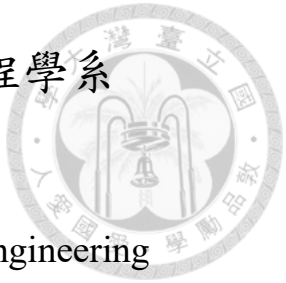


國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

基於隱藏式馬可夫模型的中文改錯

HMM-based Chinese Spelling Check

李啟維

Chi-Wei Lee

指導教授：張智星 博士

共同指導教授：張俊盛 博士

Advisor: Jyh-Shing Roger Jang, Ph.D.

Co-Advisor: Jason S. Chang, Ph.D.

中華民國 106 年 7 月

July 2017

doi:10.6342/NTU201701112

口試委員審定書

國立臺灣大學碩士學位論文
口試委員會審定書

基於隱藏式馬可夫模型的中文改錯

HMM-based Chinese Spelling Check

本論文係李啟維君（學號R04922125）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 106 年 7 月 13 日承下列考試委員審查通過及口試及格，特此證明

口試委員：張智星

（指導教授）

廖元甫　王逸如

張俊盛

系主任　趙坤茂

doi:10.6342/NTU201701112

誌謝

謝謝張俊盛老師、張智星老師、何總經理、清大 NLP 實驗室的同學們、台大 MIRLAB 的學長同學們、古君葳同學、斯文同學及我的家人朋友們。

doi:10.6342/NTU201701112

摘要

本篇論文透過聯合報提供的改稿記錄分析記者們的錯字與一般撰寫者的差異性，發現其中的修改主要是因記者們的需求而生，例如因報紙版型的數字轉國字，及一些句子風格的俗化，而到一些異體字的出現，最後一大塊屬於詞與詞之間極容易搞混的案例，如”紀錄／記錄”。從中可看出，相較於之前的改錯字資料，偏向於較年幼或初學中文的人，聯合報的錯字範圍性更廣，不只有形音的錯字，更有許多更實際的修改。在這之中，我們挑選出數千個標準的句子，做為第一個專門檢測專業編輯者中文系統的標準測試集。

本文亦整合了字的形音及相關特徵，透過 SVM 訓練分類器，並依此分類器建立新的錯字更正集，訓練後的錯字更正集整體搜尋時間下降許多。在系統上導入 Noisy Channel Model 與 Language Model 的句子計分方式，並比較 HMM 與 Beam Search 的差異，發現 Beam Search 的結果優於 HMM。

關鍵字：中文改錯，隱藏式馬可夫模型，集束搜尋，向量支持機，雜訊通道模型，語言模型

doi:10.6342/NTU201701112

Abstract

First, we extracted the typos from UDN edit log, and do some analysis. By the above data, we create the first benchmark to examine the Chinese Spelling Check system for professional editor, like journalist, writer and so on. Second, we build a new confusion set which can reduce search time. By extracting the features from all the pairs of Chinese character, we can train a SVM classifier to explore potential confusion set based on known typos table. Last, we compared the result between HMM and beam search. With language model and noisy channel model, we tune the parameter to find the best accuracy from our benchmark. We found that beam search work much better than the method of HMM.

Keywords: Chinese Spelling Check, Hidden Markov Model, beam search, Noisy Channel Model, Language model

doi:10.6342/NTU201701112

目　　錄

口試委員審定書 ii
誌謝 iii
摘要 iv
Abstract v
表格目錄 viii
圖表目錄 x
第１章 緒論 1
　第１節 研究動機 1
　第２節 研究方向 2
　第３節 章節概要 4
第２章 文獻回顧 5
第３章 研究方法 7
　第１節 N 元語言模型 (N-gram Language Model) 8
　第２節 雜訊通道模型 (Noisy Channel Model, NCM) 9
　第３節 隱藏式馬可夫模型架構 (Hidden Markov Model, HMM) 10
　第４節 集束搜尋 (Beam Search) 12
　第５節 重排序 (Re-ranking) 13
　第６節 錯字更正集 14
　　第１項 聲音相近 14
　　第２項 形狀相近 15
　　第３項 Fast Text 16
　　第４項 錯字更正集的生成 17
　　第５項 雜訊通道模型的機率 19
第４章 實驗結果 20
　第１節 實驗資料 20
　　第１項 中研院平衡語料庫 20

doi:10.6342/NTU201701112

第 2 項 Unihan 資料 20
第 3 項 SIGHAN 2013 資料 20
第 4 項 錯字勘誤表 20
第 5 項 聯合報改稿記錄 20
第 2 節 聯合報標準測試集建立 (UDN) 21
第 1 項 原始資料格式 21
第 2 項 錯字萃取方式 23
第 3 項 錯字類型 24
第 4 項 測試集的建立 28
第 3 節 錯字更正集的生成 30
第 4 節 評估指標 34
第 5 節 評測結果 36
第 1 項 HMM 的比較 36
第 2 項 Beam Search 39
第 3 項 HMM 與 Beam Search 的比較 42
第 4 項 重排序加入 42
第 5 章 結論及未來展望 45
第 1 節 結論 45
第 2 節 未來展望 45
參考文獻 46

doi:10.6342/NTU201701112

表格目錄

表格 1 繁體中文的常用輸入法.....3
表格 2 「金」的雜訊通道模型在錯字更正集.....10
表格 3 HMM 的分數計算.....12
表格 4 相似音的規則.....15
表格 5 倉頡碼的相似度規則.....16
表格 6 （世／氏）的字對特徵值.....17
表格 7 “世”的雜訊通道模型.....19
表格 8 FONT 的標籤屬性.....22
表格 9 錯字的取法.....23
表格 10 聯合報的前 10 常見修正.....24
表格 11 聯合報的前 10 常見修正（去掉數字的轉換）.....26
表格 12 互換類型的修正.....27
表格 13 〈紀，記〉詞的相關資訊.....28
表格 14 錯字的類型.....28
表格 15 編輯者修改的類型.....29
表格 16 文字表.....30
表格 17 部件“青”的相關字.....31
表格 18 倩在 SIGHAN 的相似音相似字.....31
表格 19 教育部錯別字表.....32
表格 20 青與其他字的特徵值表.....32
表格 21 各種錯字更正集的資訊及在標準測試集的涵蓋率.....33
表格 22 混淆矩陣 (Confusion Matrix).....34
表格 23 五種評測的標準.....34

doi:10.6342/NTU201701112

表格 24 評測標準的範例.....35
表格 25 評測標準範例的結果.....35
表格 26 HMM 在 Bigram 的結果.....37
表格 27 HMM 在 Trigram 的結果.....38
表格 28 Beam Search 在堆疊 10 的結果.....40
表格 29 不同的堆疊大小比較.....42
表格 30 在 HMM 上外加重排序.....43
表格 31 斷詞系統比較.....44

doi:10.6342/NTU201701112

圖表目錄

圖表 1 改錯字.....2
圖表 2 中文改錯字系統的架構.....7
圖表 3 HMM 在中文改錯字.....12
圖表 4 Skip-gram 的基本架構.....17
圖表 5 錯字更正集自動擴增的流程圖.....18
圖表 6 聯合報頭版的橫式標題直式內文.....25
圖表 7 聯合版全台焦點的橫式標題及內文.....25
圖表 8 HMM 在 Bigram 的分析圖.....37
圖表 9 HMM 在 Trigram 的分析圖.....38
圖表 10 HMM 的 Bigram 與 Trigram 的比較（FPR, F1-Score）.....39
圖表 11 HMM 的 Bigram 與 Trigram 的比較（Recall, Precision）.....39
圖表 12 Beam Search 在 Bigram, Trigram 的 F1 Score 比較.....41
圖表 13 Beam Search 在 Bigram, Trigram 的 Recall/Precision 比較.....41
圖表 14 HMM 與 Beam Search 的 F1-score 比較.....42
圖表 15 Bigram 在加入重排序前後的 Precision 與 Recall 比較.....43

doi:10.6342/NTU201701112

第 1 章　緒論

第 1 節　研究動機

拼字檢查在自然語言處理是個常見且重要的任務，主要原因是語言寫作在許多地方皆會使用到而且容易有錯，例如書信往來，日常的報紙，書籍的編寫。目前報章雜誌每天需要產出大量的文字，若是能透過此系統建立初步的錯字排除一些常見的拼字錯誤，可讓編輯者更能將時間花在更重要的潤飾、編輯上，提升工作上效率與文章品質。中文是目前世界常用的語言之一，但因中文的特性使得其語言相關的處理比較複雜，這篇論文則主要是針對中文處理的特性探討中文文章錯字的偵測及改正。

英文的文書處理器，經常會配有簡單的拼字改錯及文法檢查，中文在這方面卻一直依賴人工的批改，這原因可能來自於中文書寫方式有別於英文，第一中文的句子組成不像英文是透過空白做為詞與詞之間的區分，所以當閱讀文章時需要自行斷詞來取得句子的原意為何，第二中文為字構成詞，而相較於英文的 26 個字母來說，中文的常用字卻高達 5400 字，字的不同搭配形成不同的詞，不論是初學中文的外國人，甚至是專業的中文編輯人員，都曾遇到相似音形狀相似字的選擇問題，例如污染與汚染這兩個詞，這兩個字（污，汚）雖具有相同的音節音調（ㄨ），且形狀非常的相近，但後者才是正確的字詞。上述的情況，在我們分析聯合報的改稿記錄中，即使記者們也有極大的比率會混淆打錯字，是故改錯字技術可以幫助專家及一般人，應用的範圍非常廣泛。

近幾年來，關於中文改錯字論文不斷的增加，在這其中的任務，主要是要偵測句子中的錯字，並選取適合的字更換。改錯字系統最簡單的架構如下，第一是找到錯字，第二是換上正確的字，第三是確認換上的字是合理的，每個環節的處理都會深深影響到最後改正的結果，所以幾乎相關的研究都環繞在這幾個環節。

doi:10.6342/NTU201701112

舉一個含錯字的句子〞金天是陰天“，其中的“金”是我們的錯字，應該修改成“今”，我們首先需要有一個錯字更正集來得知當“金”被假設成潛在的錯字時，會有哪些候選的正確字來更換，例如音相似於“金”的可能有「今斤津...等等」，形狀相似的可能有「全淦...等等」。當我們更換上去理想的候選字（“今”）時，若使得更換後的句子（“今天是陰天”）相較於原先的句子（“金天是陰天”）更為通順。如此我們就可以認定原句子中的“金“很有可能是錯字，詳細如圖表 1。上述是簡單的介紹，此篇論文的研究重點就集中在錯字更正集上。

