search 演算法，在樣式段落 B_p 的實體 Path 矩陣來做實體 Path 搜尋，Instance Path Search 的一般性的遞迴關係為(3.20)、(3.21)，初始化關係為(3.22)：

$$Score(i,j) = IPM(i,j) + \max\left(Score(k,j-1) + Ts(k,i)\right) \quad (3.20)$$

$$Ts(k,i) = \begin{cases} S_{diag}, \text{if } i = k+1 \\ S_{other}, \text{otherwise} \end{cases} \quad (3.21)$$

$$Score(i,0) = IPM(i,0), \forall i \in \{0,1,\ldots,len(B_p)-1\} \quad (3.22)$$

其中 i, j 出現的位置皆在樣式段落 B_p 的實體 Path 矩陣內。此演算法的想法來自 Viterbi Algorithm，我們可以把 i 想像成狀態(state)，j 則是時間，所以總共有 $len(B_p)$ 種狀態。因此 Score(i, j)代表在時間狀態 i 的在時間 j 時的最大可能性分數。IPM(i, j)可以想像成觀察分數(Observation Score)。Ts(k, i)可以想像成從狀態 k 到狀態 i 的轉移分數(Transition Score)。圖 3.18 為一個轉移分數的說明示意圖，實線箭頭代表 $Ts(k, i) = S_{diag}$ 的情況，虛線箭頭代表 $Ts(k, i) = S_{other}$ 的情況。我們將轉移分數符合 $S_{diag} > S_{other}$，則在狀態轉移會盡量走 45 度的方向。如此的設定是為了讓搜尋出來的實體 Path 可以完整，也就是實體 Path 的長度可以接近樣式段落 B_p 的長度，盡量不要出現太多零碎的實體 Path。

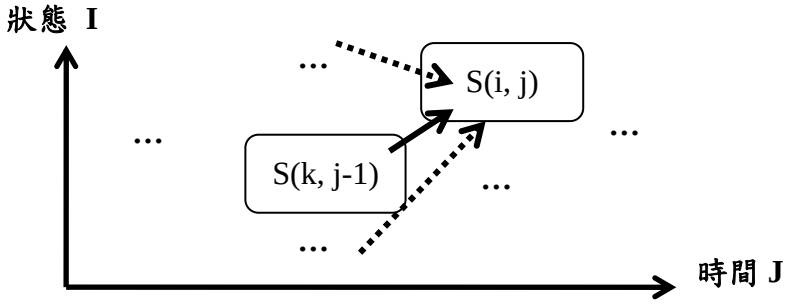


圖 3.18 轉移分數 Ts 示意圖

在此 Instance Path Search 演算法我們利用填表法求遞迴關係解。演算法虛擬碼如圖 3.19。

Algorithm Instance Path Search

Input: 樣式段落 $B_p=(s_p, e_p)$, IPM
Output: Score Matrix(分數累積矩陣)

Procedure IPS(B_p):

```
Score = matrix(len(Bp), (N – ep)) /*宣告一個大小為 len(Bp)*(N-ep)的矩陣 */
for i = 0 to len(Bp) - 1 : /*Score 矩陣初始化*/
    Score(i, 0) = IPM(i, 0)
for j = 1 to len(Bp) - 1:
    for i = 0 to len(Bp) – 1:
        score = 0
        for k = 0 to len(Bp) – 1:
            if k + 1 = i and Score(k, j - 1) + Pdiag > score:
                score = IPM(i, j) + Score(k, j - 1) + Pdiag
            else:
                score = IPM(i, j) + Score(k, j - 1) + Pother
        Score(i, j) = score
```

圖 3.19 Instance Path Search 演算法

我們利用經過 Instance Path Search 演算法的計算後，我們可以得到實體 Path，$p_1, p_2, \ldots, p_M$ 的最佳 Path 分數加總，接下來我們要做 Instance Path Tracking，來得到 $p_1, p_2, \ldots, p_M$，演算法如圖 3.20。

Algorithm Instance Path Tracking

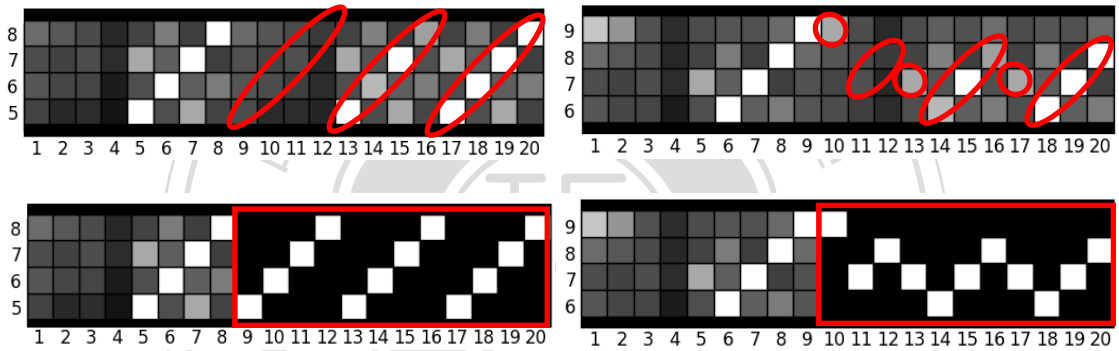
Input: Score Matrix
Output: Instance Path List

```
Procedure InstancePathTracking(Score):
    IPL = <∅>
    nowState = argmax(Score(sp~ep, N)) /* N 為歌詞行數，取分數最大的狀態*/
    path = < (nowState, N) >

    for j = (N -1) to (ep + 2)
        comeScoreList = ComeScore(nowState, j, Score)
        prevState = argmax(comeScoreList)

        if prevState + 1 == nowState:
            path.append( (prevState, j) )
        else:
            IPS.append(path)
            path = < (prevState, j) >

        nowState = prevState

    return IPL

Procedure ComeScore(nowState, j, Score):
    comeScoreList = <∅>
    for i = sp to ep :
        if i + 1 == nowState:
            comeScoreList.append(Score(i, j - 1) + Pdiag)
        else:
            comeScoreList.append(Score(i, j - 1) + Pother)
    return comeScoreList
```

圖 3.20 Instance Path Tracking 演算法

國立政治大學 National Chengchi University

圖 3.21 為兩個 Instance Path Tracking 在趙傳《我很醜，可是我很溫柔》所找出實體 Path 的例子。圖 3.21(a)下方為給定的樣式段落 B_p=(5, 8) Instance Path Tracking 在實體 Path 矩陣(紅色框)搜尋的結果，上方為 Instance Path Tracking 搜尋對應到的實體 Path，總共有 3 條長度為 4 的實體 Path。圖 3.21(b)下方為給定的樣式段落 B_p=(6, 9) Instance Path Tracking 在實體 Path 矩陣(紅色框)搜尋的結果，上方為 Instance Path Tracking 搜尋對應到的實體 Path，總共有 6 條，長度有 1、2 或 3。可以發現圖 3.21(b)找到的實體 Path 相較於圖 3.21 (a)來的多，並且長度都小於給定的樣式段落長度 4，如此也便為一個找出的實體 Path 較為零碎的例子。



(a) (b)

圖 3.21 Instance Path Tracking 的兩個例子

(a) 樣式段落 B_p=(5, 8) (b) 樣式段落 B_p=(6, 9)

針對此樣式段落 B_p 完成 Instance Path Tracking 之後，我們將得到的實體 Path $p_1, p_2, \ldots, p_M$，若這些 Path **同時滿足**：(1)$Ps(p_i) / len(p_i) \geq T_{sim}$，$1 \leq i \leq M$；(2) $len(p_i) \geq T_{sim}$，$1 \leq i \leq M$，則將其 Path 保留下來，$p_{k1}, p_{k2}, \ldots, p_{km}$，$km \leq M$，最後取出這些實體 Path 對應的實體段落，$I_{k1}, I_{k2}, \ldots, I_{km}$，稱之為 B_p 的**有效實體段落**，我們將樣式段落 B_p 與其有效實體段落稱為一個 **Family** $f(B_p)=\{B_p, I_{k1}, I_{k2}, \ldots, I_{km}\}$，一個 Family 也就相當於達到了段落分群的目的。

給定一個 B_p 我們可以找到其 Family，接著我們對各種可能的 B_p 產生其 Family，希望尋找最佳的 Family 組合方式，首先我們先建立一個 Family 內聚力矩陣做為搜尋最佳 Family 組合的資料結構。

