電腦輔助句子重組試題編製[1]

黃志斌　　劉昭麟　　郭韋狄　　孫瑛澤　　賴敏華
國立政治大學 資訊科學系
{g9614, chaolin, s9441, s9538, g9523}@cs.nccu.edu.tw

摘要

句子重組試題要求受測者把所給的一組詞彙組合成特定詞序的正確語句，是語文測驗常見的試題類型，可以檢驗受測者的句型和文法知識。我們試圖利用文法分析工具，協助出題教師編製句子重組試題。編製句子重組試題的第一個工作是適當地切割語句，以控制試題難度；一個好的剖析器就足以提供很好的服務。切割所得的詞彙集合，常常可以組合成與教師所欲學生回答的正確語句之外的合法語句。透過限制試題詞彙集的相對位置，我們可以限制學生的答案。我們的研究經驗顯示要自動協助出題教師預示所有可能的合法詞序卻是一件艱難的工作，而且這一研究問題與語法學有密切關係。本文詳述我們利用史丹佛剖析器，建構英文句子重組試題編製環境的經驗與實際系統。

關鍵詞：電腦輔助語文教學、文法學習輔助、句子重組練習、連結樹句法

1. 簡介

語文是人們溝通的基礎，爲了避免詞序的使用錯誤以及恰當的使用語言，本文將探討如何利用軟體技術建構出句子重組試題編製環境，以供學生練習語文的正確詞序之用。

句子重組試題是將句子打亂詞序後讓學生重新組合詞序的測驗試題，不過在重新組合詞序的時候有可能會遭遇到下面的問題。如「今天我被狗咬了。」與「I put a book on a table.」這兩句打亂詞序產生句子重組試題之後，學生除了排出原本的句子之外，也可以排出「今天狗被我咬了。」與「I put a table on a book.」在文法上合法但語意上卻有微妙差異的同字詞不同詞序組合。

目前在中文句子重組出題方面，香港宣道會葉紹蔭紀念小學提供的中文科網上學習網頁[1]裡有中文句子重組的練習題；在英文句子重組出題方面，台灣成德國小提供的英語檢測練習網頁[2]裡有句子重組的練習題。我們刻意地排出非這兩個系統預設的答案卻又是合理的句子來測試，結果發現此兩系統均無法辨別其正確性。另外，王昱鈞等學者[3]也曾提出過句子重組系統，該系統以語境(context)的方式提供了句子重組之線索來引導學生答出系統預設的答案，不過仍無法避免學生答出預設答案之外的合理句子，同時也無法辨別這些句子的正確性。考慮到建置所有同字詞不同詞序的答案之成本，本論文設法尋求其它方法，讓不同的學生都只能排出相同的答案。

建置句子重組試題編製環境第一個要面臨到的問題就是如何才能讓不同的學生都只能排出某些特定的答案。爲了稱呼上的方便，原始句子內的詞序如果經過調動，我們就

[1] 本篇論文爲符合論文集頁數限制而刪減爲 14 頁，限於篇幅不能在本文中全面交代相關細節。我們另外準備了一份較詳細的 25 頁版本之論文，公開在 http://www.cs.nccu.edu.tw/~chaolin/papers/rocling_huang.pdf

稱詞序調動後的句子爲***亂序句***(deranged sentences)。若亂序句仍然符合文法規範的話，這樣的亂序句就稱爲***重組句***(scrambled sentences)。Becker 等學者[4]是尋找重組句的先進，他們爲了解決在德語上使用「Tree Adjoining Grammars」(TAG)[5]分析重組句時的不足，將 TAG 擴展成「Multi-Component TAG」與「Free Order TAG」來分析與預測可能的重組句。我們打算將某些字固定在特定的詞序上，藉此來過濾掉其它合法的同字詞不同詞序之答案(也就是重組句)，而這些被固定在特定詞序上的字我們就稱爲***錨字***(anchors)。當一個句子在固定某些字的詞序之後，如果不同的學生只能排出一樣的答案的話，那麼我們的目的就達成了。

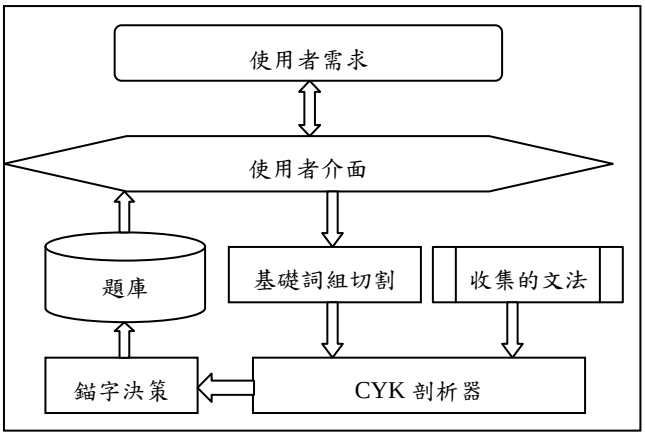

圖 1、編製環境架構圖

重組試題編製環境的第二個問題就是如何利用電腦輔助篩選出可能的重組句？我們將一句 *n* 個字的句子透過「基礎詞組」的幫助把這個句子切割成 *m* 個區塊($n \geq m$)，並藉由這 *m* 個區塊排列出 *m*!(*m* 階乘)個句子。然後，經由「史丹佛剖析器」(Stanford Parser)[6]，從中收集用來剖析(parse)的文法，接著將這些文法提供給利用動態規劃剖析演算法「Cocke-Younger-Kasami Algorithm」(簡稱 CYK 演算法)[7][8]所組建的剖析器，來對這 *m*!個句子篩選出可能的重組句，這些等待 CYK 剖析器篩選的句子我們就稱爲***候選句***。篩選的方式則是利用 CYK 剖析器，依據所給定的文法規則，來檢驗所輸入的候選句是否合乎文法；若候選句可以被完整剖析，就通過篩選，反之則被篩選掉。

圖 1 爲本論文之編製環境架構圖。當教師想要建置句子重組題目時，可以透過我們所提供的使用者介面輸入想要做爲題目的句子，這個被輸入的句子經過基礎詞組切割的流程後所排列產生出的句子，會和預先收集好的文法一起送入 CYK 剖析器篩選出重組句，接著透過錨字的決策固定住某些字，最後再存入題庫，供教師編輯或學生練習。

我們會在第二節說明基礎詞組的切割並呈現相關數據的分析，然後在第三節簡述我們收集到的文法和相關實驗的結果，接著在第四節討論錨字決策，再來我們在第五節展示編製環境提供給教師編輯與學生練習介面，最後第六節則是簡短的結語。