圖表 1 改錯字

第 2 節　研究方向

不論是在偵測錯字或者是改正錯字上，錯字集都扮演很重要的角色，但因中文改錯的資料沒有像英文來的豐富，直接運用中文錯字的勘誤表做為錯字集，可能導致在進行字的更換中，會因為正確字未收納到錯字更正集，使得錯字沒有正確字作為參考，因此而無法成功的偵測到句子的錯字；但若是我們直接從相似音相似字作為錯字集的話，會因中文的輸入法千奇百種（如表格 1），有些是透過拼音，像是注音符號，是將字的發音拆成數個音節做為輸入，有些是拼字，像是

doi:10.6342/NTU201701112

倉頡，屬於字形狀的描繪，除了上述之外，還有些是來自於使用者本身的句子錯誤，綜合以上的錯字類型，若將該錯字所有類型的候選正確字全部加入錯字集，會使得系統在過多的候選字中，嚴重的拖累系統的改正效率。除了效率之外，錯字中更有許多的候選字屬於極少見的錯誤，實務上是不需要也納入錯字集中。錯字集中正確候選字的選擇，即是我們這篇論文所要探討的。我們試圖建立一個中文錯字集的系統，抓取字與字之間的關係來塑造我們的錯字集的模型。抽取許多字與字之間的特徵，做為我們建立錯字集模型的訓練資料，再來透過機器學習的方法，試圖做出可自動調節生成錯字機率的模型。

表格 1 繁體中文的常用輸入法

注音輸入法	倉頡輸入法	大易輸入法	行列輸入法
嘸蝦米輸入法	自然輸入法	其他源自香港	其他源自大陸

除了錯字集外，當我們更換可能的正確字後，要如何確定修改後的句子是成功的修正，語言模型則扮演這關鍵性的角色，透過修正前後的語言模型分數及排序決定正確的句子為何。除了中研院的平衡語料庫的資料外，我們此次的研究方向主要鎖定在專業的編寫者，所以我們也嘗試導入聯合報提供的語料庫作為我們語言模型的基礎。以下是截取自聯合報所提供的報紙改稿紀錄（其中的錯誤“污染”應改成“汙染”）。

*國際研究團隊日前在學術刊物《柳葉刀・神經學》發表報告，二○一三年全球約三分之一因中風所導致壽命的減短，與空氣**污**汙染相關，空**污**汙成為中風的重要風險因素。*

早前的中文改錯字研究主要的資料都來自於學生的作文或者是非中文母語所寫的文章，再由中文老師抓出錯字，而做成的錯字系統的訓練資料。此篇論文主要是針對專業編寫者（例如：記者，作家等等）所做的改錯字系統，利用聯合報提供的近幾個月的報紙改稿記錄，作為我們建立系統的訓練資料。

doi:10.6342/NTU201701112

第 3 節　章節概要

本論文接下來的結構說明如下，我們在下一章簡短的介紹其他人在中文改錯使用的方法。第三章將從基礎的語言模型、雜訊通道模型（Noise Channel Model）到我們訓練錯字更正集時，萃取字與字之間的特徵及訓練的方法。第四章則會先從開始的訓練資料介紹起，再來介紹實驗的第一部，主要為聯合報資料的介紹及我們運用的方式，第二部份則是我們透過萃取字與字之間的特徵所建立的雜訊通道模型，導入新的錯字集模型運用在專業編寫者的錯字偵測及改正。最後附上實作後的實驗成果，並提出結論及未來的研究方向。

doi:10.6342/NTU201701112

第 2 章　文獻回顧

中文改錯字一般可分為規則型修改，或是利用統計模型來偵測修改。兩者的共同之處都在於更換句子中的字或詞，來到改正錯誤字詞的任務。[1]提到了規則修改的優勢在於若是修正前後的詞皆具有很高的機率，則利用規則修改可以修正這種情況。但該論文亦提到關於規則型修改的問題，在於該詞的前後字有可能會形成另一個詞，所以必須額外加入一些限制作為判斷。例如：一個詞為“一但”，系統可能認為該修改成“一旦”，但若是“一但”的前面為“統”，這裡的句子可能為“統一但分裂”，這樣若是直接套用規則修改則會出錯，因此需要再多考慮更多的鄰近字。

這些限制可能無法完全的包含所有的可能性，容易誤改到正確的詞彙。除此之外，[2]提到規則型配上詞性標注可以修正因語法錯誤所導致的錯字，如“的”，“得”，“地”的錯誤。這些錯誤一般可透過語句的分析來訂正，其他如“他”，“她”，“它”也是可以透過語句的主詞分析來修正。

在統計模型的修改中，基本上需要有錯字集跟語言模型。其中錯字集包含錯字的正確字，[3]提到錯字基本上有 76%是與拼音錯誤相關，46%與形狀相關，其餘有 29%不屬於以上狀況。因此基本的錯字集是由音相似與形狀相似的字所組成的，所以[4]提出了利用 Longest Common Subsequence 比較兩個字的倉頡碼，藉以加入倉頡碼相近度高的字對進入錯字集。 (1)[5 利用說文解字和四角碼擴增錯字更正集的數量。

[6]則使用機器翻譯的方法，引進雜訊通道模型，將錯字視為外來字，候選字視為目標的語言，每個錯字都有其轉換到候選字的機率。[7]進一步利用雜訊通道模型，分析了 google web 的資料後，並結合以字為基礎的語言模型來偵測修正句子的錯誤。

在統計模型偵測錯字中，若是將句子的所有字視為錯字做更換，會降低系統

doi:10.6342/NTU201701112

的效率，所以[8]中採用兩大限制來加速系統，第一是盡量不去更動過於常見的字，第二則是忽略極其罕見的字，最後則利用 SRILM 內建的語言模型計算來加速計分。[9]將斷詞出來的單獨字視為可能的錯字，[10]則先利用 word2vector 與詞性標注的特徵訓練出能偵測錯字的模型，藉此來優化系統的效率。

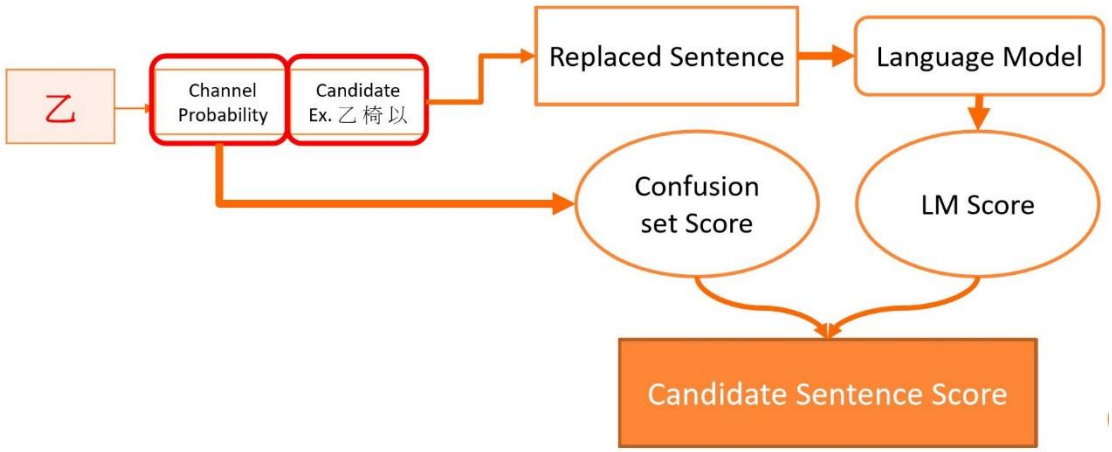
在架構上，[11]提出利用 hmm 在中文改錯字中可做為篩選正確候選字的重要關鍵，但因為最佳路徑有可能不是正確解答，所以他改成全路徑的尋找，為了效率的優化，額外對每個字的候選分支做修剪，最後取出前幾名的路徑（候選句子）。第二部分則將這些候選句子取出各種特徵值，再依向量支持訓練機中分類器的信心指數重新排序，選擇分數最高的兩句，最後再從這兩句與原先的句子挑出系統修正後的句子。[10]則是透過句子中詞的語言模型，比較修改前後的分數，再選擇何者為最後的句子。

doi:10.6342/NTU201701112

第 3 章　研究方法

以下的研究方法，先從我們中文改錯字系統的結構開始，並進而提到我們針對自動生成錯字更正集的做法。

如同緒論所說，中文改錯系統需要有錯字更正集及語言模型。此篇論文的基本架構如圖表 2，更換句子中的每一個字，錯字的每個更正字都有其轉換的機率，並將更換後的句子配合語言模型的評分，透過字與字之間的轉換機率，及句子更換前後的分數，兩者搭配來判斷語句的通順度。



圖表 2 中文改錯字系統的架構

這篇是假設句子中的每個字皆為錯字，主要是因為被認為是錯字的類型，有一大部份是詞裡的錯字，因這個而形成的改錯字方法是利用斷詞的方式，因詞裡的錯字會導致該詞被斷詞系統切成零碎的字元，再將其零碎的字依據錯字集的候選正確字去更換，並比較字更換後的語言模型分數來決定更換前的字是否有可能是錯字。

因上述的方式，取決於斷詞能否正確的切出可能的錯字，若是錯字與鄰近的詞合成其他可能的詞，會造成錯字無法偵測，或是詞沒有被斷詞系統正確的框出來，可能會誤判正確的字為錯字，這都是因斷詞不當所導致的可能問題。

因上述的原因，此篇論文在錯字的更換中，沒有特別去篩選句子中的錯字，

doi:10.6342/NTU201701112

而是預設句子中的每個字都具有錯字的可能性，所以是透過以字為主的語言模型做為基本的評分，判斷改正前後的句子。實驗中將透過隱藏馬可夫模型、集束搜尋與重排序方法來比較彼此的優劣。

以下的研究方法，會先從基本背景知識帶入，而後分別描述 HMM、Beam Search 以及 Re-ranking 的架構細節。

第 1 節 N 元語言模型 (N-gram Language Model)

藉由統計模型，計算句子的順暢度，比較新句子與原句子之間的分數依據，機率高的句子代表比較常被用在該語言上。

給定一個句子 (S) 含 n-1 個字詞，其詞序列為 $w_1, w_2, \dots, w_{n-1}$，若下一個字為 w_n 時，其機率如 (1)：

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^{n} P(w_i | w_1, w_2, \dots, w_{k-1}) \quad (1)$$

由於上式計算過於複雜，所以這邊會導入馬可夫假設，假設每一個詞的出現只會與前 m-1 個詞有關係，並藉由限制 m 的大小，以優化計算的複雜度，新的機率式如下 (2)：

$$P(w_1, w_2, \dots, w_n) = \prod_{i=1}^{n} P(w_i | w_{i-(m-1)}, w_{i-1}) \quad (2)$$

$$\text{其中 } P(w_i | w_{i-(m-1)}, \dots, w_{i-1}) = \frac{count(w_{i-(m-1)}, \dots, w_{i-1}, w_i)}{count(w_{i-(m-1)}, \dots, w_{i-1})}$$

一般 m 的大小會限制在 2, 3，主要是因為過長的語句會使 N 元語言模型失去準確度，另外也因為錯字主要是藉由鄰近字的資訊來得知正確性，是故將 m 限制在比較小的範圍，以下是 2 元及 3 元語言模型的公式 (3), (4)：

$$\text{Bigram: } P(S = w_1, w_2, w_3 \dots w_n) = P(w_1) \prod_{i=2}^{n} P(w_i | w_{i-1}) \quad (3)$$

$$\text{Trigram: } P(S = w_1, w_2, w_3 \dots w_n) = P(w_1) P(w_2 | w_1) \prod_{i=2}^{n} P(w_i | w_{i-2}, w_{i-1}) \quad (4)$$