給定一個 Family $f(B_p)$，其中所有的段落(包含樣式段落) $f(B_p)=\{b_1,b_2,\ldots,b_n\}$，任意兩兩段落 b_i, b_j 之間的相似度定義為(3.23)：

$$B_{sim}(b_i, b_j) = \max(Ps(P)) \qquad (3.23)$$

為了一般性設 $len(b_i) \le len(b_j)$，其中 P 為一個 Path 的集合，集合 P 是由輸入的 $b_i=(s_i, e_i)$, $b_j=(s_j, e_j)$所形成的子矩陣 $SSM_{hybrid}(s_i\sim e_i, s_j\sim e_j)$中，所有長度為 $len(b_i)$的 Path 構成。圖 3.22 為給定段落 $b_i=(4, 6)$, $b_j=(7, 10)$所形成的子矩陣，總共有兩條 Path，$len(Path1) = len(Path2) = 3$。最後 $B_{sim}(b_i, b_j)$選 Ps(Path1)、Ps(Path2)最大的分數做為 b_i, b_j 的段落間的相似度。

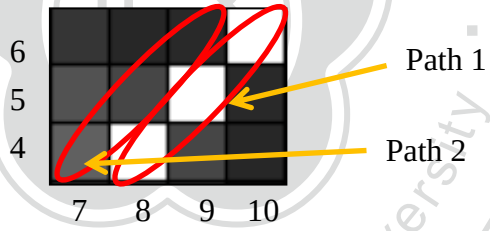


圖 3.22 有效實體段落間相似度例子

有了段落間的相似度 B_{sim}，我們將給定 Family $f(B_p)$的內聚力 Co 定義為(3.24)：

$$Co(f(B_p)) = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} B_{sim}(b_i, b_j)}{n*(n-1)/2} \qquad (3.24)$$

其中 n 為 Family $f(B_p)$中段落的總個數。

國立政治大學 National Chengchi University

現在給定一個 N 乘 N 的 SSM_{hybrid} 我們將產生所有滿足 $2 \le len(B_{pi}) \le N/2$ 的樣式段落 $B_{p1} = (s_{p1}, e_{p1})$, $B_{p2} = (s_{p2}, e_{p2}),\ldots,B_{pk} = (s_{pk}, e_{pk})$，$1 \le i \le k$，並且產生各自的 Family $f(B_{p1})$, $f(B_{p2})$,.., $f(B_{pk})$。最後形成一個 Family 內聚力矩陣 F：

$$F = \forall i \in \{1,2,\ldots,k\}: \ F\left(len(B_{pi}), s_{pi}\right) = Co(f(Bpi)) 。$$

圖 3.23 為一個趙傳《我很醜，可是我很溫柔》所形成的 Family 內聚力矩陣 F，橫軸數線代表 Family $f(B_{pi})$中 B_{pi} 的起始行號 s_{pi}，縱軸代表長度 $len(B_{pi})$，每一個元素代表 $f(B_{pi})$的內聚力，右方的白色倒三角形為無填值的區塊。F 中顏色樂白表示 $f(B_{pi})$的內聚力越高，反之越暗則 $f(B_{pi})$的內聚力越低，內聚力值介在[0, 1]。可以發現有很多內聚力為 0(黑色)的 $f(B_{pi})$，那就代表此 $f(B_{pi})$沒有重複樣式存在。

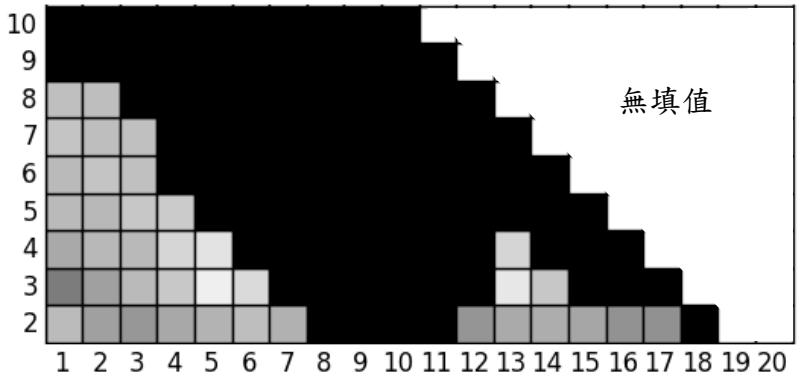


圖 3.23 趙傳《我很醜，可是我很溫柔》的 Family 內聚力矩陣

我們認為歌曲中最重要的就是**主歌**與**副歌**這兩種樣式，因此接下來我們利用 Family 內聚力矩陣 F 做任意兩兩 Family 的組合，其中兩個 Family $f(B_{pi})$, $f(B_{pj})$可以組合的條件為，滿足 $A = f(B_{pi}) \cup f(B_{pj})$，集合 A 中的兩兩段落之間**不能有範圍重疊**即可做組合。為了要取得最佳化的組合分數，我們將任兩個 Family $f(B_{pi})$與 $f(B_{pj})$的組合分數 CS 參考[17]定義為：

$$CS\left(f(B_{pi}), f(B_{pj})\right) = \frac{2 * precision * recall}{precision + recall} \qquad (3.25)$$

CS 的計算方式是源自 f-score，接下來定義 precision 與 recall 為(3.26)與(3.27)：

$$\text{precision}\left(f(B_{pi}), f(B_{pj})\right) = \text{Co}\left(f(B_{pi})\right) * \frac{\sum \text{len}(b_x)}{\sum \text{len}(b_x) + \sum \text{len}(b_y)} + \text{Co}\left(f(B_{pj})\right) * \frac{\sum \text{len}(b_y)}{\sum \text{len}(b_x) + \sum \text{len}(b_y)} \quad (3.26)$$

$$\text{recall}\left(f(B_{pi}), f(B_{pj})\right) = \frac{\sum \text{len}(b_x) + \sum \text{len}(b_y)}{N} \quad (3.27)$$

其中(3.26)與(3.27)中的 b_x 代表 Family $f(B_{pi})$中任一段落，b_y 代表 Family $f(B_{pj})$中任一段落。(3.27)中的 N 代表歌詞總行數。

在此 precision 的意義為希望找出來的 Family 組合結果的內聚力依長度 Family 段落總長度比例的總和越高越好。可是長度較短的 Family 出現內聚力高的可能性較大，因此 recall 在此是希望此 Family 組合對於歌詞總行數的覆蓋率越高越好，用以平衡兩者之間的關係。

圖 3.24 為一個組合分數計算的例子。輸入的歌詞總共 10 行，上方的編號為行號。段落樣式 B_{p1}=(1, 3)、B_{p2}=(4,5)。Family $f(B_{p1})$ = {(1, 3), (7, 9)}(藍色區塊)、$f(B_{p2})$ = {(4, 5), (10, 11)}(橘色區塊)。假設 Co($f(B_{p1})$) = 0.7、Co($f(B_{p2})$) = 0.6，則 precision = 0.7 *6 / 10 + 0.6 * 4 / 10 = 0.66，recall = 10 / 11 = 0.91。最後組合分數 CS($f(B_{p1})$, $f(B_{p2})$) = 2 * 0.66 * 0.91 / (0.66 + 0.91) = 0.76。

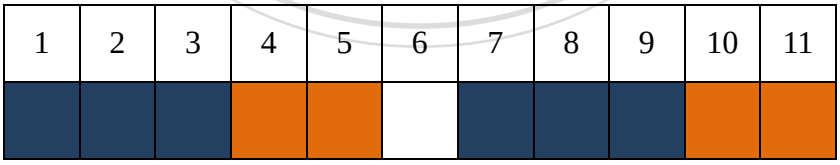


圖 3.24 可能性分數計算例子

我們將計算所有 Family 組合的可能性，取最大組合分數的 Family 組合做為最佳的組合。我們利用 Family 內聚力矩陣 F 可以減少部分的組合計算量。例如 F(4, 3)代表長度為 4，起始行為 3 的 $f(B_{pi})$，可以推斷在 F 矩陣上第 3 到 6 欄的 $f(B_{pj})$

的段落樣式必定會與 $f(B_{pi})$的段落樣式有段落重疊，因此在第 3 到 6 欄範圍裡出現的 Family 可以不必計算 Family 之間的段落重疊檢查，用以減少時間計算量。圖 3.25 為 Family 最佳組合搜尋演算法。

```
Algorithm Optimal Family Combination Search(OFCS)
Input: Family Cohesion Matrix (F)
Output: Optimal Family Combination (OFC)

bestCombination = <∅>
bestScore = 0

Procedure OFCS( F):
   combineList = <∅>
   Combination(F, 1, 0, combineList)
   return bestCombination

Procedure Combination(F, length, start, combineList):
   if   combineList.length == 2: /* 確定已經組合兩種 Family */
      score = CS(combineList)   /*計算此 Family 組合的分數*/
      if score > bestScore:
         bestScore = score
         bestCombination = combineList

   maxStart = N
   if combineList.length == 0:
      maxStart = ⌈N/2⌉

   nextStart = start + length
   for nextLength = 2 to ⌈N/2⌉:     /*N 為歌詞總行數*/
      for nextStart = nextStart to maxStart:

         isOverlap = overlap(combineList, f(F(nextLength, nextStart) ))/*檢查重疊*/
         if !isOverlap:
            combineList.append(f(F(nextLength, nextStart))
```