2. 切割基礎詞組與篩選重組句

我們在第一小節介紹基礎詞組切割的想法，然後在第二小節提出尋找基礎詞組與合併詞組的方式，在第三小節簡單介紹如何利用機率分數給亂序句排序，第四小節則是篩選門檻值與合併詞組的比較實驗。

2.1 基礎詞組與重組句的關聯

表 1 是假設教師以(1a)的句子輸入使用者介面做爲原始句所產生出的一個重組句範例，(1a)、(1b)以及(1c)這三句都是重組句。同時仔細觀察這三句，都擁有基礎詞組「good students」。我們企圖透過重組句中的基礎詞組，發展出找重組句時較好的方式。

最直覺的做法就是將一串 *n* 個字的英文句打亂詞序之後重新排列的 *n*!個句子透過 CYK 剖析器來篩選可能的重組句。但打亂詞序之後重新排列產生「10!」個句子，就表示剖析器就必須處理「10!」(3628800)個候選句，其總花費的時間或許會令出題教師感到不耐。

表 1、簡單的重組句範例

1a	I have never seen such good students.
1b	Such good students I have never seen.
1c	Never have I seen such good students.

爲解決這樣子的問題，我們觀察到了基礎詞組的一般性概念，即句子中最小的形容詞片語(adjective phrase，ADJP)、名詞片語(noun phrase，NP)以及介系詞片語(prepositional phrase，PP)的結構。在一個句子中可能只有一個基礎詞組，也可能有很多個基礎詞組，在表 1 中(1a)裡的基礎詞組只有「good students」。而在(1a)打亂詞序後，除了可以排成原本的句子(1a)之外，也可以排成(1b)或是排成(1c)包含「good students」的重組句。但在文法的規範下，重組句不會出現「Students good such seen never have I.」這種不包含「good students」的句子。也就是說，不同的重組句之間詞序的調動並不會把基礎詞組作進一步的切割。

透過這樣的想法，在表 1 的(1a)在排列成亂序句時，若將「good students」綁成一個區塊，CYK 剖析器處理的句子就可從原本的「7!」(5040)句減爲「6!」(720)句。藉由這樣子的方式就能省下不少時間。

2.2 詞組分析與合併

我們在 2.2.1 節中探討如何分析句子中的基礎詞組，2.2.2 節則討論當句子經過基礎詞組切割程序後的區塊數目仍然相當多時，使用合併詞組來處理的程序。

2.2.1 詞組分析

我們利用史丹佛剖析器分析試題編輯者輸入的句子，在取得機率分數最高的結構樹後，再根據該結構樹的內部結構來擷取基礎詞組。

圖 2 是用史丹佛剖析器分析表 1 中的(1a)所得到的機率最高之結構樹結構(詞性標記的部分請參照[9] Penn Treebank Tags)。爲了能夠精確地分析基礎詞組，我們設計了圖 3 的尋找基礎詞組的演算法，用來從結構樹中分析基礎詞組，並搭配圖 2 的結構樹來說明演算法的程序。首先根據圖 3 演算法的第 4 列在圖 2 的結構樹中尋找「ADJP」、「NP」、「PP」的蹤影，而在圖 2 我們找到了第四層的「PP」(假設節點「ROOT」爲第零層)，以及第二層和第五層的「NP」。但我們要找的「ADJP」、「NP」、或「PP」其子樹必須爲單字層級，如同圖 3 中第 5 列所述；然而第四層的「PP」其左子樹雖爲單字層級，右子樹卻爲「NP」的片語層級，也因此第四層的「PP」並不是我們想尋找的基礎

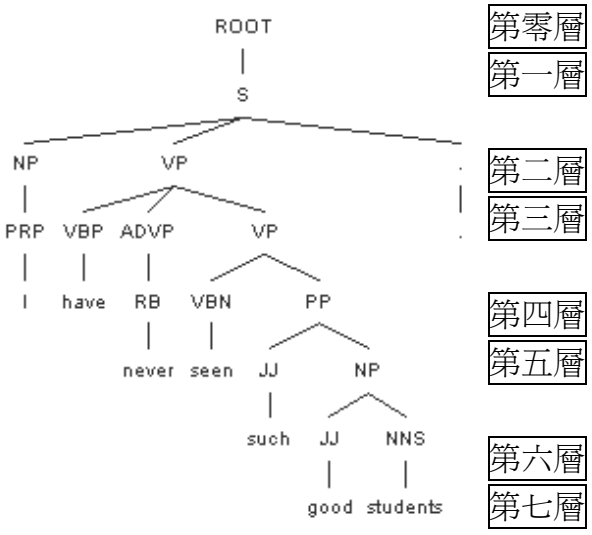

圖 2、史丹佛剖析器產生的結構樹範例

```
輸入：英文句 E={w_1, ..., w_n}及其結構樹 T。
輸出：英文句 E 中的所有基礎詞組。
程序：
在 T 中尋找所有的 ADJP、NP 以及 PP。
如果某個 ADJP、NP 或 PP 在 T 中的子節點為 E 中的單字層級，
    那麼就將這個 ADJP、NP 或 PP 視為基礎詞組。
```

圖 3、尋找基礎詞組的演算法

詞組，第二層和第五層的「NP」才符合我們要找的基礎詞組。所以我們就將「I」和「good students」視爲基礎詞組，並把基礎詞組分別綁在一起產生亂序句。透過這種方式，我們就可以將表 1 中的(1a)切成「I」、「have」、「never」、「seen」、「such」以及「good students」這六個區塊來產生「6!」個候選句。

但表 1 中的(1a)是一個較簡單的句子，假若輸入一個 10 個字的句子，而且這個句子經過基礎詞組切割的程序之後產生了 8 個區塊，就代表我們還是得產生「8!」(40320)個候選句傳給 CYK 剖析器處理。剖析器要處理的句子越多，所花費的時間越長，同時剖析句長爲 n 的時間複雜度(computational complexity)爲$\theta(n^3)$。因此我們設計了兩個簡單的處理方法，試圖減少時間的花費。