N 元語言模型會因訓練資料的不足，無法包含所有可能的字串組，所以會透過一些平滑化技術，來解決資料稀疏的問題。關於語言模型的訓練，這邊採用的是 SRILM 工具包，利用內建的指令統計文本中的字／詞計數，並再利用內建的

doi:10.6342/NTU201701112

指令做出文本的語言模型。

第 2 節 雜訊通道模型 (Noisy Channel Model, NCM)

原先的用途是在機器翻譯上面，藉由平行語料庫作為訓練資料，來獲得句子翻譯成另一個句子的模型。平行語料庫以句子為主，句子分別由兩種語言寫成，可藉由統計來得知某個句子翻譯成另一句子的機率為何。在我們中文改錯字中，先前有提到的錯字更正集，只包含了錯字的更正字，若是導入雜訊通道模型，可將錯字更正集的重要性更加的擴大，而不只單純錯字的更正字提供，而是將每組錯字與更正字間建立各緊密的配合。

在中文改錯的雜訊通模型應用中，是將錯字視為待翻譯的字，正確的字為欲翻譯的字，將錯字與所有更正字做配對，其中更正字包含了不變的情況，代表了此字不是錯字。在已知寫錯字的機率遠小於正確的字，這個可能性在字與字之間的轉換應該會是最高。下方是中文改錯的雜訊通道模型的方程式，Origin 為原先的字，Replace 則為更換後的字，分別統計原先的字出現在文本的次數，及統計原先的字被轉換成其他字的次數，其中包含了原先的字轉換成原先的字。

$$\mathrm{NCM}(\mathrm{origin}, \mathrm{replace}) = \frac{Count(origin \rightarrow replace)}{Count(origin)} \quad (5)$$

假設在某個平行語料庫（具有錯字修正的文本）中，以“金”做為例子，屬於“金”的更換字可能有“金淦針斤今巾…”，首先我們會先統計在文本中“金”的出現次數，再分別計算在各轉換字對從“金”被改成其他字的次數，最後再去統整各字對的機率。金與其他的更正字的資訊如表格 2。因為句子的字是正確的機率遠高於錯字的可能，所以從表中可以預測（金，金）的轉換比率占最高，第二則是（金，今），第三則是（金，斤）。

doi:10.6342/NTU201701112

表格 2 「金」的雜訊通道模型在錯字更正集

原先的字	修改後	次數	NCM	NCM (log)
金	金	9933	0.9933	-0.02919
	淦	5	0.0005	-3.30102
	斤	10	0.0010	-3.0
	今	50	0.0050	-2.30102
	巾	2	0.0002	-3.6989

但因雜訊通道模型所需要的平行語料庫，要有足夠的改錯字資料才能做成更準確的模型轉換機率，若是資料不夠，很容易因訓練文本中不足的字對，而使得句子中的錯字沒有正確的更正字，或者是因文本涵蓋的主題關係，使轉換機率有所偏差。所以這篇是想藉由已知勘誤表中的字對，並萃取字與字之間的關係特徵值，透過 SVM 找到可加入錯字更正集的潛在字對。在此章節的第 5 節會更詳細的描述此方法。

第 3 節 隱藏式馬可夫模型架構 (Hidden Markov Model, HMM)

HMM 常被運用在語音辨識及自然語言處理，這邊主要是利用 HMM 中的 Viterbi 演算法做最佳路徑的搜尋，首先將句子的每一個字視為非連續的序列，每個字的更正字則視為該字的狀態矩陣，原先的字與更正字的雜訊通道模型機率則視為觀察矩陣。因此，序列之間的轉移即是透過語言模型的分數計算，再加上雜訊通道模型的機率做為觀察項的分數，這樣的架構即是我們針對 HMM 應用在中文改錯上的修改，更詳細的說明如下。

首先將句子中的字視為非連續的序列，並依錯字更正集的查詢，生成對應序

doi:10.6342/NTU201701112

列中各個字的可能更正集。初始序列的狀態機率直接使用第一個字在錯字更正集中的各更正字的雜訊通道模型機率。

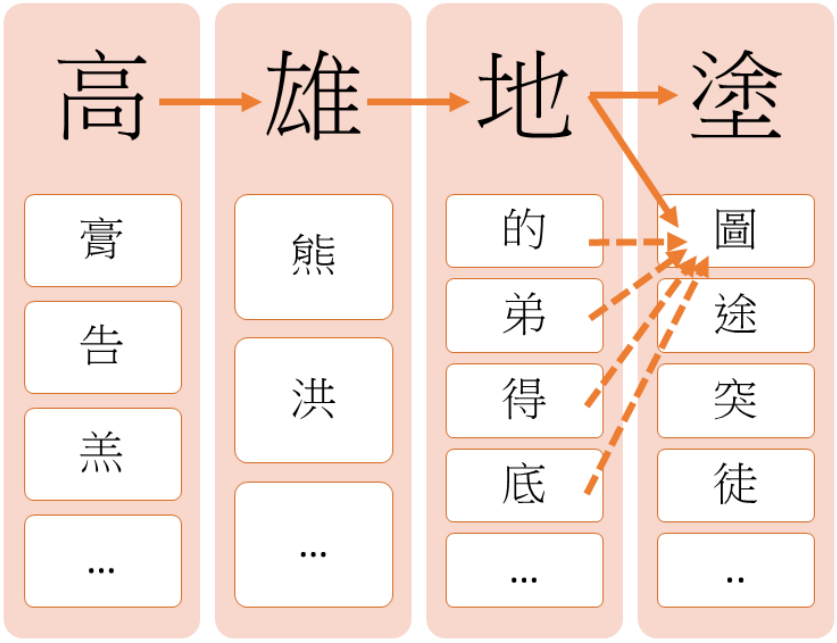
令一個句子為$q_1, q_2, \ldots, q_T$的序列，q_1代表目前處理句子中的第一個字。假設目前序列到q_t，q_t有 n 個更正字（包含q_t），令其為 $c_{t1}, c_{t2}, .., c_{tn}$，$c_{t1}$為原先的字（$q_t$），令 $\delta_t(k)$ 為在序列 t 時第 k 個更正字中的最高機率如方程式 (6)。要計算$\delta_t(k)$則需要先計算從序列 t-1 時的每個更正字過來的路徑，再從中挑選最高的分數做為數值，如方程式 (7)。而分數的計如方程式 (8)，計算原先的字與更換的字之間的雜訊通道模型機率，再計算到目前的最佳路徑與更換字的語言模型分數，最後透過一個權重總和這兩個分數。這邊使用的權重是為了調整兩種分數的重要性。

$$\delta_t(k) = \max(P[q_1, q_2, \ldots, q_t, q_t = c_{tk}]) \quad (6)$$

$$\delta_t(k) = \max_{1 \le i \le n} [\delta_{t-1}(c_{(t-1)i}) \alpha_{c_{(t-1)i}}(c_{tk})] \quad (7)$$

$$\alpha_{c_{(t-1)i}}(c_{tk}) = w * NCM(c_{t1}, c_{tk}) + (1-w) * LM(c_{(t-1)i}, c_{tk}), Bigram \quad (8)$$

依序進行序列的路徑搜尋直到完成最後一個序列，再從最後一個序列挑出最高機率的路徑，並回推找到這條路徑的序列，這條序列即為我們修正後的句子。



doi:10.6342/NTU201701112

圖表 3　HMM 在中文改錯字

假設我們正在處理的句子為“高雄地塗”，如圖表 3，目前處理到最後一個字“塗”，我們將生成“塗”的錯字正確集為“塗圖途突徒”，假設目前處理到“圖”這個字，首先會計算整句的雜訊通道模型及語言模型的機率分數（log 機率），並取適當的比率結合兩個的分數，詳細的數據如表格 3，從表中可得知分數最高為“高雄地圖”，這時我們即知道從第三個字到第四個字的候選字“圖”，最佳的路徑是由“地”過來的。

表格 3 HMM 的分數計算

句子	NCM	LM	Weighted Sum= 0.3*NCM+0.7*LM
高雄地圖	-5.3010	-8.8027	-15.9755
高雄的圖	-10.602	-8.2189	-17.0869
高雄弟圖	-10.602	-15.2638	-24.1664
高雄得圖	-10.602	-10.1520	-19.8321
高雄底圖	-10.602	-14.6380	-22.4126

第 4 節　集束搜尋 (Beam Search)

主要的應用是在路徑的搜尋，當可能的路徑過多，在期待的時間內無法找到最佳的路徑，這時為了減少搜尋所占用的空間和時間，在進行每一層的深度擴展時，系統會針對已知較差的路徑進行剪除，只保留前幾名的優良路徑，已經被砍除的路徑，其路徑在接下來的路徑擴展皆不會用到，重覆以上的步驟，直到遇到終點。這樣的做法可減少空間的使用率，當在路徑極多的時候，透過限縮每一層最後的路徑可大幅降少時間的消耗。主要的缺點在於在砍除的過程中，可能會把潛在的最佳路徑給砍除。

在我們的系統上，我們視句子的一個字為一層深度，第一個字即為我們深度

doi:10.6342/NTU201701112

一的點，終點為句子的尾端，即句子的最後一個字。我們會先從深度一開始生成起始的點，即透過我們的錯字更正集生成第一個字的可能更正集，然後拿更正字去替換句子中對應的中文字，得到新的句子後再去計算新句子的以字為基礎的語言模型分數及雜訊通道模型的分數，最後將得到的分數及新句子扔到我們的堆疊中。待完成深度一第一個字所有可能更正字的分數計算後，我們會先依堆疊內的分數從大到小做排序，再依起初的堆疊設定大小，將超過堆疊的尾端的元素給刪除，這樣在進行下一個深度的探索時，因路徑點的減少，我們的效率得以提升。

依上方的步驟，重覆句子的每一個字直到完成最後一個字。這時我們將會得到許多修改完的句子及分數在我們的堆疊裡，從結果中挑出最佳的句子做為我們系統的輸出。

第 5 節　重排序 (Re-ranking)

上述的 HMM 及 Beam Search 都是利用字的語言模型評斷句子的順暢度，我們可以利用以詞為基礎的語言模型來重新計分。在 HMM 中架構中的每個點不再只有採計最佳的路徑，而是保留前幾名的路徑。在句子結束後將每個字的最終路徑輸出到我們的堆疊中，即許多新生成的句子，留待之後的重排序。在 Beam Search 中我們直接使用最後的堆疊做重排序。

在完成堆疊的生成階段後，我們會拿堆疊的元素以不同的方式重新計算。在重新排序堆疊的時候，我們會改用詞的語言模型來做計算，主要的原因在於我們在進行堆疊生成前的時候，只有一個原始句子做為我們的輸入，為了避免有些不常見的錯字沒有被更換到，所以我們採取將句子的全部字皆視為可能的錯誤進行更換，在這個過程中，藉由語言模型及雜訊通道模型，會找到數個最有可能的句子修正，其中原始句子錯誤的詞，應該也會在這個過程被修正，再經過斷詞，修正後的句子應該會比原先的句子斷詞的整體詞數更少。再來，利用多種以上的分數計算方式做為我們句子修正的評估，可有助於我們系統的準確度及可靠度上升

doi:10.6342/NTU201701112

不少。

系統會將堆疊內的元素一個一個取出，將元素中的句子經過斷詞後，計算以詞為基礎的語言模型及雜訊通道模的分數，我們將知道目前處理句子的集束搜尋及重新排序階段所獲得的分數，最後以這兩階段總和最高分的句子做為我們最後的改正句子。