圖 3.25 Family 最佳組合搜尋演算法

3.5. 詞式段落標記

3.6 節已經找出最佳的 Family 組合，接下來我們希望電腦自動標記歌詞的段落標籤，我們分成五種標籤，分別為主歌、副歌、前段、橋段與尾聲。我們整理了簡單的規則來讓電腦標記。在 3.6 節我們可以得到一個最佳的 Family 組合，$f(B_{pi})$與 $f(B_{pj})$，我們認為此兩個 Family 必定為主歌與副歌段落，接下來利用三個規則對兩個 Family 投票：

1. 若 $f(B_{pi})$中的段落個數(包含樣式段落) > $f(B_{pj})$中的段落個數，則 $f(B_{pi})$票數加一，若一樣則 $f(B_{pi})$與 $f(B_{pj})$各加一票。
2. 若 $Co(f(B_{pi})) > Co(f(B_{pj}))$ 則 $f(B_{pi})$ 票數加一。
   若 $Co(f(B_{pi})) = Co(f(B_{pi}))$則 $f(B_{pi})$與 $f(B_{pj})$各加一票。
3. 若 $f(B_{pi})$中 B_{pi}起始行 s_{pi} > $f(B_{pj})$中 B_{pj}起始行 s_{pj}，則 $f(B_{pi})$ 票數加一。

最後取票數較高的 Family 標記為副歌，其次標記為主歌。標記好了主歌與副歌，再看剩餘沒有被 $f(B_{pi})$與 $f(B_{pj})$覆蓋的段落。剩下三個標籤前段、尾聲與橋段分別的標記規則為：

- 前段：若剩餘的段落中起始行號 s = 1，則標記為前段。
- 尾聲：若剩餘的段落中結束行號 e = N(歌詞總行數)，則標記為尾聲。
- 橋段：若剩餘的段落不屬於前段或尾聲，則為橋段。

圖 3.26 為一個自動標記的例子。輸入的歌詞總共 15 行，上面的編號為行號。段落樣式 B_{p1}=(3, 4)、B_{p2}=(5, 7)。Family $f(B_{p1})$ = {(3, 4), (10, 11)}(藍色區塊)、$f(B_{p2})$ = {(5, 7), (12, 14)}(橘色區塊)。假設 $Co(f(B_{p1})) = 0.8$、$Co(f(B_{p2})) = 1.0$，則在分主副歌的規則 1 就會將 $f(B_{p1})$標記為主歌，$f(B_{p2})$標記為副歌。剩餘的段落(1, 2)、(8, 9)與(15, 15)。段落(1, 2)因為起始行號為 1 所以標記為前段。段落(15, 15)因為結

束行號為 15 等於歌詞總行數，因此標記為尾聲。最後剩下段落(8, 9)不屬於前段或尾聲，因此標記為橋段。

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

圖 3.26 詞式段落標記例子

國立政治大學 National Chengchi University

第 4 章

詞曲結構搭配

在此章我們希望自動搭配歌詞與只有人聲的主旋律，本研究主要從結構的角度出發，在搭配的過程我們分成兩個層次，第一層為詞句/樂句結構粗略的對應，第二層為漢字/音符細部的對應，其中有考慮到詞式/曲式結構。接下來的小節分別介紹各層的對應方法。

4.1. 詞句與樂句結構對應

我們在說話的時候常常會說一句話停頓一下，並且停頓點不會斷在有意義詞彙中，例如「聽音樂」這句話，我們可以說「聽」、「音樂」而不會用「聽音」、「樂」這樣的節奏來說話，因為會無法傳達原本要表達的意義。而在音樂也有所為的樂句(Phrase)，樂句是由數個音符構成，長度不定，通常為歌手或是樂器手在一次呼吸內可以唱完或演奏完的長度[41]。黃志華[36]與陳富容[40]在他們的著作中有提到詞曲搭配的一些原則，其中都有詞句與樂句需互相配合，如此就是要避免歌詞表達不正確的情況，例如圖 4.1 為芳艷芬《檳城艷》中兩句歌詞分別是「你看看那邊」與「艷侶雙雙花蔭下」。中間黑粗線為樂句斷點，第一個樂句為四個音符，第二個樂句為九個音符。可以看到詞彙「那邊」剛好斷在樂句斷點，原因是第一個樂句只有四個音符，可是歌詞第一句卻有五個字，才會造成最後一個字不得不合併到下個樂句，如此便是一個詞句與樂句搭配不好的例子。

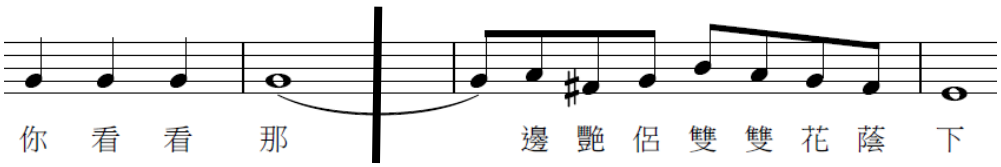


圖 4.1 芳艷芬《檳城艷》歌詞

樂句並沒有一個統合性的定義，在本研究中我們將歌詞中的一句話對應的音符定義為一個樂句。現在給定一首歌詞的詞句字數序列 S 與一首歌曲主旋律的樂句音符數序列 P：

$$S = s_1, s_2, \ldots, s_I, \text{ where } s_i \in N^+, 1 \le i \le I$$

$$P = p_1, p_2, \ldots, p_J, \text{ where } p_j \in N^+, 1 \le j \le J$$

其中 s_i 為第 i 個詞句的字數，p_j 為第 j 個樂句的音符數，並且 S 與 P 滿足 $\sum_{i=1}^{I} s_i * sing_{limit} \le \sum_{j=1}^{J} p_j$，其中 $sing_{limit}$ 代表一字可以唱的音符數。給定一個對應 m，使得 s_i 對應 $p_j, \ldots, p_{j+t} p_i$ 或 p_j 對應 $s_i, \ldots, s_{i+t}$，對應 m 表示為：

$$m = <(i, j), \ldots, (i, j+t)> \text{ 或 } m = <(i, j), \ldots, (i+t, j)>, \text{ where } t \in N$$

m(x)表示為此對應的第 x 個元素，則一個對應 m 必須滿足兩個限制：(1)連續性：$\forall x \in \{1, \ldots, t\}, m(x) - m(x-1) \in \{(0,1), (1,0)\}$；(2)配唱性：$p_j \le s_i \le sing_{limit} * p_j$ 或 $\lceil p_j / N \rceil \le p_j \le s_i$，其中 $sing_{limit}$ 代表一字可以唱的音符數。第二項配唱性是因為唱歌的時候，一個音符只能對應一個字，而一個字可以唱 N 個音符。最後我們將 S 與 P 的對應關係用一條對應序列 M 表示：

$$M = m_1, m_2, \ldots, m_q, \ldots, m_K, \text{ where } K \in N$$

圖 4.2 為給定 S 與 P 對應序列 M 二維圖式表示圖，S 為縱軸，P 為橫軸，對應序列 $M=m_1,m_2,m_3$，其中 $m_1=<(1, 1), (1, 2), (1, 3)>$，$m_2=<(2, 4)>$，$m_3=<(3, 5), (4, 5)>$。

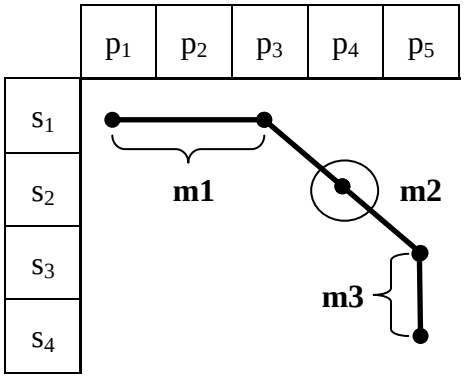


圖 4.2 對應序列二維表示

一個對應序列 M 必須要符合三個限制，分別為(1)單調性：$m_{q-1}(\text{last element}) < m_q(1)$；(2)連續性：$m_q(1) - m_{q-1}(\text{last element}) \in \{(\mathbf{1}, \mathbf{1})\}$；(3)邊界性：$m_1(1) = (1,1)$ 且 $m_K(\text{last element}) = (I, J)$。

現在定義一個成本函數 c_{total}，此函數是用來計算一個對應 m 的成本分數，c_{total} 如(4.1)：

$$c_{total}(m) = \alpha * c_{nwr}(s_{idx}, p_{idx}) + (1-\alpha) * c_{merge}(s_{idx}, p_{idx}) \quad (4.1)$$