2.2.2 合併詞組

若是句子經過基礎詞組切割後的區塊數仍很多的話，則用合併詞組來處理。我們從結構樹中最深的葉節點(leaf node)開始合併，直至亂序句降到理想的範圍內。

本論文合併詞組的方式是取由下而上(bottom up)的方式來實作。我們首先尋找結構樹中深度(depth)最深的詞組，接著再將該詞組向上拉高一層來合併，如果區塊還很多，就重複「尋找」和「合併」的動作，直至區塊數 m 降到理想的範圍內。我們用圖 2 的結構樹和圖 3 的演算法找到的基礎詞組，以及圖 4 合併詞組的演算法來說明這樣的方式。

```
輸入：英文句 E={w_1, ..., w_n}與其結構樹 T，英文句 E 的基礎詞組集合
    K={k_1, ..., k_y}，以及英文句 E 經過基礎詞組分析後切割出的
    區塊數 m，y≤m≤n。
輸出：集合 K
宣告：
令 K 中最深的詞組為k_h。
令 near 為與k_h同子樹且互為兄弟節點的相鄰詞組或英文字，|near|≤2。
隨機函數 R：隨機取出集合中的某個元素，每個元素被取出的機會相等。
參數：u，u 為理想的範圍。
程序：
當 m≥u 時，則重複以下步驟：
    如果集合 K 中最深的詞組不只一個，
        則透過 R 從這些同是最深的詞組集合中選一個設為k_h。
    在 K 中移去k_h。
    將k_h與此k_h的 near 合併成新詞組，
        但若是此k_h的|near|等於 2，
            則透過 R 從此k_h的 near 集合中選一個與k_h合併成新詞組。
    將合併後的新詞組加入 K 中。
    區塊數 m 減 1。
```

圖 4、合併詞組的演算法

首先從圖 3 輸出的基礎詞組中尋找最深的詞組，在圖 2 的結構樹中即是(NP(JJ good)(NNS students))，接著將此詞組與其 *near* 合併來達到縮減區塊數 *m*，以求將 *m* 降到合理的範圍內。在圖 4 中，這個合理的範圍被參數化爲第 7 列的 *u*；而 *near* 則被描述在第 5 列，指的是與最深的詞組k_h同子樹且互爲兄弟節點的鄰接詞組或英文字，同時因爲與k_h鄰接的位置只有k_h的左相鄰與右相鄰兩個位置，所以第 5 列也明列出$|near| \leq 2$。我們透過第 9 列到第 17 列的迴圈來重複從詞組集合 *K* 中尋找最深的詞組k_h，以及將最深詞組k_h和此k_h的 *near* 合併，最後，當區塊數 *m* 降到理想目標，則將集合 *K* 做爲輸出。

在這個迴圈裡面，第 13 列會將最深詞組k_h及同子樹且互爲兄弟節點的鄰接詞組或英文字(即此k_h的 *near*)來合併，以在圖 2 中最深的詞組(NP(JJ good)(NNS students))來說，找到的 *near* 就是(JJ such)，於是就合併成(PP(JJ such) (NP(JJ good)(NNS students)))，並在第 16 列加入詞組集合 *K* 中，做爲下一次尋找最深詞組的候選人。這樣就可以把「such good students」綁在一起，如此一來變成「I」、「have」、「never」、「seen」以及「such good students」五個區塊，產生「5!」的候選句。

另外，有兩個問題的處理原則是必須額外交代的。第一個問題是當合併區塊時，若同時有多個基礎詞組在結構樹中都是最深的，那應該挑選何者來合併。第二個問題則是我們設定的區塊數並不是每一個結構樹都能找到剛好的切分點來符合我們的要求。

第一個問題可能沒有標準的答案，我們是透過一個隨機函數 *R* 來從多個最深的詞組中挑選一個。這個 *R* 被描述在圖 4 中的第 6 列，每個詞組被取出的機會均等。隨機的機制則在圖 4 中的第 10 到 11 列中說明。

而第二個問題目前也沒有標準答案，也是由隨機函數 *R* 來決定哪兩個相鄰的區塊要做合併的動作。以三個連續區塊 A、B 和 C 來說，我們先用解決第一個問題使用的隨機機制挑選其中一個最深的區塊，如果挑到 A 或 C 就沒有合併的困擾，因爲此時兩者的 *near* 均爲區塊 B(即|*near*|等於 1)，無論挑到哪個都只能和 B 合併。但若挑到區塊 B 的話，便有 A 和 C(|*near*|等於 2)，這是又引出了隨機的機制，進入第 15 列用隨機函數 *R* 來選擇要合併的是哪個區塊。

我們在此試圖用((green)(and)(pink))這個小結構來說明合併成兩個區塊時，上述兩個問題的實際情況。首先「green」、「and」和「pink」都是最深的詞組，那麼我們就隨機選擇一個來合併；若是隨機選到「green」或「pink」都只能和「and」合併產生「『green and』、『pink』」或「『green』、『and pink』」；而若是隨機選到「and」的話則可再隨機選擇與「green」或「pink」合併，產生「『green and』、『pink』」或「『green』、『and pink』」。

2.3 利用機率分數篩選重組句

當史丹佛剖析器處於「probabilistic context-free grammar」(簡稱 PCFG；也稱爲 stochastic context-free grammar，SCFG)[7][8]的模組下，當輸入一個句子後，就可以操作參數來產生出許多該句的可能結構樹結構，同時每個結構樹都帶有一個機率分數，該機率分數代表著句子被剖析爲對應的結構樹之機率有多少。

我們利用結構樹機率分數越高，合法句的機會也越高的想像，將切割後的區塊所產生的亂序句輸入史丹佛剖析器，擷取每句亂序句中最高的機率分數，同時依此機率分數將亂序句排名。相關的實驗我們將在 2.4.1 節中提出。

2.4 篩選門檻值與合併詞組的實驗

我們在 2.4 節中進行比較實驗。2.4.1 節探討機率分數排名的門檻值，2.4.2 節分析合併詞組後產生的亂序句是否能涵蓋重組句，2.4.3 節則是篩選門檻值與合併詞組的綜合實驗。

2.4.1 篩選門檻值的比較實驗

此節是用基礎詞組切割的區塊產生出所有的亂序句之後，利用史丹佛剖析器所給的機率分數，把亂序句排名次然後再用門檻取出特定數量的亂序句做爲候選句，藉此降低 CYK 剖析器花費在篩選的時間。

表 2、實驗資料區塊切分

	4 區塊	5 區塊	6 區塊	7 區塊	8 區塊
7 字	7 句	9 句	13 句	7 句	0 句
8 字	1 句	9 句	7 句	11 句	7 句
9 字	0 句	1 句	10 句	11 句	7 句

我們進行了一個排名門檻的實驗。實驗對象是 100 個從網路上隨機擷取的句子，當中包含了 36 個 7 字句、35 個 8 字句、以及 29 個 9 字句，並對這些句子用基礎詞組的方式切割，形成 4、5、6、7、8 五種區塊類別，如表 2 所示。表 2 裡第二列第四行的「13 句」表示「有 13 句 7 個字的句子被切分成 6 個區塊」，其他依此類推。