第 6 節 錯字更正集

在中文改錯字中，系統需要將錯字更換成正確的字，每個字在未篩選前都具有錯字的可能性，這代表錯字更正集的涵蓋率極為重要。若系統將所有的中文字做為錯字的更換字，會因常用中文就已經高達 5000 多字，這個系統的運算成本會過多，而且有些字之間，極少被誤用。若我們只採用已知的錯字勘誤表，會因更正字過少，缺乏正確的更正字做為新句子，系統永遠無法正確修改錯字。所以我們應該要將錯字的更換字去蕪存菁，同時也保留常見的更正字，減少系統運算的成本，使系統能在更短的時間內找到正確的字。

我們已知的錯字勘誤表數量過少，所以我們介紹新的方法，利用字與字之的各種相關特徵值（如：字形、字音、字義...等等），透過單純貝氏分類器先做初步的分類，再利用向量支持機（SVM）訓練出的分類器，分辨字對是否可以加入我們的錯字更正集。以下將介紹各特徵值的取法，跟更詳細的模型訓練。

第 1 項 聲音相近

注音輸入法是目前中文繁體字常見的輸入方式，所以在錯字的類型中這類的錯誤屬於大宗。特別是寫者有時會在音節跟音調上，因過於相似而誤用成其他的字，像是注音裡的 ㄋ ㄌ 時常會被搞混，例如：“籬（ㄌㄧˊ）笆“，與”泥（ㄋㄧˊ）巴“，這兩個詞的發音極為相似，在中文輸入的過程中，很容易弄錯。

這邊設計了兩種層次的篩選，第一是音調的差異，第二則是音節相近的萃取，

doi:10.6342/NTU201701112

詳細的分類表及例子如表格 4。我們將兩個字的注音比較，從相同音、同音異調、近音同調、近音異調，依序給予這組字對在注音的分數。

表格 4 相似音的規則

類型		內容	例子
音調		一聲 (-)、二聲 (ˊ)、三聲 (ˇ)、四聲 (ˋ)、輕聲 (˙)	青 ㄑㄧㄥ 情 ㄑㄧㄥˊ 請 ㄑㄧㄥˇ 慶 ㄑㄧㄥˋ
音節	聲母	ㄈ／ㄏ ㄋ／ㄌ ㄓ／ㄕ ㄔ／ㄘ	飛機／灰雞 泥巴／籬笆 知識／姿勢 拆／猜
	韻母	ㄧ／ㄩ	夜光／月光
	介音	ㄢ／ㄤ ㄜ／ㄦ ㄣ／ㄥ	麵攤／麵湯 惡人／二人 出身／出生

第 2 項 形狀相近

相較於拼音的注音輸入法，也有不少人是利用拼字的倉頡碼做為中文輸入的方式，所以我們也將倉頡碼的相似程度做為字對的特徵值。這邊是參考 (2)透過 Longest Common Sequence (LCS) 比較兩個字的倉頡碼，倉頡碼相近度高的字對可納入錯字更正集。因字的倉頡碼由 1~5 個字元所組成，所以在比較 LCS 的時候，會參考彼此倉頡碼的長短。假設目前要比較的字對為（A,B）兩字，參考彼此的倉頡碼及 LCS 長度，條列如下的表格 5 為可列入相似字的依據。

舉“好“(A) 的倉頡碼為 VND，比較“子”(B) 的倉頡碼 ND，滿足 A 的倉頡碼

doi:10.6342/NTU201701112

長度為 3，B 的倉頡碼長度為 2，其 LCS 長度為 2，則我們認定（好，子）具有形狀相似的關係。

表格 5 倉頡碼的相似度規則

A 字的倉頡碼長度	B 字的倉頡碼長度	LCS(A,B)長度
1	1	1
2	2	1
	3	2
3	2,3,4	2
4	3,4,5	3
5	4	4

第二部分則是利用整理後具有同偏旁的字串，若兩字具有同偏旁，則將其視為形狀相近的字。

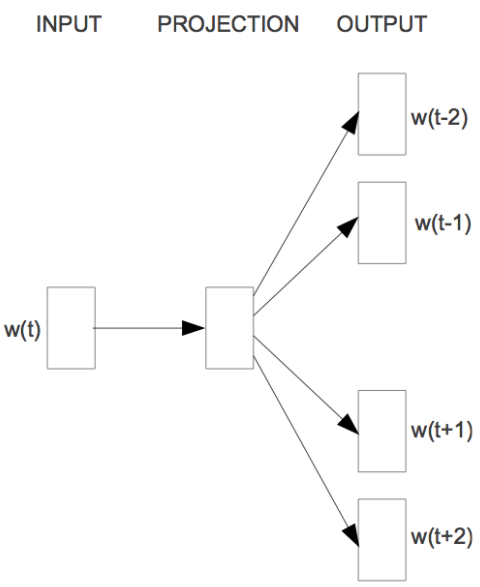
舉“采”做為基礎的偏旁，則具有“采“偏旁的字串如下：

采，倸，埰，採，啋，婇，棌，睬，綵，踩，彩，菜，寀

第 3 項 Fast Text

為了增加字與字之間的關係，擴增字對的特徵值，我們這邊使用了 Facebook 提供的 fastText API，這個 API 利用 word2vec 計算，將每個字視為一個向量，以 Skip-gram 的結構（圖表 4），透過深度學習的訓練，得到字的向量。透過比較兩個字向量的餘弦相似度，我們可以得知兩字的使用相似度。

doi:10.6342/NTU201701112



圖表 4　Skip-gram 的基本架構[1]

第 4 項 錯字更正集的生成

我們採計了字與字之間的 9 種特徵值做為模型訓練用資料，取（世／氏）這組字對做為我們的範例，如表格 6。我們的目的是要將未知的更正字加入到我們新的錯字更正集中，我們視已知的勘誤表為正樣本，目標是分類未知的樣本，即其他不在勘誤表的字對，找到適合的字對加入到我們的錯字更正集中。

表格 6 （世／氏）的字對特徵值

特徵值	世／氏
Unihan 的相似音	4 同音同調
Unihan 的相似形	0 無
Unihan 的倉頡碼相似度	0 無
Unihan 提供的更正字字頻	1
SIGHAN 提供的相似音	4 同音同調
SIGHAN 提供的相似形	0 無

[1] Image from: http://sebastianruder.com/word-embeddings-1/

doi:10.6342/NTU201701112

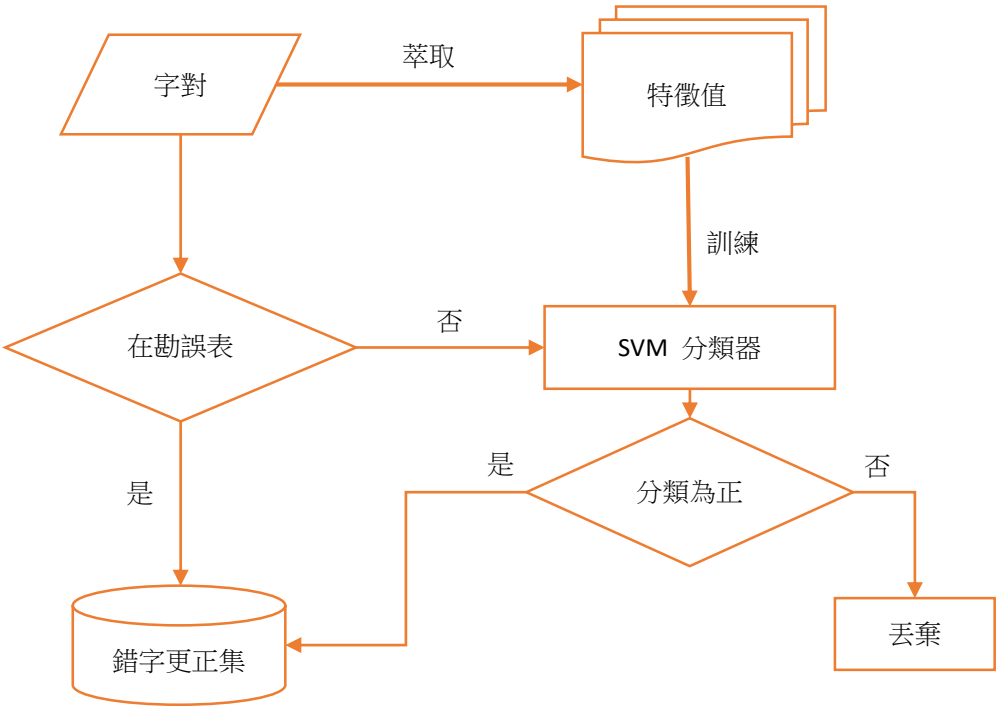
一元的語言模型分數	-3.10 (世) / -3.77 (氏)
FastText 餘弦相似度	0.4102

我們這邊將從勘誤表中的字對視為正樣本，未被收納進勘誤表的字對則視為未標籤的範例，分別從正樣本及未知樣本取出部分的字對，做為 SVM 的訓練資料，將獲得的 SVM 分類器分類在所有的字對中，決定是否要將該字對加入到我們的錯字更正集中，流程圖如圖表 5。

第一步，我們在取字對的時候是針對一萬多個中文字做萃取，但因中文字包含異體字將近一萬多個字，其中有許多的中文字不是我們常用的字，所以我們首先將我們的更正字限縮在常用的 5400 字以內，減少更正字數量的同時，也可以增進系統的效率。

第二步，則是將上個步驟獲得的樣本及其對應的標籤透過 SVM 來訓練出的模型，這個模型可以拿來分辨字對是否可加入錯字更正集。

最後，則是透過上個步驟的模型，判斷每組字對是否可加入到我們的錯字更正集中。



圖表 5 錯字更正集自動擴增的流程圖

doi:10.6342/NTU201701112

第 5 項 雜訊通道模型的機率

得到的錯字更正集後，則導入我們設定的機率，我們使用的是全域的錯字機率 0.0000487≒0.00005，這個機率是統計聯合報的文字所得到的機率。將錯字轉換成本身，即正確字，設為 0.99995，錯字轉換成其他字，設為 0.00005。最後標準化每個錯字的更正字，使其總和等於一。舉“世”為例子，“世”有“世是氏事“四個更正字，其雜訊通道模型機率如表格 7。