其中 c_{nwr} 代表音字比(Note Word Ratio)成本，c_{merge} 代表合併成本，α代表權重，s_{idx} 代表 m 中所有元組(tuple)中的 i 值集合，$s_{idx} = \{i \mid \forall x: m(x)\text{中的 i 值}\}$。$p_{idx}$ 代表 m 中所有元組中的 j 值集合，$p_{idx} = \{j \mid \forall x: m(x)\text{中的 j 值}\}$。$c_{wnr}$ 的定義如(4.2)，c_{merge} 的定義如(4.3)：

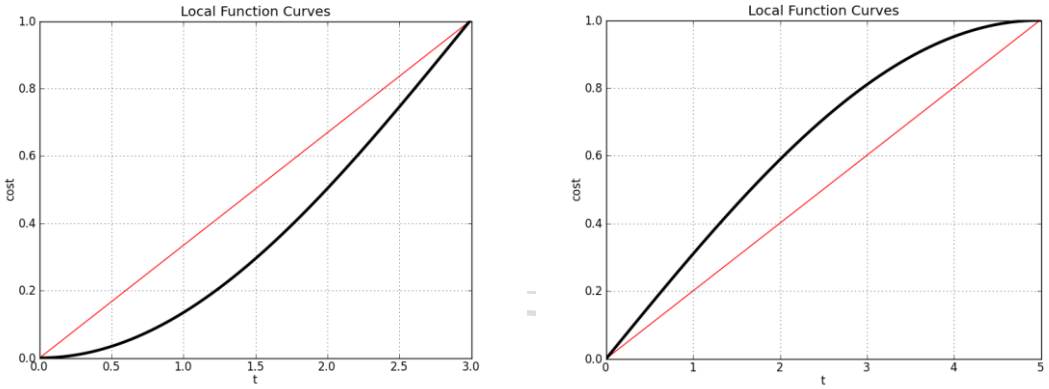
令 $\frac{\sum_{j \in p_{idx}} p_j}{\sum_{i \in s_{idx}} s_i}$ 為 NWR，則 $c_{wnr}(s_{idx}, p_{idx}) = \sin\left(\frac{2\pi * NWR}{4 * sing_{limit}} + \frac{3\pi}{2}\right) + 1$ (4.2)

令 $|p_{idx}| + |s_{idx}| - 2$ 為 MC，

則 $c_{merge}(s_{idx}, p_{idx}) = \begin{cases} \sin\left(\frac{2\pi * MC}{4 * merge_{limit}}\right), \text{if } MC > merge_{limit} \\ 1, \text{otherwise} \end{cases}$ (4.3)

其中 MC 表示合併次數(Merge Count)，代表對應 m 中，可能是詞句合併的次數或是樂句合併的次數。$sing_{limit}$ 為一個參數，代表一字最多唱幾音。$merge_{limit}$ 為一個參數，代表對應 m 中無論樂句或詞句的合併次數上限。

圖 4.3 為成本函數曲線圖，圖 4.3(a)為 c_{nwr} 對於 t 的曲線圖，$sing_{limit}$=3，相較於線性呈現一個下凹的曲線，圖 4.3(b) 為 c_{merge} 對於 t 的曲線圖，$merge_{limit}$=5，相較於線性呈現一個上凸的曲線。



(a) (b)

圖 4.3 成本函數曲線圖

(a) c_{nwr} 曲線圖，$sing_{limit}$=3 (b) c_{merge} 曲線圖，$merge_{limit}$=5

現在有了對應 m 的成本函數 c_{total}，則我們希望求 S 與 P 之間找到一條長度為 K，總成本最小的對應序列 M， C(S, P)如(4.4)：

$$C(S,P) = \frac{\underset{M}{\text{Min}}\left(\sum_{q=1}^{K} c(m_q)\right)}{\min(|S|,|P|)} \tag{4.4}$$

其中|S|與|P|為序列 S 與 P 各自的長度。為了求 C(S, P)的解，我們利用動態規畫的技巧，找到其一般關係的遞迴關係式為(4.5)：

$$D(i,j) = \min \begin{cases} D(i-1,a) + c_{total}(<(i,a),\dots,(i,j)>) \\ \dots \\ D(i-1,b) + c_{total}(<(i,b),\dots,(i,j)>) \end{cases} \begin{cases} D(c,j-1) + c_{total}(<(c,j),\dots,(i,j)>) \\ \dots \\ D(d,j-1) + c_{total}(<(d,j),\dots,(i,j)>) \end{cases} \tag{4.5}$$

D(i, j)的初始關係為(4.6)、(4.7)與(4.8)：

$$D(0,0) = 0 \tag{4.6}$$

$$D(0,j) = \infty, \forall j \in \{1,2,\dots,J\} \tag{4.7}$$

$$D(i,0) = \infty, \forall i \in \{1,2,\dots I\} \tag{4.8}$$

D(i, j)代表 S 的第 i 個元素的 prefix 與 P 的第 j 個元素的 prefix 的最小對應成本。因此我們可將目標(4.4)改寫成(4.9)，便可求解。

$$C(S,P) = \frac{D(I,J)}{\min(|S|,|P|)} \tag{4.9}$$

在(4.5)中。每一個狀態 D(i, j)分成兩個部分，分別是上方方框的單一詞句元素對應多個樂句元素的遞迴式，總共有(b – a +1)種可能。下方方框的單一樂句元素對應多個詞句元素的遞迴式，總共有(d – c + 1)種可能，因此在每一個狀態 D(i, j)總共需要考慮(b-a+d-c+2)種選擇來取得最佳解。a, b, c, d 的關係為，j ≥ b ≥ a，i ≥ d ≥ c，。我們可以依據一個對應 m 中的**配唱性**限制求得(4.5)中的 a, b, c, d，a, b, c, d 必須滿足下列各自的不等式(4.10)~(4.13)：

$$\text{a 滿足 } \max \sum_{t=j}^{a} p_t \le s_i * sing_{limit} \tag{4.10}$$

$$\text{b 滿足 } \min \sum_{t=j}^{b} p_t \ge s_i \tag{4.11}$$

$$\text{c 滿足 } \max \sum_{t=i}^{c} s_t \le p_j \tag{4.12}$$

$$\text{d 滿足 } \min \sum_{t=j}^{d} s_t \ge \left\lceil \frac{p_j}{sing_{limit}} \right\rceil \tag{4.13}$$

最後利用回溯的方式我們可以得到 S 與 P 最佳的對應序列 $M=m_1,m_2,\dots m_K$。

4.2. 漢字與音符對應

有了最佳的對應序列 $M=m_1,m_2,\ldots m_K$，接下來我們將 m_q 中樂句與詞句的對應關係做漢字與音符的第二層對應，$1 \leq q \leq K$。 我們將樂句中的音符與詞句中的漢字都轉換成一個三維的特徵向量=(音高, 音長, 詞式/曲式標籤)。

由於我們的音樂主旋律是採用 MIDI 格式，因此很容易可以得到音符的音高，音長，並且利用人工標記的得到主旋律的曲式。而漢字的音高我們一樣利用中文聲調語言的特性，先求得主旋律的平均音高 $Pitch_{mean}$，接下來每種聲調與音高的對應關係如表 4.1，其中 128 為 MIDI 中音高的最大值。表 4.1 對應關係依然有符合表 3.1 聲調與音高走勢搭配的規則。

表 4.1 聲調與音高對照表

第一聲	第二聲	第三聲	第四聲	輕聲
$\frac{Pitch_{mean}+8}{128}$	$\frac{Pitch_{mean}}{128}$	$\frac{Pitch_{mean}-1}{128}$	$\frac{Pitch_{mean}+7}{128}$	同前一個漢字

漢字的音長，我們參考 Wong 等人[30]的想法，他們發現一句歌詞的最後一個字與下一句歌詞的第一個字之間的時間間隔，通常都比同一句話內的每個漢字的時間間隔來的長。因此我們根據這樣的想法，先求得主旋律的平均音長 $Duration_{mean}$，接下來將詞句中的每個漢字依照其位置產生漢字音長，如表 4.2，其中 8 為 MIDI 中音長的最大值。

表 4.2 漢字與音長對照表

其他位置	最後一個字
$\frac{Duratoin_{mean}}{8}$	$\frac{Duratoin_{mean}+2}{8}$

現在經由對應 m_q 產生漢字特徵序列 $W=w_1,w_2,\ldots,w_I$ 與音符特徵序列 $N=n_1,n_2,\ldots,n_J$，我們利用 DTW 演算法尋找 W 與 N 的最小對應距離。其中兩元素之間的距離 $d(w_i, n_j)$利用歐氏距離定義為(4.14)：

$$d(w_i, n_j) = \frac{\|w_i - n_j\|}{\sqrt{3}} \qquad (4.14)$$

為了符合唱歌時一字唱多音，一音只能對一字的限制，我們設計一個新 Step Type 如圖 4.4，這樣的 Step Type 代表一字只能唱三個音符。我們觀察大部分的歌曲一個字通常不會唱超過三個音符，否則會造成演唱者演唱上的困難。w_1,w_2 與 w_3 分別為一音唱一音、二音與三音的權重，唱越多音權重越大。