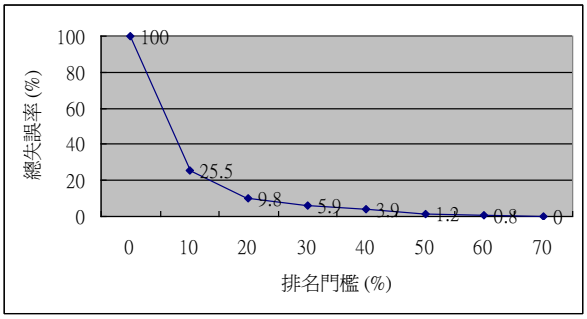

圖 5、排名門檻總失誤率的實驗數據

我們也用人工的方式盡可能地尋找這 100 句的重組句，並對重組句組用史丹佛剖析器的機率分數做排名百分比後，進行門檻值的分析。

圖 5 即爲排名門檻的實驗數據，橫軸代表著我們取句子產生的亂序句中「前 10%」、…、「前 70%」來篩選的門檻值，縱軸則是以「255 句重組句」爲分母，「未在門檻內的重組句數」爲分子的總失誤率。這 100 個重組句組一共有 255 句，因此當門檻值爲「0%」，在圖 5 中會對應到有 255 個重組句不在我們所取的門檻內；而當我們的門檻值設爲「70%」的時候，則可以將 255 句全部囊括進來。

雖然取「70%」爲門檻值是個穩定作法，但 7 區塊句子有 5040 個亂數句，取「70%」的句數(3528 句)對於 CYK 剖析器來說是一個不小的篩選數量。或許可以嘗試用較低的門檻值來篩選。若以總失誤率來看的話，將門檻值設爲「20%」失誤率有 9.8%，設在「30%」、「40%」的話失誤率則有 5.9%、3.9%。

雖然將門檻值由「70%」改爲較低的門檻值會有失誤率的風險存在，但卻可以爭取到計算時間上的效益。也就是說門檻值的選定並沒有絕對的標準，而是可以有彈性變化的。

2.4.2 合併詞組的比較實驗

我們利用表 2 中第四列第六行的那 7 組重組句組(參見附錄二)做爲實驗語料，分析合併詞組後產生的亂序句是否能涵蓋重組句。

我們將 7 組重組句組裡的 7 句原始句使用圖 4 提到的演算法從 8 個區塊分別合併成 7、6、5 和 4 個區塊的句子，接著再以這些句子各自產生亂序句。實驗目的在分析這些

表 3、合併詞組實驗

	合併區塊數			
	7	6	5	4
1.1	T	T	T	T
1.2	T	T	T	F
2.1	T	T	T	T
2.2	T	F	F	F
2.3	T	T	T	T
2.4	T	F	F	F
3.1	T	T	T	T
3.2	T	T	T	T
3.3	T	F	F	F
4.1	T	T	T	T
4.2	T	T	T	F
5.1	T	T	T	T
5.2	T	T	T	T
6.1	T	T	T	T
6.2	T	T	T	F
6.3	T	T	T	T
7.1	T	T	T	T
7.2	T	T	F	F
7.3	T	F	F	F

表 4、綜合實驗數據

	合併區塊數							
	7		6		5		4	
	排名	排名百分比	排名	排名百分比	排名	排名百分比	排名	排名百分比
1.1	468	9.29	69	9.58	16	13.33	2	8.33
1.2	208	4.13	37	5.14	8	6.67	F	F
2.1	19	0.38	2	0.28	1	0.83	1	4.17
2.2	7	0.14	F	F	F	F	F	F
2.3	486	9.64	78	10.83	15	12.5	6	25
2.4	487	9.66	F	F	F	F	F	F
3.1	2	0.04	2	0.28	2	1.67	2	8.33
3.2	3	0.06	3	0.42	3	2.5	3	12.5
3.3	965	19.15	F	F	F	F	F	F
4.1	3	0.06	1	0.14	1	0.83	1	4.17
4.2	12	0.24	10	1.39	2	1.67	F	F
5.1	3	0.06	2	0.28	1	0.83	1	4.17
5.2	64	1.27	40	5.56	16	13.33	6	25
6.1	312	6.19	79	10.97	37	30.83	6	25
6.2	163	3.23	37	5.14	23	19.17	F	F
6.3	1272	25.24	248	34.45	78	65	15	62.5
7.1	97	1.92	27	3.75	6	5	1	4.17
7.2	98	1.94	28	3.89	F	F	F	F
7.3	605	12.00	F	F	F	F	F	F

各自產生的亂序句是否能涵蓋重組句，如果包含指定的重組句的話，則標記「T」；反之則標記「F」。實驗數據如表 3，最上面的「7」、「6」、「5」和「4」即爲將原始句合併成 7、6、5 和 4 個區塊數來產生亂序句。

從表 3 中可以看到，當區塊數目越合併越少的時候，所產生的亂序句能夠涵蓋的重組句也就越來越少。因此當我們在圖 4 中考慮將區塊數 m 降到合理的範圍 u 時，也和 2.4.1 節的門檻值一樣，是可以有彈性變化的。

2.4.3 篩選門檻值與合併詞組的綜合實驗

我們把表 3 中 7、6、5 和 4 個區塊各自產生的亂序句，再用史丹佛剖析器產生這些亂序句的機率排名，並使用 2.4.1 節討論的門檻值來分析是否能涵蓋重組句。

表 4 即爲篩選門檻值與合併詞組的綜合實驗數據。表 4 中「合併區塊數」下方的「7」、「6」、「5」和「4」與表 3 的意義相同；「排名」表示該重組句在所屬的亂序句中的機率分數排名；「排名百分比」則是該重組句在所屬的機率分數排名中的百分比。由於表 4 是承接表 3 所進行的實驗，因此表 3 裡「F」代表沒有被涵蓋在亂序句中的重組句，在表 4 裡也不會有機率分數，所以也用「F」表示無法排名。

如果機率分數的門檻值設在「30%」與「40%」的話，這 19 個重組句在「7」、「6」、「5」和「4」個區塊數所對應的總失誤率(表 4 標記爲「F」的也算失誤)爲 0%、26.3%、36.8%和 47.4%，以及 0%、21.1%、31.6%和 47.4%。這樣子的結果顯示出區塊數與門檻值稍高，總失誤率就能夠稍微降低，但要付出的代價則是 CYK 剖析器從候選句中篩選出可能的重組句之花費時間。

3. 文法分析

由於 CYK 剖析器需要文法來篩選亂序句，但我們缺乏現成的文法規則資料，於是想到了利用語料來獲取文法規則。我們在第一小節說明如何從史丹佛剖析器收集文法，接著在第二小節提報相關實驗的結果。

3.1 文法萃取

我們隨機從網路中收集七千多句國中英文課本的相關例句做爲語料，並將這些語料傳入史丹佛剖析器做結構樹的分析。接著從分析出來的結構樹中統計出現的文法規則，再將這些規則做爲 CYK 剖析器的文法輸入。