表格 7 “世”的雜訊通道模型

〈錯字，更正字〉	機率	標準化機率 (log)
〈世，世〉	0.99995	-6.5142E-05
〈世，是〉	0.00005	-4.301073423
〈世，氏〉	0.00005	-4.301073423
〈世，事〉	0.00005	-4.301073423

doi:10.6342/NTU201701112

第 4 章　實驗結果

第 1 節 實驗資料

第 1 項 中研院平衡語料庫

由中研院所編寫的文本，句子中的詞以空白分開，並包含了詞性的資訊，總共 13 萬句，112 萬個詞，175 萬個字，

範例句子：出國|Nv 期間|Na 院務|Na 由|P 羅|Nb 副院長|Na 代行|VC 。|。

第 2 項 Unihan 資料

包含了中文字的許多屬性資料，如：注音、倉頡碼、部首...等等。詳細的屬性資料會在接下來的錯字集訓練介紹。

第 3 項 SIGHAN 2013 資料

包含了相似音及相似字的資料。

第 4 項 錯字勘誤表

1. 教育部錯別字表
2. 東東錯別字
3. 常見錯別字一覽表

第 5 項 聯合報改稿記錄

來自於聯合新聞網及聯合報體系的每日報導的改稿紀錄，資料日期從 2016/06/19 到 2017/01/12，共 107 萬檔案，每份檔案平均 400 個字，共 4.7909

doi:10.6342/NTU201701112

億個字。原始的文章會先由記者撰寫，再交由審稿編輯去確認修改潤飾，這整個修改過程在聯合報的系統都會全部紀錄下來。詳細的內容會在下一節介紹。

第 2 節 聯合報標準測試集建立 (UDN)

已知的中文改錯標準測試資料，主要都來自於國中小學的作文改錯或者是學習中文的外國人文章改錯，但使用中文寫作的人不只這些人，更有些職業每日以撰寫中文文章為主，如記者每日的報導撰寫或者是作家的專業文章寫作，他們的錯字類型相較於目前已知的改錯字資料，應該會有非常不一樣的差異性質。透過聯合報提供的改稿紀錄，我們將有這個機會了解專業寫作者的錯字類型，接下來會先從改稿紀錄的格式介紹，經整理後，對於其中錯字的類型簡單的描述，最後從中挑選適合測試為專業寫作者的標準測試資料。

第 1 項原始資料格式

聯合報提供的資料是由 HTML 所編寫成，如下所示：

原文
【記者葉君遠／專訪】梁赫群 45 歲中年得子，對兒子疼愛有加，其實，老婆懷孕時，他就是個會凌晨 4 點為了老婆肚子餓，半夜起床跑去買消夜的男人。有了兒子，更一手包~~辦~~辦泡牛奶、包尿布等~~鎖瑣~~事，…
原始檔
<P>【記者葉君遠／專訪】梁赫群 45 歲中年得子，對兒子疼愛有加，其實，老婆懷孕時，他就是個會凌晨 4 點為了老婆肚子餓，半夜起床跑去買消夜的男人。有了兒子，更一手包 <FONT title=章淑曼刪除, class=3 style="BORDER-TOP-COLOR: ; BORDER-BOTTOM-COLOR: ; TEXT-DECORATION: line-through; BORDER-RIGHT-COLOR: ; BORDER-LEFT-COLOR: " color=#555588>辦

doi:10.6342/NTU201701112

</FONT>

<FONT title=章淑曼新增, class=1 style="BORDER-TOP-COLOR: ; BORDER-BOTTOM-COLOR: ; BORDER-RIGHT-COLOR: ; BORDER-LEFT-COLOR: " color=#265e8a>辦</FONT>泡牛奶、包尿布等

<FONT title=章淑曼刪除, class=3 style="BORDER-TOP-CLOR: ; BORDER-BOTTOM-COLOR: ; TEXT-DECORATION: line-through; BORDER-RIGHT-COLOR: ; BORDER-LEFT-COLOR: " color=#555588>鎖</FONT>

<FONT title=章淑曼新增, class=1 style="BORDER-TOP-COLOR: ; BORDER-BOTTOM-COLOR: ; BORDER-RIGHT-COLOR: ; BORDER-LEFT-COLOR: " color=#265e8a>瑣</FONT>事，

</P>

每一個段落由 HTML 的 P 標籤所組成。若是段落內部有修改，則會用 FONT 標籤夾住修改的字串，標籤的屬性如表格 8。

表格 8　FONT 的標籤屬性

範例		
<FONT **title**=章淑曼刪除, **class**=3 style="BORDER-TOP-COLOR: ; BORDER-BOTTOM-COLOR: ; TEXT-DECORATION: line-through; BORDER-RIGHT-COLOR: ; BORDER-LEFT-COLOR: " color=#555588>**辦**</FONT>		
屬性名稱	**屬性含意**	**範例**
Title	處理人的名字及其動作	章淑曼刪除
Class	以數字表示操作的動作 1/新增，3/刪除 （不會同時有新增刪除的動	3

doi:10.6342/NTU201701112

	作）	
內文	欲處理的字串	辦
其他屬性	CSS 格式	略

第2項 錯字萃取方式

從上節的資訊中得知，修改的 FONT 標籤屬性只有"新增"、"刪除"，若是相鄰在一起的屬性為”新增””刪除”，則屬於修改，但在改稿紀錄中，不會只有侷限在一個字的修改，會出現數個字以上的修改，甚至會有整段被刪減或刪除的情況發生。因為我們主要專注在錯字類型的了解，因此以下有 XX 個條件是我們在截取聯合報的錯字所使用的：

1. 在萃取的過程中，為了避免句子過長，所以依標點符號（，。！？）重新斷句。
2. 設定 FONT 標籤為新增的字是 1，FONT 標籤為刪除的字是 3，其餘的則設定為 0，文章的每一個字都會有自己的數字。
3. 錯字為標籤 3 的字，更正字則為標籤 1 的字，兩個標籤必須鄰近。
4. 標籤的內文長度必須要等於 1，才代表是字的修改。
5. 句子中有錯字的數字編號會有 0130 或 0310 的編碼存在

滿足以上條件才會被我們認定為句子有錯字。實例如表格 9 所示，”辦”為刪除 (3)，右邊為新增字”辦” (1)，則辦辦的兩邊必須是沒有修改過的內容 (0)，即沒有透過 FONT 標籤所夾住的內文，滿足這樣的條件才會被認定句子具有錯字。

表格 9 錯字的取法

字串		更	一	手	辦	辦	泡	牛	奶	，	
編號		0	0	0	3	1	0	0	0	0	

最後從資料中，總共截取出 54624 個含有錯字的句子，共 58541 個錯字。

doi:10.6342/NTU201701112

平均長度為 19.28 個字。

第 3 項 錯字類型

待依第 2 項的條件取出聯合報中的所有包含錯字的句子，經整理後，前 10 名常見的錯字如表格 10。

表格 10 聯合報的前 10 常見修正

排名	錯字	更正字	修正次數	在文本中錯字出現次數	在文本中更正字出現次數
1	3	三	2126	417612	652617
2	1	一	1999	529485	3083003
3	4	四	1356	283723	291276
4	劃	畫	1346	102782	299320
5	5	五	1295	285854	310788
6	週	周	1161	38114	373956
7	2	二	1077	448784	460420
8	6	六	1020	241726	200996
9	8	八	946	244123	160428
10	7	七	926	263815	146728

從表格 10 得知，在聯合報最常見的修正是數字國字的修正，這與報紙的排版有所關係，若是標題或內文為直式書寫，此時會將數字轉成國字，通常在報紙中，頭版為橫式標題直式內文，其他版皆為橫式書寫，如圖表 6 及圖表 7。

doi:10.6342/NTU201701112

綠→願將8年8900億對半砍 藍→堅持刪除部分軌道建設

前瞻協商未果 立院今對決

【記者丘采薇／台北報導】立法院臨時會昨天處理「前瞻基礎建設特別條例」草案，朝野黨團協商一整天，今天將直接進入院會對決。昨天民進黨團一度釋出善意，主動提議將前瞻計畫從原本的八年八千九百億元縮減成四年四千兩百億元，但因國民黨團堅持刪除部分軌道建設而未果，朝野黨團內廣鴿派相互較勁，也使得今天三讀程序充滿變數。

綠鷹派主戰 黨團不放棄協商

據了解，民進黨團內部對何時通過前瞻條例有不同看法，鷹派認為應直接與國民黨團開戰，但黨團幹部認為不應破壞朝野氣氛，尤其接下來還有第二、三次臨時會，若現在就徹底激怒國民黨團，恐怕導致戰線延長，不利行政院長林全未來到立院報告前瞻預算，黨團也不斷說服府院高層，才得到願向藍營讓步的承諾。

國民黨團對和戰也有不同意見，一方面是真正戰場應在下次臨時會預算詢答，也看準民進黨非過不可的壓力，要「以時間換取空間」，逼民進黨團讓步，因而有減半成四千億的轉圜，但國民黨也擔心綠營只是緩兵之計，不斷加碼，要逼民進黨持續讓步。

昨天朝野協商從早一直進行到深夜，國發會主委陳添枝遭國民黨及時代力量黨團輪番砲轟；下午開始針對前瞻條例的十一條條文逐一討論，但藍綠黨團總召都沒有參加。傍晚柯建銘突然現身協商現場，勸在野黨要理性看待前瞻，一旁國民黨立委要求柯建銘一起坐下來協商，但柯拒絕，直言已經為協商前瞻花了太多時間，國民黨最後總是要面對。

隨後立法院長蘇嘉全宣布將院會開會時間延長至午夜十二時，但藍綠都已開始動員，一度傳出民進黨要發動攻勢讓前瞻條例強渡關山，氣氛緊繃。但經藍綠黨團總召持續溝通，民進黨團決定釋出善意，解除動員；蘇嘉全承諾會讓在野黨暢所欲言，但國民黨立委仍堅持占領主席台，並再度夜宿議場。

藍圍剿軌道建設 綠籲理性面對

昨天朝野協商幾乎無交集，國民黨團除要求將新增的軌道建設路線刪除，也以修正動議方式提出軌道建設不可超過總預算三成，並要求將少子化、監督機制等意見納入前瞻計畫。民進黨團書記長李俊俋表示，國民黨對每件事情都十幾個意見、版本，很難有共識，他希望在野黨能理性，而非什麼項目都要納入前瞻，但又一直要求刪減預算，這樣無法做事。

面對議場主席台被國民黨占領，李俊俋也回應，民進黨已有相關評估及解決方法，民進黨團並不是沒有面對過這種場面，「我們自然會處理。」據悉，為避免立院上演全武行導致社會觀感不佳，民進黨團希望讓前瞻條例在和平氣氛下通過，黨團幹部因此不放棄與國民黨團協商，盡量尋求形成默契的可能。

相關新聞見A3

圖表 6 聯合報頭版的橫式標題直式內文[2]

台中七媽會缺一 錯過再等百年

媽變六媽 鹿港天后宮▶受邀時已來不及 中市府▶去年6月就發文 拜會2次仍被拒

台中市府主辦七媽會，但因鹿港天后宮缺席，改名百年媽祖會。圖為李興宮廟之一的李興宮媽祖廟內七媽會宣傳資料，中間為7尊媽祖百年前聚首老照片。 記者黃寅／攝影

【記者喻文玟、劉明岩／連線報導】100年前慶祝台中火車站、第2市場落成，中南部7座媽祖廟駐駕台中40天，史稱「七媽會」。台中市政府籌辦「百年媽祖會」，本想重現百年前盛況，但因鹿港天后宮缺席，「七媽」無緣再聚首。

百年前參加七媽會的媽祖廟為台中萬春宮、台中樂成宮、北港朝天宮、梧棲朝元宮、新港奉天宮、彰化南瑤宮和鹿港天后宮。百年後何以七媽變六媽，鹿港天后宮和中市府說法不同。

鹿港天后宮主委張偉東說，一開始並未收到邀請參加活動，事後才從網路得知。天后宮媽祖出門及回鑾必須要有周詳的策畫與籌備，因受邀參加「七媽會」時，「時間來不及」故不克參加，並表達祝福。