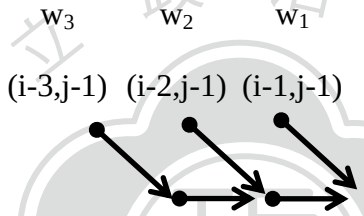


圖 4.4 歌唱限制的 Step Type

接著利用 DTW 求得一個對應 m_q 產生的 W 與 N 的對應距離 Dist(W, N)如(4.15)，其中 D(I,J)為求取最小的 DTW 對應距離，最後除以 W 與 N 的序列長度加總做正規化。

$$\mathrm{Dist}(m_q) = \mathrm{Dist}(W, N) = \frac{D(I, J)}{|W| + |N|} \qquad (4.15)$$

最後利用回溯的方式我們可以得到 W 與 N 最佳的音符特徵與漢字特徵對應序列。有了每個對應 m_q 的距離，詞曲搭配的總分數為(4.16)：

$$C(S, P) + \frac{\sum_{q=1}^{K} \mathrm{Dist}(m_q)}{K} \qquad (4.16)$$

第 5 章

實驗結果評估

本研究實作環境在 CentOS 作業系統上，使用 Python 2.7.2 程式語言實作，硬體設備 CPU 為 Intel(R) Core™ i3 540 3.07Ghz、記憶體 2.0GB。

歌詞來源取自 KKBOX 音樂網站，歌詞斷詞利用中央研究院中文斷詞系統，其系統可以處理出特別的名詞，很適合利用在歌詞的文體上，例如陳昇《布考斯基協奏曲》中的一句歌詞為「布考斯基的蒼蠅酒吧」，經由中央研究院中文斷詞系統得到的結果為「是」、「布考斯基」、「的」、「蒼蠅」、「酒吧」，其中「布考斯基」為一個外國名字。歌詞中每個漢字的注音，取自線上英漢字典(http://cdict.net/)，我們將歌詞中的每一句丟入線上英漢字典，並抓取回傳的注音，如此是為了解決中文破音字的問題，像是「行」字可以念「ㄏㄤˊ」或「ㄒㄧㄥˊ」，可是若用「銀行」或「行走」等詞彙，則就可以確定注音符號的讀音。

詞式分析的實驗，我們會先介紹詞式標記方法與評估方法，最後為詞式的參數設定與評估結果。接著為詞曲結構搭配實驗，我們會先講述評估方法以及評估結果。

5.1. 詞式標記方法

網路上有些歌友會將歌詞分好段落，可是會有相同歌詞不同分段方式的情況出現，這表示每個人對於分段的意義有不同的理解。我們採用聽音樂來標記歌詞詞式的方式，若有重複的旋律則為相同的段落標記，否則為不同的段落標記。

例如圖 5.1 為詞式標記的兩個例子。圖 5.1(a)為張宇《用心良苦》的前八行歌詞，段落(1,4)與段落(5,8)由於音樂旋律是相同的，並且內容是在述說故事的背景，因此段落(1,4)與段落(5,8)標記同為主歌。圖 5.1(b)為周華健《朋友》的前八

行歌詞，段落(1,4)與段落(5,8)由於音樂旋律不同，並且段落(5,8)對應的音樂比段落(1,4)對應的音樂來得豐富，形成一個很大的對比，因此我們將段落(1,4)標記為主歌，段落(5,8)標記為副歌。

1.	妳讓他用戒指把妳套上的時候	1.	這些年 一個人 風也過 雨也走
2.	我察覺到妳臉上複雜的笑容	2.	有過淚 有過錯 還記得堅持什麼
3.	那原本該是我 付予妳的承諾	3.	真愛過 才會懂 會寂寞 會回首
4.	現在我只能隱身熱鬧中	4.	終有夢 終有你 在心中
5.	我跟著所有人向妳祝賀的時候	5.	朋友 一生一起走 那些日子 不再有
6.	只有妳知道我多喝了幾杯酒	6.	一句話 一輩子 一生情 一杯酒
7.	我不能再看妳 多一眼都是痛	7.	朋友 不曾孤單過 一聲朋友 你會懂
8.	即使知道暗地裡妳又回頭	8.	還有傷 還有痛 還要走 還有我
	(a)		(b)

圖 5.1 詞式標記方法

5.2. 詞式分析評估方法

MIREX(The Music Information Retrieval Evaluation eXchange)為音樂資訊檢索演算法的競賽，這個競賽是每年辦隨 ISMIR(International Society for Music Information Retrieval)會議一同舉辦的活動。MIREX 當中有很多關於音樂研究議題的競賽項目，像是音樂類型分類(Audio Genre Classification)、音樂調性偵測(Audio Key Detection)或主旋律擷取(Audio Melody Extraction)等等的項目，其中我們參考音樂結構分析(Structural Segmentation)項目中三個評測的面向，分別為：(1)Boundary Retrieval；(2)Line Clustering；(3)Over and Under Segmentation。最後我們多參考了[22]中的 Label Recover Rate 方法。因此我們總共有四個評估的面向，以下分別一一詳述。

5.2.1. Boundary Retrieval (邊界檢索)

Boundary Retrieval 目的在於評估詞式分段點切割的情況，其評估方法是採用 f-score 中的 precision 與 recall。現在給定兩個分段集合分別為系統預測 E_B 與標準答案 T_B，則 BP(Boundary Precision)定義為(5.1)，BR(Boundary Recall)定義為(5.2)：

$$BP = \frac{|E_B \cap T_B|}{|E_B|} \quad (5.1)$$

$$BR = \frac{|E_B \cap T_B|}{|T_B|} \quad (5.2)$$

BF(Boundary f score)為(5.3)：

$$BF = \frac{2 * BP * BR}{BP + BR} \quad (5.3)$$

BP 的意義為系統預測的斷點中有多少是標準答案的斷點。BR 的意義為標準答案的斷點中有多少是系統預測的斷點。圖 5.2 為一個 Boundary Retrieval 計算例子，上方為系統預測的斷點，總共 3 個。下方為標準答案的斷點，總共 2 個。這個例子中 BP = 1 / 3 = 0.66、BR = 1 / 2 = 0.5、BF=0.57。

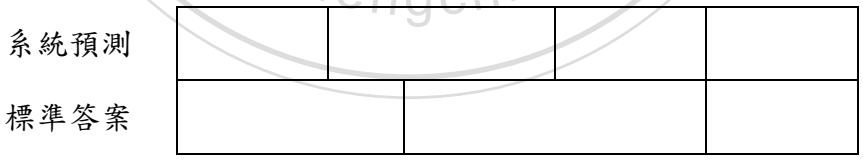


圖 5.2 Boundary Retrieval 計算

5.2.2. Line Clustering (詞行分群)

Line Clustering 目的在於評估詞式段落分群與分群段落坐落行號的正確性，評估的方法為 Levy 等人[12]提出的 Pairwise f-score。這個方法主要是評估同群之間的關係，而不是在乎群的類別，例如系統預測的段落分群結果 ABCBC，而標準答案為 ACBCB，則應該視為一樣。

現在給定一個系統預測的分群結果 E，首先要將 E 轉換成一集合的配對 pair=(i, j)，其意義為歌詞中的第 i 行與第 j 行屬於同一群，E_C={$pair_1$, $pair_2$,…, $pair_3$}。標準答案的分群 T 也經由相同的轉換，成為 T_C。圖 5.3 為一個分群結果轉換的例子 ，上方數字為行號，下方 A 與 B 為分群代號，則產生的配對集合為{(1, 2), (1, 6), (2, 6), (3, 4), (3, 5), (4, 5)}。

1	2	3	4	5	6
A		B			A

圖 5.3 分群結果轉換

若有了系統預測配對集合 E_C 與標準答案配對集合 T_C，precision 與 recall 分別為(5.4)與(5.5)：

$$\text{precision} = \frac{|E_C \cap T_C|}{|E_C|} \qquad (5.4)$$

$$\text{recall} = \frac{|E_C \cap T_C|}{|T_C|} \qquad (5.5)$$

PF(Pairwise f-score)為：

$$\text{PF} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \qquad (5.6)$$

在 Line Clustering 中的 precision 代表系統分群配對的精確度，recall 代表標準答案配對的召回率，與 Boundary Retrieval 不一樣的地方在於這邊是考慮分群是否正確。