當我們在史丹佛剖析器中輸入表 1 中的(1a)時，就可以獲得圖 2 這個最高機率的結構樹。這個結構樹除了葉節點之外的內部節點分支其實都是一組文法規則，如第二層的內部節點「VP」擁有「VBP」、「ADVP」以及「VP」三個子節點，那麼就表示在這邊使用到了「VP → VBP ADVP VP」這樣子的文法規則來進行剖析。

我們對這七千多句的每一句的結構樹都做這樣子的分析，並做出一個統計資料，如圖 6。圖 6 的橫軸是我們統計相同文法規則出現的次數之後取其對數的值，縱軸則是出現一樣次數的不同文法規則之條數。

在這個數據中總共有 985 條文法規則，出現次數最多的前四名從第一名開始依序是 4450 次的「NP → PRP」、4329 次的「S → NP VP」、3079 次的「NP → DT NN」以及 2415 次的「PP → IN NP」。

第一名的文法說明了其相關例句不外乎圍繞在與人稱代名詞相關的句子，如「I love you.」等等之類的句子；第二名的文法則點出了這些例句是以直述句居多；第三、第四名則是告訴了我們簡單的名詞片語以及介系詞片語也是常出現於這些例句，如「an apple」、「in the morning」等等。另外我也可從圖 6 中得知大多數的規則都出現很少次，實際上真正常被拿來做剖析的文法只有 985 條中的一小部份而已。

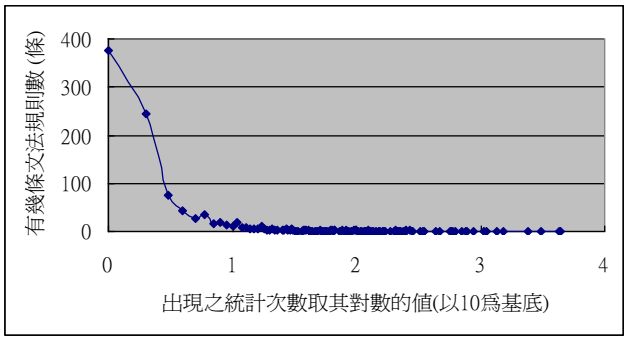

圖 6、文法規則的統計資料

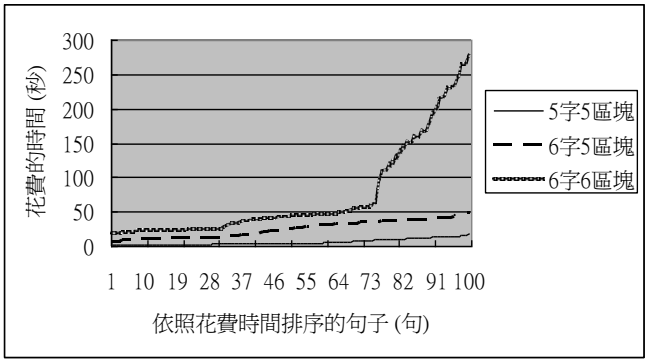

圖 7、所費時間之數據

我們在一部使用 2GB 的 RAM 和 Windows XP 的 Pentium 4 2.68G 機器上做了一個簡單的時間實驗，該實驗是在 CYK 剖析器使用上述的 985 條文法規則下，在網路上隨機找了 5 個字、6 個字的 100 個句子，並將 5 個字切成 5 個區塊、6 個字切成 5 和 6 兩種區塊產生亂序句，再把它們匯入 CYK 剖析器記錄所需時間。實驗目的在探討 CYK 對於同句數不同字數及同字數不同句數的篩選時間。圖 7 是這 200 句資料跑的三組實驗所得的花費時間之內部情況，縱軸是 CYK 篩選產生的亂序句時所花的時間，橫軸則是依照篩選每句產生的亂序句總花費時間由小到大排序。

我們從圖 7 的記錄中可以計算出 CYK 剖析器篩選 5 字 5 區塊、6 字 5 區塊以及 6 字 6 區塊的平均時間分別爲 6.4 秒、24.3 秒及 72.3 秒。雖然同樣都是切 5 區塊，但 6 個字的平均篩選時間是高於 5 個字的，可得知字數越多則所需時間越多。再來看到 6 字 5 區塊、6 字 6 區塊的篩選時間，可得知句數越多則所需時間越多。

值得注意的是，6 字 6 區塊的折線在最後的部份開始急速上升，這個奇怪的部份我們試著用以下的例子來說明。假設文法集中有很多規則，當中有四條規則爲「A → a c」、「B → a c」、「C → a c」、「D → a c」，同時我們有 3 個字的原始句「a b c」。如果將「a b c」切成 2 個區塊來產生候選句，無論什麼情形都不會引入上述的四條規則來剖析。然而當我們將「a b c」切成「a」「b」「c」這 3 個區塊來產生候選句時，就會出現「bac」和「acb」的組合，此時就會引入上述四條規則，所花費的時間也就可能以倍數成長了。我們推測圖 7 中 6 字 6 區塊的折線在最後急速上升的部份和上面例子的 3 字 3 區塊情況類似，是由文法集本身的規則所產生的時間倍數成長，如在我們所萃取的文法集當中的「NP VP」就有「SQ → NP VP」、「NP → NP VP」與「S → NP VP」這三條規則。

由以上我們對時間實驗數據的討論可知，句子字數越多、切分的區塊越多，那麼所花時間也會越長，同時與文法集的規則數也有關。

3.2 文法集分析

我們將 985 條文法規則分別取不同的範圍進行實驗，藉此比較不同範圍的文法集對 CYK 剖析器判斷重組句時的影響。

我們可以試著去想像，當文法的數目相當龐大複雜的時候，即使是亂組的亂序句，都有可能找到剛剛好的文法來剖析(這樣的問題在計算語言學裡面稱爲「overgeneration」[10])，使得 CYK 剖析器誤判爲重組句；又或者，當文法數目相當稀少的時候，即使是教師透過使用者介面輸入的原始句，都有可能找不到合適的文法來剖析，該原始句也會被 CYK 剖析器誤判成不合法。前述第一種情況，將導致 CYK 剖析器在過濾可能的重組句時太過寬鬆，使得我們必須產生較多的錨字(錨字的選擇問題將會在第四節仔細說明)，最後影響到的就是句子重組試題的效果。試想，如果一個句子重組試題有 *n* 個字，而我們固定住了 *n*-1 個字，這樣子的句子重組試題完全沒有意義。而第二種情況，則將導致 CYK 剖析器在過濾可能的重組句時太過嚴苛，使得最後甚至沒有任何亂序句能通過篩選。