中市民政局副局長盧政達表示，去年6月萬春宮就發文邀請鹿港天后宮，並在同年8月親自拜會。天后宮去年11月回函表示不克參加；市長林佳龍、民政局長蔡世寅今年4月也親自前往邀請，廟方仍婉拒。

林佳龍昨在百年媽祖會宣傳活動中說，1917年台中車站的第2代站體「台中驛」竣工，「新富町市場」即現今第二市場也在這一年啟用，為了這兩件盛事舉辦「七媽會」，當時大家都坐火車來。重現這段百年歷史意義重大，「沒來絕對會後悔，後悔100年，下次要來，要再活到100多歲。」

因為無法如願催生七媽會重現，台中市文化局邀每年3月參與「媽祖宮廟風華」的百年宮廟共襄盛舉，共計有15座宮廟出席「百年媽祖會」。最重要的「六媽」13日至16日將駐駕台灣體育運動大學體操館，15日舉辦「媽祖遶境祈安盛會」，六媽將聯合遶境台中市區，重現百年盛況。

台中市文化局長王志誠說，百年媽祖會擴大七媽會內涵，安排一系列陣頭匯演；董事長樂團16日下午5點10分在台中公園演出「眾神護台灣」。台鐵騰雲號、BK10型蒸汽小火車活動期間也會在台中公園亮相，限時供民眾搭乘，活動訊息可上官網查詢，網址：http://www.chenyi100.com.tw/2017CentenaryMazu/。

圖表 7 聯合版全台焦點的橫式標題及內文[3]

但數字國字的轉換不在我們錯字標準測試集的考慮範圍，所以我們將從中排除所有這類型的錯字。重新整理後，前 10 名的錯字如表格 11。

[2] 聯合報 2017 年 07 月 05 日的 A1

[3] 聯合報 2017 年 07 月 05 日的 B1

doi:10.6342/NTU201701112

表格 11 聯合報的前 10 常見修正（去掉數字的轉換）

排名	錯字	更正字	修正次數	在文本中錯字出現次數	在文本中更正字出現次數
1	劃	畫	1346	102782	299320
2	週	周	1161	38114	373956
3	佈	布	894	24869	344871
4	佔	占	825	20822	108078
5	份	分	657	92373	1060568
6	污	汙	619	15626	36573
7	臺	台	567	40152	2469198
8	今	昨	552	1024066	335207
9	越	愈	356	161095	65305
10	之	的	325	679474	8443044

因報紙的特殊性，其中第 8 名的〈今, 昨〉出現的主要原因，在於報紙出刊時間的先後，同一篇文章會同時出現在新聞網及報紙上，在新聞網上的事件發生是在“今”天，但隔天在報紙上整理出版時，就會因事件的時間關係，而將“今”天修改成“昨”天。第 9 名〈越，愈〉，則是因編輯者的習慣，導致句子的修改，究整體而言，反而是“越”這個字的使用頻繁度高於“愈”這個字。

從上方兩點來看，要從聯合報中排除非典型的錯字，即要排除是因編輯者相關的修改，第一點可查看兩者之間是否有互換，若是彼此之間互換的程度過多，這可能代表兩個用法都是正確的，因情況不同，皆有所使用。在表格 12，計錄一些互換比例比較高，而且常出現的修正字對。在表中新增一個欄位”Ratio”，代表著兩個字之間的轉換比例，其計算方式如方程式 9，Ratio 值越小，代表著彼此的轉換越相近。舉〈再，在〉為例，這組字對的 Ratio.為 0，代表著兩者之間

doi:10.6342/NTU201701112

的更正次數相同。而意義上，”再”常用在再一次、重覆的意思上面[4]，”在”則用在動作正在進行或表示位置處所時間之類的[5]，這兩個字的用法需要考慮到語句上的意思才能判斷。

$$\text{Ratio} = \frac{|count(ch_1 \rightarrow ch_2) - count(ch_2 \rightarrow ch_1)|}{\max(ch_1 \rightarrow ch_2, ch_2 \rightarrow ch_1)} \quad (9)$$

表格 12 互換類型的修正

Ch1	Ch2	ch1->ch2	ch2->ch1	Ratio	在文本中 Ch1 出現次數	在文本中 Ch2 出現次數
兩	二	109	217	0.497696	588550	460420
做	作	75	182	0.587912	312483	919259
記	紀	149	74	0.503356	654501	149506
濕	溼	83	44	0.46988	18550	3018
至	到	36	75	0.52	724444	1624287
須	需	34	58	0.413793	158319	342109
連	聯	33	53	0.377358	340924	382285
在	再	40	40	0	2932239	472810
姐	姊	32	42	0.238095	29388	20025

第二點則是延伸第一點的想法，加入鄰近的字詞做為考量錯字的依據，因錯字的確定幾乎都是因為與周遭的字無法形成已知的詞所被認定的

若是字與字之間皆有互換，那我們可以搭配鄰近的字來了解進一步的情況，這樣可以更確定錯字的正確度。取出被修改字的前後一個字，並在文本查詢是否

[4] 教育部 重編國語辭典修訂本 “再” http://dict.revised.moe.edu.tw/cgi-bin/cbdic/gsweb.cgi?ccd=g4HTHh&o=e0&sec=sec1&op=v&view=0-1

[5] 教育部 重編國語辭典修訂本 “在” http://dict.revised.moe.edu.tw/cgi-bin/cbdic/gsweb.cgi?ccd=s4LQHi&o=e0&sec=sec1&op=v&view=1-1

doi:10.6342/NTU201701112

存在及在字典是否有記錄，若是有則標記為詞並掃描其在文本的出現次數。舉〈記，紀〉這組字對作為例子，如表格 13，分別取字的前後字，即 O 紀(記)／紀(記)O 的各種可能性，但因 O 紀或 O 記沒有成有效的詞，所以表格 13 列都是紀(記)與其後面一個字的情形。從中看到編號 1 的"記錄"與"紀錄"，在文本的出現次數與修正次數蠻相近的，不像"紀念"與"記念"相差很多，且出現次數過低，這代表著"記錄"與"紀錄"具有不同的用法，且兩者的使用皆滿頻繁的，像這樣的字對不會納入到我們的測試集中。

表格 13 〈紀，記〉詞的相關資訊

類別	詞 A	詞 B	詞 A 出現次數	詞 B 出現次數	詞 A→詞 B 修正次數	詞 B→詞 A 修正次數
1	記錄	紀錄	15329	68145	139	56
2	記念	紀念	93	33827	1	0
3	記載	紀載	3854	136	0	3

第 4 項測試集的建立

綜合以上的說明及條件，濾除掉這些錯字，但無法濾除掉所有這類型的錯誤，所以我們參考這些數據外，還必須做進一步的篩選。因為除了上述的類型外，有一部分字的更換是源自於編輯者的想法，所以我們針對分析出來具有可能錯字的句子，再額外去做人工篩選。最後我們將句子分成兩類，"編輯者的修改"及"錯字"。

屬於錯字類型的字，我們大概歸納一些類別，如表格 14，表中括號內為更正字。除了常見的形音錯字外，有一些是簡體字的錯字，更有些是極少見的異體字，例如〈內，內〉 後方才是正確的字，通常錯字與更正字都非常的相似，在人工偵錯的情況下，相對其他錯字比較難檢查到。

表格 14 錯字的類型

doi:10.6342/NTU201701112

編號	錯字
1	● 相似音相似形的字 空氣污(汙)染對中風影響的比重如此之高， 這是蔡英文首次出訪友邦和以總統身份(分)過境美國。 在生技、隱型(形)眼鏡等產業，
3	● 與代名詞相關 「你(妳)的秧仔好醜」是她聽到的第一句話，
4	● 與詞性相關 因而再(在)床的四周加上一圈布製的床圍，
5	● 簡體字 体(體)院的學生， 不断(斷)提及。
6	● 異體字（形狀非常相近） 涉(涉)溪下山， 反而都没(沒)事。 説(說)來說去都是材料的事，

至於編輯者的修改，除了上方提到的以外，有些修改是介紹詞的修改，例如〈之，如〉，更詳細如表格 15。

表格 15 編輯者修改的類型

編號	編輯者的修改
1	● 報紙的特殊修改（如：今／昨） 中共機關報人民日報所屬的環球時報今(昨)天發表社評， 第一及二航廈今(昨)天又漏水，
2	● 數字與國字之間的轉換 附近5(五)公里半徑內沒有一個鄉鎮，

doi:10.6342/NTU201701112

	且三(3)年期長期績效也相當突出， 市區積水嚴重達二(兩)百多處，
3	● 文白風格 華航應對於未能及時登機之(的)旅客， 其實為(是)數理和語文能力的綜合表現， 英國如(若)徹底「脫歐」將是一大損失。 被空拍機撞至(到)頭部血流滿臉，

最後我們從中 3626 有修改的句子中挑選出 1217 句含有錯字的句子，每個句子平均長度 20.48，每個句子至少有一個錯字，另外我們再加入相等數量不含錯字的正確句子，做為我們負樣本的句子，所以總共有 2434 個句子在我們的聯合報標準測試集。

第 3 節 錯字更正集的生成

我們抽取了數個資料庫做為建立我們錯字更正集的特徵，首先我們利用網路上提供的 Unihan 資料，抽取出發音及部首形狀的部份，當作我們特徵的基石，第二我們抽取了字頻的部份，做為字的常用度特徵之一，第三則抽取倉頡碼的部份，詳細可見表格 16。

表格 16 文字表

字	拼音	部首形狀	筆畫	字頻	倉頡碼
青	ㄐㄧㄥ ㄑㄧㄥ	青	8	3	手一月
倩	ㄑㄧㄢˋ ㄑㄧㄥˋ	人	10	5	人手一月
請	ㄐㄧㄥˋ ㄑㄧㄥˊ	言	15	1	卜口手一月

doi:10.6342/NTU201701112

	ㄑㄧㄥˇ				

除了 Unihan 的資料，我們還有份資料是部件組成的相關字，如表格 17 所舉“青”的例子。

表格 17 部件“青”的相關字

青	青|倩|凊|请|請|堷|掅|啨|崝|猜|清|情|婧|棈|晴|賕|賰|腈|踁|碃|睛|錆|錆|睛|靖|晴|蜻|精|綪|輤|鯖|鯖|鼰|鄑|靚|靚|靑|靈|鵹|鶄|靕|靓|靜|靜|菁|甯|箐|圊|氰

再來我們整理了 SIGHAN 2013 年提供的相似字相似音做為我們輔佐的參考，此份資料提供了常用字的相似音（同音同調、同音異調、近音同調、近音異調）、相似形狀的字及同部首同筆畫的字。詳細可見表格 18。