5.2.3. Over and Under Segmentation (高估與低估分段)

Over and Under Clustering 目的評估系統預測的分群結果是包含的資訊是否有正確傳達標準答案的資訊。Lukashevich[13]參考自資訊理論(Information Theory)中 Conditional Entropy 進而提出 Normalized Conditional Entropy 評估方法。

兩個資料集 d_1 與 d_2，Conditional Entropy $H(d_2 \mid d_1)$代表以資料集 d_1 為基礎，編碼(encode)資料集 d_2 會多出的資訊量。現在給定系統預測的分群結果 E 與標準答案 T，則 $H(E \mid T)$代表系統預測的分群結果比標準答案**多餘**的資訊量，$H(T \mid E)$代表系統預測的分群結果比標準答案**遺失**的資訊量。現在給定 E 與 T，其中 E 有 N_e 群，T 有 N_t 群則 $H(E \mid T)$與 $H(T \mid E)$的計算方式如(5.7)與(5.8)：

$$H(E|T) = -\sum_{t=1}^{N_t} p(t) \sum_{e=1}^{N_e} p(e|t) \log_2 p(e|t) \tag{5.7}$$

$$H(T|E) = -\sum_{e=1}^{N_e} p(e) \sum_{t=1}^{N_t} p(t|e) \log_2 p(t|e) \tag{5.8}$$

其中 t 為標準答案 T 中的某一群， e 為系統預測分群結果 E 中的某一群。(5.7)中的 p(t)為第 t 群中所有段落長度總和占所有歌詞的比例。p(e|t)代表在第 t 群中段落出現的詞行範圍下，出現類別 y 的比值。(5.8)則以此類推。

隨著分群數目不同，Conditional Entropy 分數的最大值也會跟者變化，如此會造成兩首不同歌詞無法正確比較效果差距，因此 Lukashevich 提出正規化的方法稱為 Normalized Conditional Entropy。(5.7)中其最大的值出現的情況為當 p(e|t)呈現均勻分布(Uniform Distribution)的時候，$p(e|t)=1 / N_e$。帶回(5.7)則 H(E|T)的最大值 $H(E|T)_{max} = \log_2 N_e$，其計算過程如下：

$$H(E|T)_{max} = -\sum_{t=1}^{N_t} p(t) \sum_{e=1}^{N_e} \frac{1}{N_e} \log_2 \frac{1}{N_e}$$

$$= -\sum_{t=1}^{N_t} p(t) \log_2 \frac{1}{N_e}$$

$$= -1 * \log_2 \frac{1}{N_e} = \log_2 N_e$$

(5.8)也依此類推得到 H(E|T)的最大值 $H(E|T)_{max} = \log_2 N_t$。

有了 Conditional Entropy 與其最大值，最後 Over(Over Segmentation)與 Under(Under Segmentation)的定義分別為(5.9)與(5.10)：

$$\text{Over} = 1 - \frac{H(E|T)}{\log_2 N_e} \qquad (5.9)$$

$$\text{Under} = 1 - \frac{H(T|E)}{\log_2 N_t} \qquad (5.10)$$

Over 分數越接近 0 代表系統預測分群結果越有多餘的資訊。Under 分數越接近 0 代表系統預測分群結果越有遺失的資訊。圖 5.4 為一個 Over 與 Under Segmentation 說明例子，圖上方的數字為行號，下方的英文為分群代號。可以看出系統預測的 A 群對應到標準答案的 A 與 B 群，若放寬來說系統預測的分群結果其實與標準答案是一樣的，只不過標準答案切割的比較細。因此其 Over=1，Under=0.58，呈現 Over 高於 Under 的情況，如此有反應出不同階層分群的特性。

	1	2	3	4	5	6
系統預測	A		B		A	
標準答案	A	B	C		A	B

圖 5.4 Over 與 Under Segmentation 說明

5.2.4. Label Recovering Ratio (標籤回復率)

Label Recover Rate 的目的在於測試自動標記段落標籤的效果。現在給定系統預測的段落標記結果 E 與標準答案 T，LRR(Label Recover Ratio)定義為 E 與 T 標籤的交集占總歌詞行數的比例。圖 5.5 為一個 LRR 計算的例子，圖上方的數字為行號，下方的主、副為段落標籤。則加總每一行相同標籤對於總行數所形成的比值，LRR= 5 / 6 = 0.83。

	1	2	3	4	5	6
系統預測	主		副		主	
標準答案	主		副	主		

圖 5.5 Label Recover Rate 計算

5.3. 詞式分析實驗結果

首先我們列出整個詞式分析演算法各個階段的參數設定，分成三個階段：(1)SSM 建立參數設定；(2)Instance Path Search 參數設定；(3)有效實體段落門檻值設定。

5.3.1. SSM 建立參數設定

句結構、拼音與詞性三種特徵序列利用詞行結構排比演算法計算兩行特徵序列之間的相似度。在行結構排比演算法需要設定 P_c(跨空白逞罰分數)、P_o(第一個 Gap 逞罰分數)與 P_e(第二個 Gap(含)之後的逞罰分數)。表 5.1 為三種特徵元素對應的三種懲罰分數設定。

表 5.1 三種特徵元素分別的懲罰分數設定

	P_c	P_o	P_e
句字數	6.02	2.89	1.45
拼音	-2	-1	-0.5
詞性	-2	-1	-0.5

聲調特徵序列利用 DTW 來計算兩行特徵序列之間的相似度。其中 DTW 的 Step Type 的路徑權重，我們設定如圖 5.6，w_1=1.5，w_2=1.0，w_3=1.5。

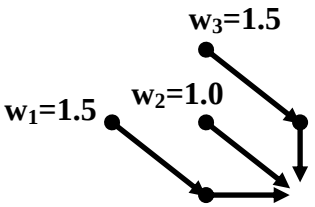


圖 5.6 DTW 路徑權重設定

線性組合 SSM 我們將四種特徵值產生的 SSM 的權重都設定為一樣，因此 $SSM_{hybrid} = 0.25 * SSM_{sen} + 0.25 * SSM_{pinyin} + 0.25 * SSM_{pos} + 0.25 * SSM_{tone}$。

5.3.2. Instance Path Search 參數設定

在 Instance Path Search 中我們要設定轉移分數 P_{diag} 與 P_{other}。我們將參數設定為 P_{diag}=0.3，P_{other}=0.0。

5.3.3. 有效實體段落門檻值設定

要成為有效段落有兩個門檻值，分別是相似度門檻值與長度門檻值。相似度門檻值為給定的 SSM，先去掉相似度為 1 的元素，將剩下的元素取平均加上一個標準差。長度門檻值為將給定的樣式段落的長度乘以 6/7。

5.3.4. 實驗結果

我們標記了 85 首歌詞的詞式，利用上述的參數設定，對四種特徵產生的 SSM 與線性組合 SSM 跑詞式分析演算法，結果如表 5.2，第一欄為，其中粗體數字代表該項評測最高的分數，BF 為 Boundary f-score，PF 為 Pairwise f-score，LRR 為 Label Recover Rate，BP 為 Boundary Precision，BR 為 Boundary Recall。可以發

現線性組合 SSM 的分數在每一項評測都是最高，這表示線性組合 SSM 有整合到各個特徵部分的優點。

表 5.2 詞式分析結果

	BF	PF	LRR	BP	BR	Over	Under
線性組合	**0.67**	**0.78**	**0.73**	**0.66**	**0.69**	**0.73**	**0.79**
句字數	0.63	0.74	0.68	0.61	0.69	0.70	0.75
拼音	0.65	0.74	0.66	0.64	0.67	0.68	0.77
聲調	0.61	0.72	0.69	0.60	0.63	0.67	0.74
詞性	0.63	0.73	0.67	0.62	0.66	0.67	0.77

Pairwise f-score 不論在哪一種特徵產生的 SSM 都比 Boundary f-score 高，這表示系統預測的詞式結果，段落切割的落點雖然有瑕疵，可是段落分群不太受影響。例如圖 5.7 為任賢齊《不要變》的線性組合 SSM 分析結果，E 為系統預測，T 為標準答案。可以看到系統預測把主歌部分合併了，因此系統預測少了兩個斷點，可是分群結果都是正確的，造成 BF=1.0，而 BP=1.0、BR=0.67。而圖 5.8 為蕭亞軒《甩啦甩啦》的線性組合 SSM 分析結果，相較於圖 5.7 是一個反過來的例子。系統預測把副歌部分多切割了，即使多切割了段落還有將其段落分成相同的類別，造成 PF=1.0，而 BP=0.71、BR=1.0。