我們對於這些通過篩選的亂序句中是否含有重組句，以及在面對真實生活中的句子時文法是否能夠篩選出可能的重組句感到興趣。於是從生活週遭的對話，以及網路文章中隨機挑選出來九個句子做爲測試語料，同時也用人工的方式找出每一個句子可能的重組句(參見附錄一)。實驗數據如表 5。表 5 中的「n」、「m」代表這個句子是 *n* 個字的句

表 5、測試通過篩選的亂序句中是否含有重組句的實驗數據

	n	m	前 150 條			前 300 條			前 450 條			全部 985 條		
			通過句數		包含在內	通過句數		包含在內	通過句數		包含在內	通過句數		包含在內
1.1	8	6	32	4%	T	78	11%	T	96	13%	T	240	33%	T
1.2					T			T			T			T
2.1	6	6	120	17%	T	120	17%	T	120	17%	T	120	17%	T
2.2					F			F			F			F
2.3					T			T			T			T
2.4					T			T			T			T
3.1	6	5	48	40%	T	48	40%	T	48	40%	T	72	60%	T
3.2					T			T			T			T
3.3					T			T			T			T
3.4					T			T			T			T
4.1	6	4	6	25%	T	6	25%	T	6	25%	T	6	25%	T
4.2					F			F			F			F
4.3					F			F			F			F
5.1	7	6	120	17%	T	120	17%	T	120	17%	T	120	17%	T
5.2					F			F			F			F
5.3					F			F			F			F
6.1	7	5	0	0%	F	0	0%	F	0	0%	F	0	0%	F
6.2					F			F			F			F
7.1	7	6	192	27%	T	192	27%	T	240	33%	T	240	33%	T
7.2					F			F			F			F
8.1	8	5	12	10%	T	18	15%	T	24	20%	T	24	20%	T
8.2					F			F			F			F
9.1	6	6	33	5%	T	120	17%	T	120	17%	T	120	17%	T
9.2					F			F			F			F

子且以基礎詞組方式切割成 *m* 個區塊來產生亂序句。左邊最上面的「1.1」與「1.2」代表除了原始句「1.1」之外，我們還用人工的方式找到了另外一句重組句「1.2」；同理，「2.1」、「2.2」、「2.3」以及「2.4」依此類推。而「前 150 條」…「全部 985 條」代表的是文法規則集中出現次數最高的前多少條文法規則。「通過句數」代表該限制的文法規則集下原始句產生的 *m*!個亂序句中到底有多少個能通過 CYK 剖析器的篩選；其下的左欄位是通過句數的實際句數，右欄位則是實際句數在 *m*!個亂序句當中所佔的比例。「包含在內」代表這些通過 CYK 剖析器篩選的亂序句中，是否包含了這一個重組句；若是包含在內，則標記「T」，反之則標記爲「F」。透過表 5 中的數據，我們做出了以下的推論與探討。

在這九組中，有某些句子即使放鬆了限制，通過 CYK 剖析器篩選的亂序句也不會增加。如第 8 組的重組句組，該句組的文法規則限制由「前 150 條」放鬆到「前 450 條」時，通過篩選的亂序句的確是增加的；但是再放鬆到「全部 985 條」時，數字卻維持在「前 450 條」時的 24 句。此現象驗證了我們在討論圖 6 時所論述的「常被拿來做剖析的文法，實際上只有 985 條中的一小部份而已」。

接著我們在表 5 中看到了一個特例，即第 6 句組。這句組相當奇特，不僅僅是 6.2 的重組句，連做爲原始句的 6.1 都沒辦法通過篩選。原始資料顯示，6.1 並非是一個簡單的人稱代名詞(如：「I」、「you」……等等)做爲主詞的句子。而國中程度的英文句大部份都是以簡單的名詞片語或人稱代名詞爲主詞的句子較多，這點也可以從被 CYK 篩選過後的原始資料中發現，以人稱代名詞做爲主詞開頭的句子，即使是不合法的亂序句，就算將文法規則限制在「前 150 條」，這些不合法的句子也可以通過篩選。所以當我們將超過國中英語課本難度之外的句子輸入時，就會被 CYK 剖析器擋下來而無法通過篩選。

4. 錨字決策

錨字決策即是在最少錨字的限制下，讓通過 CYK 剖析器篩選的亂序句中只有某些句子能夠符合該錨字限制的決策，並藉此滿足教師只要學生排出特定答案的要求。本節提出一個簡單的演算法來說明處理一道試題只有一個唯一答案的情形。

從表 1 的三個英文句來說，如果我們限制第二個字只能填入「have」的話，那麼就還要再限制其它詞序的字才能把這三句過濾到只剩下一句，因爲(1a)和(1c)會共同來競爭「第二個字只能填入『have』」這個條件；但是如果我們一開始就決定第三個字只能填入「never」的話，那麼能夠被排出來的句子就只剩(1a)這一句，同時我們的目的也就達成了。

由於教師從使用者介面輸入句子時，是希望獲得該句的句子重組試題，所以我們將教師輸入的原始句做爲目標句來產生亂序句。圖 8 是我們用來決定錨字的演算法。從 CYK 剖析器得到通過篩選的亂序句之後，就可以從這些亂序句中去統計原始句裡的字在原來位置出現的句數；爲了往後說明上的方便，我們稱這樣子得到的句數爲***對位數***。圖 8 中第 9 列就選取了對位數最少的非錨字(即不是錨字的字)固定詞序做爲錨字，並在集合 S 中留下其它和原始句在相同位置上有錨字的亂序句(圖 8 中第 13 列)，同時在集合 S 中刪去其他的亂序句。如果$|S|$大於 1 的話，那麼就重複執行第 8 列到第 13 列的步驟，直到這些亂序句被過濾後只剩下原始句。

如果教師從使用者介面輸入表 1 中的(1a)，同時假設通過 CYK 剖析器篩選的亂序句剛好也只有(1a)、(1b)和(1c)這三句的話，那麼此時圖 8 中$|S|$即等於 3，依照演算法的步

```
01  輸入：經過 CYK 剖析器篩選後的亂序句集合 S={s1, …, se}，
        e 為亂序句之數目。
02  輸出：集合 A
03  宣告：
04  隨機函數 R：隨機取出集合中的某個元素，每個元素被取出的機會相等。
05  輸出的錨字集合 A = {}
06  程序：
07  當|S|大於 1 時，則重複以下步驟：
08      對所有亂序句中的每一句計算原始句裡每個字的對位數。
09      選擇原始句中擁有最小對位數的非錨字作為錨字。
10          但若是擁有最小對位數的非錨字不只一個，
11              則用隨機函數 R 從這些擁有最小對位數的非錨字所成的
                集合中挑選一個做為錨字。
12      將錨字加入集合 A。
13      在 S 中留下擁有錨字的亂序句，刪除其他的亂序句。
```