表格 18 倩在 SIGHAN 的相似音相似字

倩	ㄑㄧㄢˋ
同音同調	欠塹縴歉
同音異調	乾簽前遣黔愆鉛千箝遷韆鈐籤譴鉗阡嵌仟縴騫虔潛謙淺牽搴錢
近音同調	件檻劍健僭見艦漸毽諫濺踐監箭間閒薦腱建鍵餞鑑鑒澗賤薦
近音異調	箋縑兼姦韉漸殲奸囝艱柬檢簡濺減筧監間肩尖湔剪鹼瞼繭堅管緘撿揀儉煎
同部首 同筆畫數	倦倥倣倡倭俯修倫倪倨俱俳俾值俸俺倆倀倍倌倉個倔倖們倒借倘候倚
相似字	債情清晴猜睛靖精蜻鯖菁請蹟青嘖漬磧積績責

最後我們依照已知的勘誤表，將有出現的字對，舉表格 19 為例，「不齒」與「不恥」為正確詞與錯誤詞，則〈恥，齒〉這組字對則會被我們標記為正樣本，代表著恥有機會被修正成齒這個字。但〈齒，恥〉不一定會在我們的正樣本中，除非在勘誤表中，有出現齒這個字被修改成恥的情況。因此這個字對屬於單向修正，不是雙向互相修正的。

doi:10.6342/NTU201701112

表格 19 教育部錯別字表

正確的詞	錯誤的詞	正確的字	錯誤的字
不齒	不恥	齒	恥
保母	褓母	保	褓
倒楣	倒霉	楣	霉
入不敷出	入不符出	敷	符
偶爾	偶而	爾	而

對每種字對進行特徵抽取，並且過濾掉不在中文常用字的字對。總共有 13658 組正樣本，有 15.5 萬未知樣本，部分例子如表格 20。取出正樣本的一半做為正的訓練樣本，同等數量的未知樣本做為負樣本，使用 SVM 訓練分類器，其餘的樣本則作為我們測試資料準確度的測資。

表格 20 青與其他字的特徵值表

(錯字 A, 更正字 B)	(青, 情)	(青, 倩)	(青, 輕)	(青, 慶)
Unihan 的相似音	3	3	4	3
Unihan 的相似形	1	1	0	0
Unihan 的倉頡碼相似度	0	0	0	0
SIGHAN 提供的相似音	3	0	4	3
SIGHAN 提供的相似形	1	1	0	0
SIGHAN 提供的同部首筆畫	0	0	0	0
Unihan 提供的更正字字頻	3	0	2	1
A 的 UnigramLMScore (log)	-3.1820	-3.1820	-3.1820	-3.1820
B 的 UnigramLMScore (log)	-2.9964	-4.7012	-3.2999	-3.5611
FastText 餘弦相似度	0.0530	0.2770	0.4030	0.2803

doi:10.6342/NTU201701112

在勘誤表	X	X	O	X

從調整 SVM 參數獲得的模型中，選取具有最高準確度做為我們最後的分類器。依據此分類器重新分類所有的字對。若是分類器判斷為正樣本，則將其納入新的錯字更正集，反之判斷為負樣本則不加入。

我們將我們新的錯字更正集與其他只使用相似音或相似形的錯字更正集合在一起比較，詳細如表格 21，表中包含了具有更正字可更換的錯字，及其平均的更正字，還有在聯合報標準測試集的涵蓋率。從表中可得出相似音在測試集的涵蓋率相對相似形的涵蓋率高很多，但因相似音的種類過多，包含了同音同調、同音異調、近音同調、近音異調這四種類別，所以會大幅增加平均的更正字數量，增加系統的計算負擔。而集合所有的種類，雖可獲得極高的涵蓋率，但同時也會大幅增加系統的計算負擔。而透過分類器得到新的錯字更正集，相較於其他的集合，具有較小的平均更正字數量，卻有著更好的涵蓋率，代表著分類器成功的從已知的勘誤表擴增到更完整的錯字更正集，包含了音節及形狀的相似字，並同時保有系統的效率。

表格 21 各種錯字更正集的資訊及在標準測試集的涵蓋率

集合	**錯字**	**平均的更正字**	**涵蓋率 Recall**
Unihan 相似音集合	5361	185.9973	0.7880
Unihan 相似形集合	5361	24.2728	0.3671
SIGHAN 相似音集合	5361	57.1428	0.7761
SIGHAN 相似形集合	5361	26.2234	0.3910
SIGHAN 同部首同筆畫集合	5361	9.6353	0.0268
勘誤表集合	3769	2.4152	0.5667
ALL	5361	253.7377	0.9014
新的錯字更正集	**5338**	**20.8262**	**0.8338**

doi:10.6342/NTU201701112

第 4 節 評估指標

為了評估中文改錯字系統，這邊我們採用了標準 SIGHAN 2015 的規定，此規定如下：將改錯的任務分成兩個階段，偵測階段與更正階段。偵測階段要找到句子中錯字的位置，更正階段則是要找出錯字的更正字。

句子的得分標準如表格 22 的混淆矩陣，依原始答案有無錯字，與系統預測的有無錯誤，分成四種情況統計系統在測試集的得分。這個得分標準會同時套用在偵測階段與更正階段，關於兩個階段的細節在下方會繼續描述。系統在測試標準集中這四種情況的統計，進一步分析可得到如表格 23。其中的 Precision 代表系統的準確率，即系統偵測有錯的句子中，於實際上為正確的比率，而另一個數值 Recall 則代表系統的召回率，即在有錯的句子中，系統能正確偵測的比率。若句子沒有錯字，但系統卻判定為有錯，這個比率即稱 False Positive Rate。

表格 22 混淆矩陣 (Confusion Matrix)

		系統結果	
		句子有錯字	句子沒有錯字
正確答案	句子有錯字	True Positive (TP)	False Negative (FN)
	句子沒有錯字	False Positive (FP)	True Negative (TN)

表格 23 五種評測的標準

1	$\textbf{False Positive Rate (FPR)} = \frac{\boldsymbol{FP}}{\boldsymbol{FP + TN}}$
2	$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$
3	$\text{Precision} = \frac{TP}{TP + FP}$
4	$\text{Recall} = \frac{TP}{TP + FN}$
5	$\text{F1} - \text{Score} = \frac{2 * Precision * Recall}{Precision + Recall}$

doi:10.6342/NTU201701112

在介紹完評測標準值後，則開始介紹每個階段相對應的細節。在偵測階段中，系統的目標是要找到句子錯字的位置，若是句子中有一個錯字未被找到或是正確字被誤判，則該句在偵測階段，不會獲得任何分數。舉表格 24 為例子，這組標準集總共 5 句，有錯的句子占了 3 句，沒有錯的句子則占了 2 句。編號 1 的句子沒有錯字，但因系統回傳有錯字，這句不給予任何分數。編號 2 句子錯字的位置是 3,4，系統也回傳 3,4，則這句可獲得全部分數。編號 5 不會獲得任何分數，因為系統回傳的錯字位置不是正確的。依表格 24 系統在測試集的表現，在偵測階段的得分如表格 25 所示。

表格 24 評測標準的範例

編號	正確答案	系統結果	偵測階段	更正階段
1	0	5, 玩	X	X
2	3, 健, 4, 康	3, 件, 4, 康	O	X
3	8, 誤, 41, 情	8, 誤, 41, 情	O	O
4	0	0	O	O
5	10, 觀	11, 觀	X	X

表格 25 評測標準範例的結果

	偵測階段	更正階段
False Positive Rate	1/2	
Accuracy	3/5	2/5
Precision	2/4	1/4
Recall	2/3	1/3
F1-Score	0.57	0.28

在更正階段中，系統的目標是要更正在偵測階段中找到的錯字，所以其先決條件為有錯的句子且在偵測階段是正確的。其計分如下，若有成功的更換句子中的所有錯字，則獲得該句的分數，若有一個沒有正確的更正，則失去該句在更正

doi:10.6342/NTU201701112

階段的分數。舉上方的表格 24 為例，有錯字的句子只有編號 2,3,5，但因編號 5 在偵測階段時的位置錯誤，所以在更正階段直接認定失敗，而編號 2 的句子，則因為第 3 個字的錯誤更正，所以也認定失敗。因此在這個測試集中，只有編號 3 是成功的。在更正階段的結果請看表格 25。

第 5 節評測結果

第 1 項 HMM 的比較

不同的權重

首先我們使用二元語言模型(Bigram)做為我 LM 的計算，配上雜訊通道模型(NCM)之間的權重公式 (10)，並將這個系統在我們的聯合報標準測試集做測試。

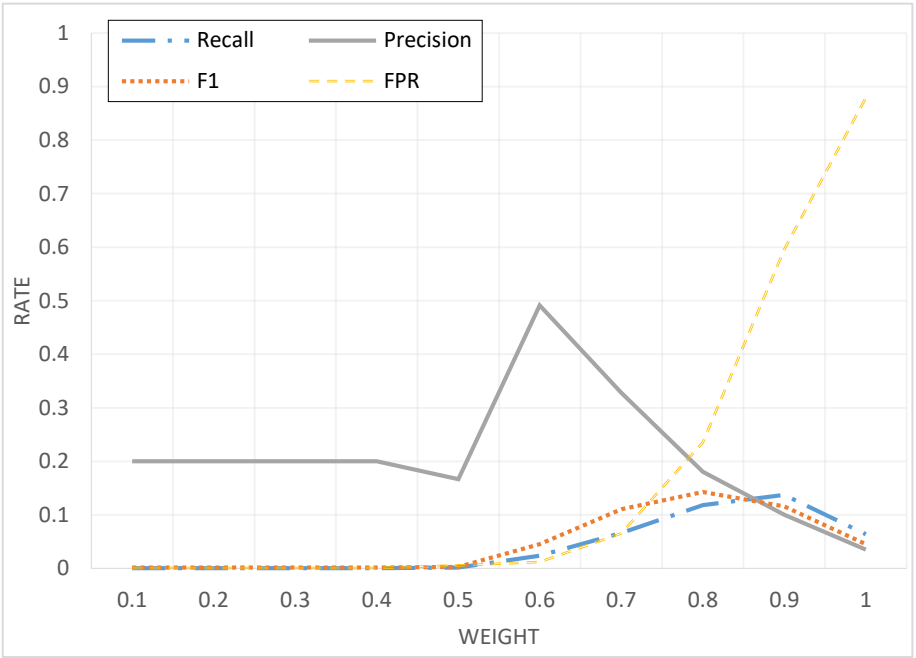
$$\text{Score} = (\text{Weight}) * \text{LM} + (1 - \text{Weight}) * \text{NCM} \quad (10)$$