E	主		副	主		副	副
T	主	主	副	主	主	副	副

圖 5.7 任賢齊《不要變》

E	主	副	副	主	副	副	副	尾
T	主	副		主	副		副	尾

圖 5.8 蕭亞軒《甩啦甩啦》

Boundary Recall 不論在哪一種特徵產生的 SSM 都比 Boundary Precision 高，這代表我們演算法分析結果切的斷點比較多，在斷點比較多的情況下，就會有較大的機率產生 Over Segmentation 的情況，因此 Over 分數會比 Under 分數來的低。例如圖 5.9 為孫燕姿《我要的幸福》的線性組合 SSM 分析結果，E 為系統預測，T 為標準答案。系統預測有 7 個斷點，標準答案只有 6 個，系統預測將標準答案中的主歌分成切割成兩塊，並且這兩塊分別是屬於前段與主歌，如此造成 BP=0.86、BR=1.0、Over=0.89、Under = 1.0 的結果。

E	前	主	副	主	副	橋	副	尾
T	主		副	主	副	橋	副	尾

圖 5.9 孫燕姿《我要的幸福》

四種特徵值的 SSM 與線性組合 SSM 在 77 首歌詞當中都有分析不出詞式的歌詞，分不出的歌詞數分別為，數線性組合 SSM 5 首，句字數 SSM 5 首，拼音 SSM 5 首，詞性 SSM 4 首，聲調 SSM 7 首。彼此的交集為 3 首，聯集為 9 首。若將各自分析不出詞式的歌詞不列入分數計算，則結果如表 5.3，可以發現整體的分數都有提升，比沒去除無法分析的歌詞大約提升 5%左右。

表 5.3 去除分析不出詞式歌詞的實驗結果

	BF	**PF**	**LRR**	**BP**	**BR**	**Over**	**Under**
線性組合	**0.71**	**0.83**	**0.78**	**0.70**	0.73	**0.77**	**0.84**
句字數	0.69	0.80	0.74	0.66	**0.75**	0.76	0.82
拼音	0.69	0.78	0.70	0.68	0.71	0.72	0.82
聲調	0.66	0.79	0.75	0.65	0.68	0.73	0.80
詞性	0.66	0.76	0.70	0.65	0.70	0.71	0.81

表 5.4 為無法分析的歌詞與特徵值 SSM 對應表，無法分析代表在此特徵 SSM 上找不到任兩種 Family 組合的結果，其中打叉為此特徵 SSM 無法分析，數字代表此特徵 SSM 可以分析，其值為分析結果的 Pairwise f-score。無法分析的原因，我們歸納出三個原因：(1)網路上歌詞斷句不正確，導致歌詞行數過短；(2)歌詞內容以人來看，看不出有重複樣式；(3)特徵值本身的缺陷。

表 5.4 無法分析的歌詞與特徵值 SSM 對應表

歌曲名稱	線性組合	句字數	拼音	聲調	詞性
堅強的理由	×	×	×	×	×
長流不息	×	×	×	×	×
告別校園時	×	×	×	×	×
合久必婚	×	0.40	×	×	×
蝸牛	×	0.58	×	×	0.41
被動	1.0	×	1.0	1.0	1.0
全日愛	1.0	×	1.0	1.0	1.0
突然好想你	0.79	×	1.0	0.5	0.23
紅豆	0.74	×	0.93	0.93	0.93
千千萬萬個我	0.66	0.72	0.43	×	0.44
真的	0.62	0.45	0.45	×	0.57

《堅強的理由》、《長流不息》與《告別校園時》的歌詞分別行數為八行、九行與十行。《堅強的理由》是因為歌友將太多句並成一行，導致副歌段落為一行，造成找不出至少長度為二的 Famliy，造成沒有 Family 組合結果。《長流不息》與《告別校園時》則是內容短，並且內容上重複樣式很少，因此造成沒有 Family 組合結果。《合久必婚》歌詞的內容寫的像是新詩，因此內容上沒有重複，可是究句子數結構則有些許的重複。《蝸牛》的歌詞只有兩個特徵可以找出詞式，可

是 Pairwise f-score 都不高，原因是歌友標記的斷句方式造成演算法不好分析。《被動》、《全日愛》、《突然好想你》與《紅豆》只有句字數 SSM 無法分析，這是由於如果主歌與副歌的句結構結構很像的話，找出的Family段落範圍容易有重疊，造成無法組合句字數結構，其中《被動》與《全日愛》其他特徵的 Pairwise f-score 都為 1.0。《紅豆》不論拼音、聲調與詞性 SSM 分析的 Pairwise f-score 皆為 0.93，反而線性組合 SSM 降為 0.74，這可能是句字數特徵在這首歌來說。《千千萬萬個我》與《真的》只有聲調 SSM 無法分析，這兩首即使相同為主歌或副歌的內容歧異度太大，造成其他特徵分析的詞式結果 Pairwise f-score 分數也不高。

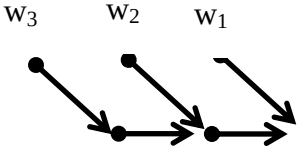
5.4. 詞曲搭配實驗

主旋律的樂句分段我們是採用人工標記的方式，標記的時候必須要聽 MIDI 旋律並且搭配原詞的詞句標記相當費時，因此我們只標記了 8 首華語流行音樂的 MIDI 主旋律樂句分段，並且標記上曲式。接著我們對這 8 首曲子推薦歌詞，歌詞資料庫總共有 85 首歌詞，其中有包含被推薦的 8 首 MIDI 原本搭配的歌詞，因此我們預期原詞推薦的名次要越前面越好。

5.4.1. 演算法參數設定

詞句與樂句結構搭配中，一個對應 m 的 c_{total} 中的α 設定為 0.6。c_{nwr} 中的 $sing_{limit}$ 設定為 3，也就是一字最多唱三個音符。c_{merge} 中的 $merge_{limit}$ 設定為 5，也就是不論詞句或樂句合併次數到 5 次以上，c_{merge} 的成本為 1。

漢字與音符中的 Step Type 權重，w1=0.18, w2=0.28, w3=0.54，我們是根據 c_{nwr} 中成本曲線，t 分別為 1, 2, 3 時對應的成本分數。



5.4.2. 實驗結果

表 5.5 為八首歌曲的歌詞推薦結果，可以發現所有的歌曲其原詞推薦的名次都是第一名，其中曲子《用心良苦》推薦搭配第二名的歌詞為《天與地》，這首歌曲是王菲翻唱《用心良苦》的版本，而我們的詞曲搭配結果有將其排名到第二名。

表 5.5 歌詞推薦結果

被推薦的歌曲名稱	音符數	原歌詞排名	符合配唱性的歌詞數	符合結構搭配歌詞數	計算時間(s)
用心良苦	394	1	77	46	27
孤單北半球	463	1	84	63	32
我願意	296	1	38	25	10
紅豆	311	1	46	18	8
被動	422	1	80	42	20
如果有一天	406	1	79	14	13
我的愛	320	1	48	26	13
曲終人散	427	1	81	44	23

符合配唱性的歌詞數代表在 85 首歌詞資料庫中有多少首歌詞符合配唱性，例如用心良苦的音符數為 394，那歌詞的字數就必須介在 132~394 之間。而符合結構搭配歌詞數表示在符合配唱性的歌詞中，有多少歌詞可以找到一個最佳的結構排比結果，因為結構排比時會受到詞句字數與樂句字數的分佈情況影響。《如果有一天》符合配唱性的歌詞數有 79 首，可是符合結構搭配的歌詞數卻只有 14 首，少了 65 首，這比其他歌曲的差距都來的大，這表示《如果有一天》的樂句音符數分佈相較於其他歌曲比較特別。計算時間主要是受到歌曲音符數多寡的影

響，其中《如果有一天》符合結構搭配歌詞數為 14 首，《紅豆》符合結構搭配歌詞數為 18 首，但是《紅豆》的計算時間卻比《如果有一天》少了 8 秒，這可能是因為《如果有一天》符合結構搭配歌詞數的 14 首中與《如果有一天》的對應序列比較長，造成有較多的對應需要計算漢字與音符的對應，因此拉長了計算時間。

國立政治大學 National Chengchi University

第 6 章

結論與未來研究

現在的流行音樂基本上是由音樂與歌詞互相搭配而成，我們希望透過詞曲重新組合搭配達到新曲舊詞、新詞舊曲或老歌新唱的效果，因此我們先透過詞式分析，分析歌詞的主歌、副歌等等段落，接下來再利用詞式的資訊做詞曲搭配。

詞式分析中，我們主要是尋找歌詞中的重複樣式，我們提出四個角度來尋找重複樣式，分別是句字數、拼音、詞性與聲調音高。我們用詞行結構排比演算法計算與 DTW 計算行與行之間的特徵序列相似度。最後我們可以得到四種特徵建立的 SSM，接著再用線性組合的方式得到一個線性組合 SSM。在給定的一個 SSM 上，我們對於所有的段落樣式做 Instance Path Search，如此可以找出此段落樣式所形成的 Family，接下來在任意組合兩種 Family 尋找最佳的 Family 組合。最後再將找到的最佳 Family 組合形成的分段方式自動標記段落標籤。詞式分析的 Pairwise f-score 可以達到 0.83，Labeling Recover Ratio 達到 0.78。詞曲結構搭配採用兩層式的對應，實驗結果顯式推薦的歌曲原詞皆為第一名，

四種特徵的 SSM 或許會因為不同時期的歌詞而必須要有不同的權重，例如民歌時期的歌詞與現在的歌詞形式差很多，在未來可以做進一步的探討。我們還希望將詞式分析的方法應用在曲式分析，只不過音樂所形成的 SSM 會很大，可能的段落樣式會太多，導致在 Family 組合時會花很多時間，因此需要設計一個有效率的演算法。詞曲結構搭配，在詞句與樂句對應的階段可以增加考量多對多的情況，而目前沒有考慮到中英夾雜的流行音樂，在未來也可以將英文的特性一併考量。

參考文獻

[1] F. Bronson, The Billboard Book of Number One Hits, Billboard Books, 1997.