圖 8、決定錨字的演算法

驟計算，得到原始句中「I」的對位數爲 1，「have」的對位數爲 2，「never」的對位數爲 1，「seen」的對位數爲 2，「such」的對位數爲 2，「good」的對位數爲 2，「students」的對位數爲2。然後依據程序在擁有最小對位數的非錨字中隨機選取一個字(假設選到「I」)，並把原始句第一個字「I」設爲錨字。而當「I」被取爲錨字之後，亂序句就只剩下原始句(1a)，此時|*S*|等於 1，程序就結束了。所以我們得到了錨字「I」的輸出。

此外，當我們尋找擁有最小對位數的非錨字時，如果擁有最小對位數的非錨字不只一個的話，那麼演算法將會在這些同是擁有最小對位數的非錨字中隨機選取一個字做爲錨字。這個隨機的機制是透過一個被描述在圖 8 中第 4 列的隨機函數 *R* 來完成，相關的程序則在圖 8 中第 10 到 11 列。實際的例子則可以看到上述我們假設選到「I」做爲錨字時，其實「never」也是錨字的候選人之一，因爲這兩個字的對位數同時都是最小的 1。

5. 使用者介面

我們分別爲教師、學生製作了不同的使用者介面。教師介面爲試題編輯主選單，選單中可以建立新的題目，或編輯、刪除已有題目。而學生介面爲試題練習主選單，選單中可以選擇已經建立好的題目來練習。

5.1 編輯介面

圖 9 是試題編製環境的實際編輯介面，第一個和第四個白色的文字編輯欄位允許教師填入語境線索以提供給學生做爲重組句子時的參考，王昱鈞等學者[3]也認爲這種語境誘答設計的想法是有其價值的。第二個和第三個欄位則是填入題目的內容以及要測驗的句子。

在圖 9 的最下方有三個功能性按鍵，分別是「自動完成編輯」、「手動完成編輯」、以及「放棄編輯」。當教師把四個欄位都填完後，可透過「自動完成編輯」讓此編製環境透過前面章節介紹的程序自動地選取錨字，使得學生在練習時只能排出測驗的句子；也可透過「手動完成編輯」，讓教師自己從通過 CYK 剖析器篩選的亂序句中挑選滿意的重組句做爲答案，以保持學生答題時的自由度。

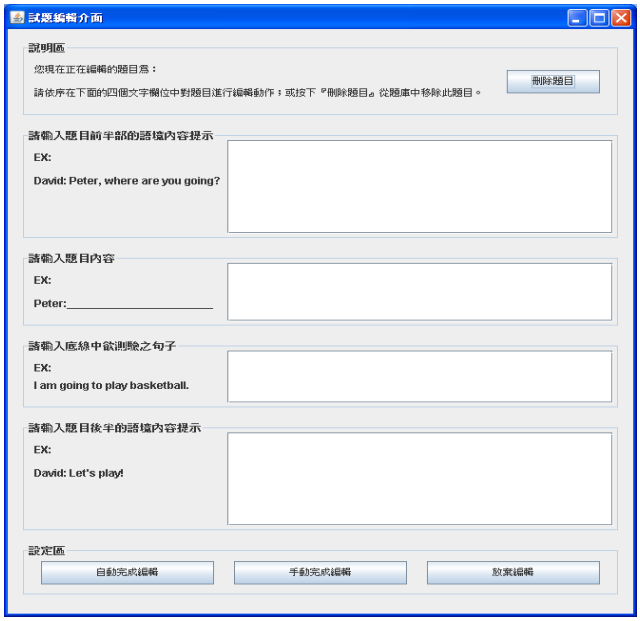

圖 9、編輯介面

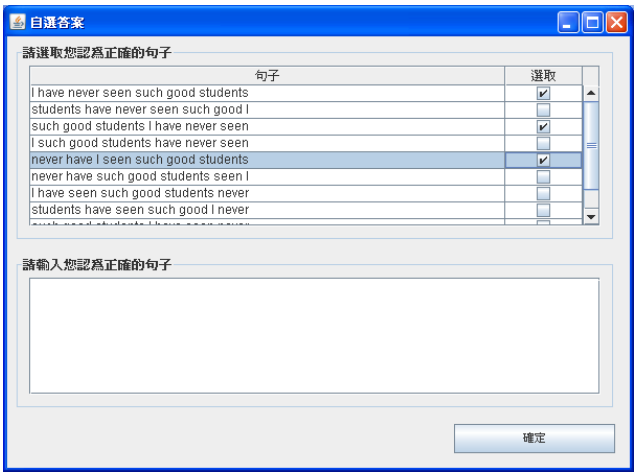

圖 10、挑選答案介面

圖 10 即爲按下「手動完成編輯」後的挑選介面，教師可在介面中選取想要的答案，也可以直接在圖 10 下方白色的文字編輯欄位中輸入答案。圖 9 最右上角的按鍵是「刪除題目」，可以用這個按鍵將編輯介面中的題目從題庫中刪除。圖 9 最右下角是「放棄編輯」，按下此鍵即不做任何更動並返回試題編輯主選單。

5.2 練習介面

當我們以學生的身分登入後，就會進入試題練習主選單。圖 11 的學生練習介面分爲最上面的「說明區」、中間的「作答區」和最下面的「解答區」三個主區塊。說明區是負責學生導覽的工作，藉由上面文字的說明來引導學生操作，同時該區還有一個「開始測驗」按鍵，按下該鍵後介面就會出現題目。