[2] M. Cooper, and J. Foote, “Summarizing Popular Music via Structural Similarity Analysis,” Proc. of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2003.

[3] J. Foote, “Visualizing Music and Audio Using Self-Similarity,” Proc. of ACM International Conference on Multimedia, 1999.

[4] J. Foote, “Automatic Audio Segmentation Using a Measure of Audio Novelty,”. Proc. of IEEE International Conference on Multimedia and Expo, 2001.

[5] H. Fujihara, M. Goto, J. Ogata, K. Komatani, T. Ogata, and H. G. Okuno, “Automatic Synchronization between Lyrics and Music CD Recordings Based on Viterbi Alignment of Segregated Vocal Signals,” Proc. of IEEE International Symposium on Multimedia, 2006.

[6] S. Fukayama, K. Nakatsuma, S. S. Nagoya, Y. Yonebayashi, T. H. Kim, S. W. Qin, T. Nakano, T. Nishimoto, and S. Sagayama, “Orpheus: Automatic Composition System Considering Prosody of Japanese Lyrics,” Proc. of International Conference on Entertainment Computing, 2009.

[7] S. Fukayama, K. Nakatsuma, S. Sako, T. Nishimoto, and S. Sagayama, “Automatic Song Composition from the Lyrics Exploiting Prosody of Japanese Language,” Proc. of Conference on Sound and Music Computing, 2010.

[8] D. Iskandar, Y. Wang, M. Y. Kan, and H. Li, “Syllabic Level Automatic Synchronization of Music Signals and Text Lyrics,” Proc. of ACM International Conference on Multimedia, 2006.

[9] M. Y. Kan, Y. Wang, D. Iskandar, T. L. Nwe, and A. Shenoy, ”LyricAlly: Automatic Synchronization of Textual Lyrics to Acoustic Music Signals,” IEEE

Transactions on Audio, Speech and Language Processing, Vol. 16, No. 2, 2008.

[10] T. Kitahara, S. Fukayama, S. Sagayama, H. Katayose, and N. Nagata, "An Interactive Music Composition System based on Autonomous Maintenance of Musical Consistency," Proc. of Conference on Sound and Music Computing, 2011.

[11] K. Lee, and M. Cremer, "Segmentation-based Lyrics-Audio Alignment Using Dynamic Programming," Proc. of International Conference on Music Information Retrieval, 2008.

[12] M. Levy, and M. Sandler, "Structural Segmentation of Musical Audio by Constrained Clustering," IEEE Transactions on Audio, Speech, and Language Processing, Vol. 16, No. 2, 2008.

[13] H. Lukashevich, "Towards Quantitative Measures of Evaluating Song Segmentation," Proc. of International Society for Music Information Retrieval, 2008.

[14] N. C. Maddage, and K. C. Sim, "Word Level Automatic Alignment of Music and Lyrics Using Vocal Synthesis," ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 6, No. 3, 2010.

[15] M. Mauch, H. Fujihara, and M. Goto, "Lyrics-to-Audio Alignment and Phrase-level Segmentation Using Incomplete Internet-style Chord Annotations," Proc. of Conference on Sound and Music Computing, 2010.

[16] A. Mesaros, and T. Virtanen, "Automatic Alignment of Music Audio and Lyrics," Proc. of International Conference on Digital Audio Effects, 2008.

[17] M. Mueller, P. Grosche, and N. Jianq, "A Segment-Based Fitness Measure for Capturing Repetitive Structures of Music Recordings," Proc. of International Society for Music Information Retrieval, 2011.

[18] M. Mueller, and F. Kurth, “Towards Structural Analysis of Audio Recordings in the Presence of Musical Variations,” EURASIP Journal on Advances in Signal Processing, 2007.

[19] M. Mueller, and F. Kurth, “Enhancing Similarity Matrices for Music Audio Analysis,” Acoustics, Speech and Signal Processing, 2006.

[20] E. Nichols, D. Morris, S. Basu, and C. Raphael, “Relationships between Lyrics and Melody in Popular Music,” Proc. of International Society for Music Information Retrieval, 2009.

[21] H. R. G. Oliveira, F. A. Cardoso, and F. C. Pereira, “Tra-la-Lyrics: An Approach to Generate Text Based on Rhythm,” Proc. of International Joint Workshop on Computational Creativity, 2007.

[22] J. Paulus, and A. Klapuri, “Music Structure Analysis using a Probabilistic Fitness Measure and a Greedy Search Algorithm,” IEEE Transactions on Audio, Speech, and Language Processing, Vol. 17, No. 6, 2009.

[23] J. Paulus, M. Muller, and A. Klapuri, “Audio-Based Music Structure Analysis,” Proc. of International Society for Music Information Retrieval, 2010.

[24] G. Peeters, “Sequence Representation of Music Structure Using Higher-order Similarity Matrix and Maximum-likelihood Approach,” Proc. of International Society for Music Information Retrieval, 2007.

[25] S. Qin, S. Fukayama, T. Nishimoto, and S. Sagayama, “Lexical Tones Learning with Automatic Music Composition System Considering Prosody of Mandarin Chinese,” Proc. of Second Language Studies: Acquisition, Learning, Education and Technology, 2010.

[26] A. Ramakrishnan A, and S. L. Devi, “An Alternate Approach Towards Meaningful Lyric Generation in Tamil,” Proc. of NAACL HLT Second

Workshop on Computational Approaches to Linguistic Creativity, 2010.

[27] A. Ramakrishnan A, S. Kuppan, and S. L. Devi, “Automatic Generation of Tamil Lyrics for Melodies,” Proc. of Workshop on Computational Approaches to Linguistic Creativity, 2009.

[28] H. Sakoe, and S. Chiba, “Dynamic Programming Algorithm Optimization for Spoken Word Recognition,” IEEE Transactions on Acoustics, Speech, and Signal Processing, Nr. 1, p. 43-49, 1987.

[29] Y. Wang, M. Y. Kan, T. L. Nwe, A. Shenoy, and J. Yin, “LyricAlly:Automatic Synchronization of Acoustic Musical Signals and Textual Lyrics,” Proc. of ACM International Conference on Multimedia, 2004.

[30] C. H. Wong, W. M. Szeto, and K. H. Wong, “Automatic Lyrics Alignment for Cantonese Popular Music,” Multimedia Systems, Vol. 12, No. 4-5, 2007.

[31] S. Yu, J. Hong, and C. C. J. Kuo, “Similarity Matrix Processing for Music Structure Analysis,” Proc. of the 1st ACM Workshop on Audio and Music Computing Multimedia, 2006.

[32] 楊蔭瀏、孫從音、陳幼韓、何為與李殿魁，語言與音樂，丹青圖書有限公司，1986。

[33] 謝峰賜，簡易詞曲創作入門，新鳴遠出版有限公司，1993。

[34] 陳建銘，國語流行歌曲中的編曲工作，國立台灣大學音樂研究所碩士論文，2002。

[35] 徐富美與高林傳，歌詞聲調與旋律聲調相諧和的電腦檢測，世界華語文教學研討會論文集，2003。

[36] 黃志華，粵語歌詞創作談，三聯出版社，2003。

[37] 楊漢倫，粵語流行曲導論，香港特別行政區政府教育局，2009。

[38] 張嘉惠、李淑瑩、林書彥、黃嘉毅與陳志銘，以最佳化及機率分佈判斷漢字

聲符之研究，自然語言與語音處理研討會論文集(ROCLING)，2010。

[39] 胡又天，流行詞話，第三期，2011。

[40] 陳富容,現代華語流行歌詞格律初探,逢甲人文社會學報,第 22 期,第 75-100 頁，2011。

[41] 樂句(Phrase)，http://en.wikipedia.org/wiki/Phrase_(music)

國立政治大學 National Chengchi University