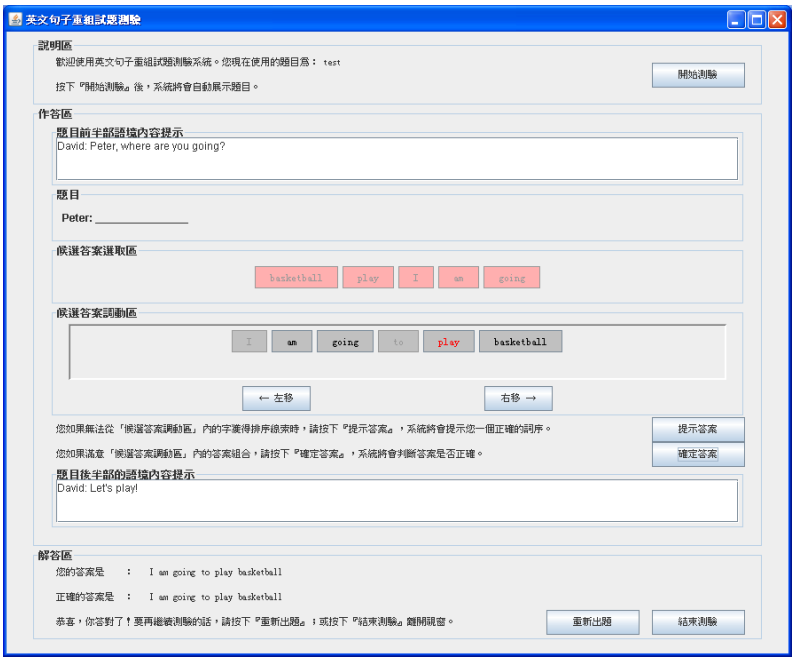

圖 11、練習介面

作答區中則包含了五個子區塊。教師之前在圖 9 的第一個和第四個欄位所填入的語境提示內容，就會出現在圖 11 作答區裡的第一個和第五個子區塊。我們希望透過前後文語境的提示，能夠帶給學生多一些解題的直覺。第二個子區塊則是放置題目的位置。要測驗的句子則在隨機打亂後放置在第三個子區塊中等著學生來挑選；錨字則是以淡灰色的字體顏色出現在第四個子區塊裡，同時無法透過「左移」或「右移」按鍵來移動錨字。

當學生從第三個子區塊中挑選某個字之後，該字會出現在第四個子區塊裡，並可用「左移」或「右移」按鍵來改變位置。我們在作答區中還設置了「提示答案」按鍵，若是學生對於題目不知從何下手，按下此鍵後介面就會在原始句裡面隨機挑選一個正確詞序的字來提示學生。最後的解答區則是比對題目的答案和學生排出的答案是否一致的地方，學生在這裡可以按下「結束測驗」返回試題練習主選單，或是按下「重新出題」繼續練習句子重組試題。

6. 結語

本篇論文報告了我們如何建構一個電腦輔助句子重組試題編製環境來幫助語言學習者掌握使用語言的能力。我們從產生可能的重組句出發，並嘗試用錨字的方式來解決以人工方式建置所有重組句成本過高的問題。透過相關的實驗數據分析，本編製環境的文法規則在處理國中程度的初學者入門句子時可能會稍微好些。

我們在實驗中看到了本編製環境的文法規則處理進階句子時所面臨的問題，因此未來會逐漸試著在國中英文課本相關的語料之外，收集其它較進階、或是現實生活中報紙

雜誌內文章的句子，來改善本編製環境文法規則的情況；又或者可以從剖析器的正確性著手，來探討文法規則剖析時的相關情況。同時未來在時間與精力的許可下，也會將本編製環境提供給教師與學生使用，並以問卷收集相關的數據來評估本編製環境的輔助效果。

致謝

本研究承蒙國科會研究計畫 NSC-97-2221-E-004-007-MY2 的部分補助謹此致謝。我們感謝匿名評審對於本文初稿的各項指正與指導，雖然我們已經在從事相關的部分研究議題，不過限於篇幅因此不能在本文中全面交代相關細節。

參考文獻

[1] 香港宣道會葉紹蔭紀念小學。中文科網上學習網頁。網址：http://www.casyymps.edu.hk/IT/Internet-Learning/chinese/chinese.htm (最後瀏覽日期爲 2009/01/04)。

[2] 台灣成德國小。英語檢測練習網頁。網址：http://w3.ctps.tp.edu.tw/today/teacher/hotexam/index.htm (最後瀏覽日期爲 2009/01/04)。

[3] M.-S. Lu, Y.-C. Wang, J.-H. Lin, C.-L. Liu, Z.-M. Gao, and C.-Y. Chang. Supporting the translation and authoring of test items with techniques of natural language processing, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 12(3), 234-242, 2008.

[4] T. Becker, A. K. Joshi, and O. Rambow. Long distance scrambling and Tree Adjoining Grammars, *Proceedings of the Fifth Conference on European chapter of the Association for Computational Linguistics*, 21-26, 1991.

[5] A. K. Joshi. An Introduction to Tree Adjoining Grammars, *Mathematics of Language*, 1987.

[6] The Stanford Natural Language Processing Group. The Stanford Parser: A statistical parser. http://nlp.stanford.edu/software/lex-parser.shtml (Last visited on 2009/05/10).

[7] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*, the MIT Press, 1999.

[8] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, 2000.

[9] Penn Treebank Tags. http://bulba.sdsu.edu/jeanette/thesis/PennTags.html (Last visited on 2009/08/9).

[10] D. Lin. Principle-based parsing without overgeneration, *Proceedings of the Thirty-First Annual Meeting on Association for Computational Linguistics*, 112-120, 1993.

附錄一

1.1 She can neither sing well nor dance beautifully.
1.2 She can neither dance beautifully nor sing well.
2.1 They are sometimes late for work.
2.2 Sometimes they are late for work.
2.3 They are late for work sometimes.
2.4 They sometimes are late for work.
3.1 Only I saw the cake yesterday.
3.2 I only saw the cake yesterday.
3.3 I saw only the cake yesterday.
3.4 I saw the cake only yesterday
4.1 Solemnly the minister addressed her congregation.
4.2 The minister solemnly addressed her congregation.
4.3 The minister addressed her congregation solemnly.
5.1 I have never seen such good students.
5.2 Such good students I have never seen.
5.3 Never have I seen such good students.
6.1 One of my favorite hobbies is reading.
6.2 Reading is one of my favorite hobbies.
7.1 He goes to the library every Sunday.
7.2 Every Sunday he goes to the library.
8.1 The hotel is next to a movie theater.
8.2 A movie theater is next to the hotel.
9.1 I can agree in neither case.
9.2 In neither case can I agree

附錄二

1.1 Fashion goes hand in hand with compassion for life.
1.2 Compassion for life goes hand in hand with fashion.
2.1 Girls are born with more sensitive hearing than boys.
2.2 Boys are born with more sensitive hearing than girls.
2.3 Born with more sensitive hearing than boys are girls.
2.4 Born with more sensitive hearing than girls are boys.
3.1 Boys and girls should be educated in different ways.
3.2 Girls and boys should be educated in different ways.
3.3 In different ways should boys and girls be educated.
4.1 Neither he nor I want to attend the meeting.
4.2 Neither I nor he wants to attend the meeting.
5.1 He finally passed the exam because he studied hard.
5.2 Because he studied hard he finally passed the exam.
6.1 Here is some good food for you to try.
6.2 Here some good food is for you to try.
6.3 Here some good food for you to try is.
7.1 A brown and white dog is at your doorsteps.
7.2 A white and brown dog is at your doorsteps.
7.3 At your doorsteps is a brown and white dog