結果如表格 26，分為偵測階段及更正階段，從表中可知當 NCM 採計的比例較高時，句子的錯誤無法抓到，且 Precision 也較低，直到權重大於一半時，即 LM 與 NCM 的分數至少平均時，Precision 及 Recall 才會明顯的上升，隨著 LM 的權重比率開始上升時，Precision 開始下降，但 Recall 反而會上升。最後，若是不採計 NCM，只考慮字更換後新句子的 LM 計分，結果在 Precision 會大幅的下降，在於 False Positive Rate 大幅上升有關，系統會誤認正確的句子為錯誤。透過圖表 8 可更了解權重影響的結果。

doi:10.6342/NTU201701112

表格 26 HMM 在 Bigram 的結果

Ngram	Weight	偵測階段					更正階段			
		FPR	A	P	R	F1	A	P	R	F1
2	**0.1**	0.0008	0.5000	0.2000	0.0008	0.0016	0.5000	0.2000	0.0008	0.0016
2	**0.2**	0.0008	0.5000	0.2000	0.0008	0.0016	0.5000	0.2000	0.0008	0.0016
2	**0.3**	0.0008	0.5000	0.2000	0.0008	0.0016	0.5000	0.2000	0.0008	0.0016
2	**0.4**	0.0008	0.5000	0.2000	0.0008	0.0016	0.5000	0.2000	0.0008	0.0016
2	**0.5**	0.0058	0.4979	0.1667	0.0016	0.0033	0.4979	0.1667	0.0016	0.0033
2	**0.6**	0.0123	**0.5058**	**0.4915**	0.0238	0.0455	**0.5025**	**0.3559**	0.0173	0.0329
2	**0.7**	0.0657	0.5004	0.3279	0.0666	0.1107	0.4930	0.2551	0.0518	0.0861
2	**0.8**	0.2350	0.4417	0.1807	0.1183	**0.1430**	0.4293	0.1430	0.0937	**0.1132**
2	**0.9**	0.5965	0.2703	0.1004	**0.1372**	0.1159	0.2555	0.0787	**0.1076**	0.0909
2	**1**	0.8784	0.0929	0.0352	0.0641	0.0454	0.0859	0.0275	0.0501	0.0355



圖表 8 HMM 在 Bigram 的分析圖

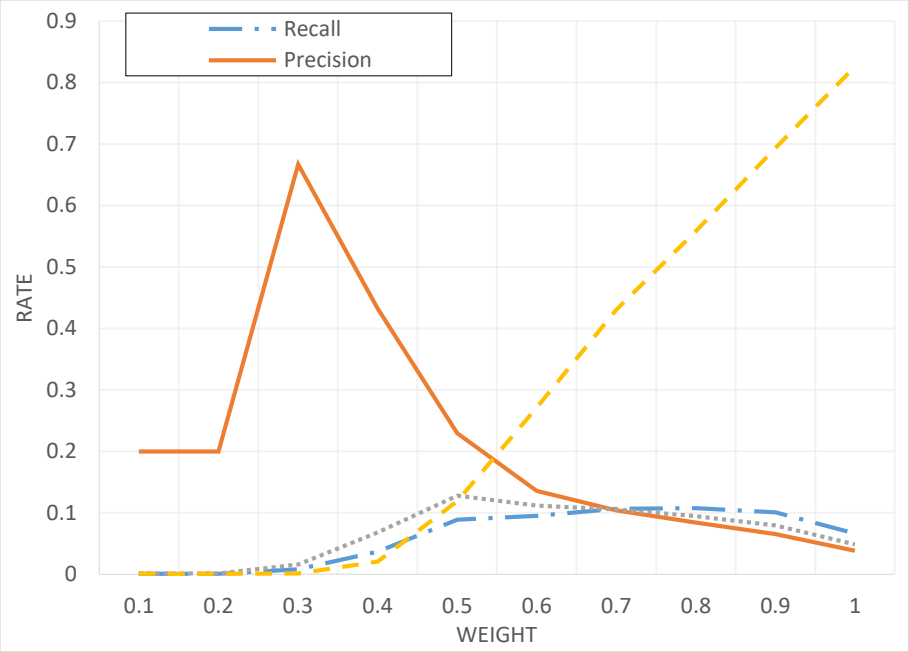
不同的語言模型計算方式

我們改使用 Trigram 作為我們的語言模型計算，其結果如表格 27。在 Trigram 中，相較於 Bigram，當權重較低時，Precision 特別的高。透過圖表 9 可更了解權重影響的結果。

doi:10.6342/NTU201701112

表格 27 HMM 在 Trigram 的結果

Ngram	Weight	偵測階段					更正階段			
		FPR	A	P	R	F1	A	P	R	F1
3	**0.1**	0.0008	0.5000	0.2000	0.0008	0.0016	0.5000	0.2000	0.0008	0.0016
3	**0.2**	0.0008	0.5000	0.2000	0.0008	0.0016	0.5000	0.2000	0.0008	0.0016
3	**0.3**	0.0016	0.5033	**0.6667**	0.0082	0.0162	0.5033	**0.6667**	0.0082	0.0162
3	**0.4**	0.0205	**0.5082**	0.4327	0.0370	0.0681	0.5037	0.3269	0.0279	0.0515
3	**0.5**	0.1191	0.4848	0.2293	0.0887	**0.1280**	**0.4692**	0.1486	0.0575	**0.0829**
3	**0.6**	0.2712	0.4121	0.1357	0.0953	0.1120	0.3969	0.0924	0.0649	0.0763
3	**0.7**	0.4306	0.3381	0.1042	0.1068	0.1055	0.3229	0.0745	0.0764	0.0755
3	**0.8**	0.5579	0.2749	0.0844	**0.1076**	0.0946	0.2613	0.0631	**0.0805**	0.0708
3	**0.9**	0.6935	0.2038	0.0657	0.1011	0.0796	0.1935	0.0524	0.0805	0.0635
3	**1**	0.8258	0.1204	0.0383	0.0666	0.0486	0.1134	0.0303	0.0526	0.0384



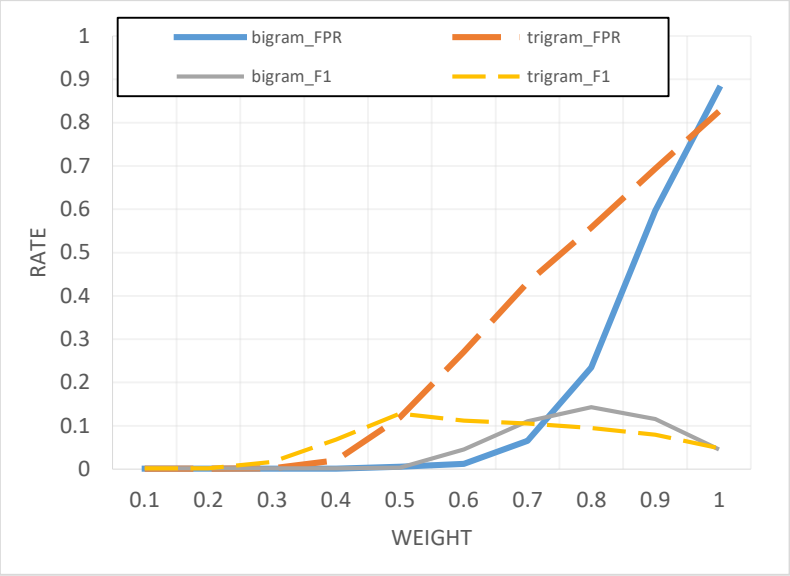
圖表 9 HMM 在 Trigram 的分析圖

小結

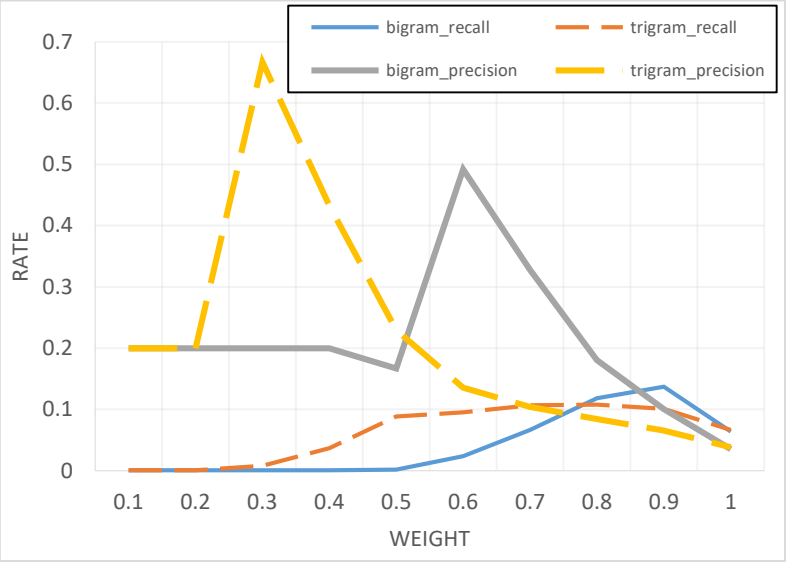
從圖表 10 中得知，不論是在 Bigram 或 Trigram 中，權重介於 0.5-0.8 時，F1-score 的成績都不錯。而隨著 LM 佔的比重越來越大時，FPR 隨著上升，可從圖表 11 也發現 Recall 也會跟著上升，代表著 LM 的提高，句子的越容易被修改，

doi:10.6342/NTU201701112

但從 Precision 的降低，也確定句子的修改增加，不代表可以抓取更多有錯的句子，而是在 LM 與 NCM 之間的搭配，才能有效的堅固 Recall 及 Precision 的準確度。



圖表 10 HMM 的 Bigram 與 Trigram 的比較（FPR, F1-Score）



圖表 11 HMM 的 Bigram 與 Trigram 的比較（Recall, Precision）

第 2 項 Beam Search

基本的測試

同 HMM 的測試方式，使用不同的 ngram，來測試結果。這邊堆疊的大小設

doi:10.6342/NTU201701112

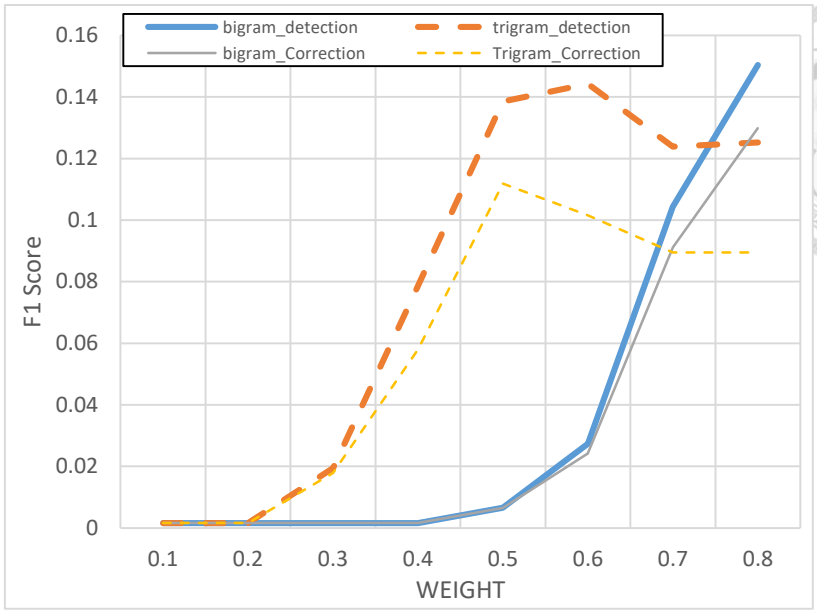
定為 10，結果如表格 28。相較於 HMM，LM 的權重需要大於一定比例才會有明顯的進步，從表中可得知權重至少要大於一半。

表格 28 Beam Search 在堆疊 10 的結果

Ngram	Weight	偵測階段					更正階段			
		FPR	A	P	R	F1	A	P	R	F1
2	**0.1**	0.0008	0.5	0.2	0.0008	0.0016	0.5	0.2	0.0008	0.0016
2	**0.2**	0.0008	0.5	0.2	0.0008	0.0016	0.5	0.2	0.0008	0.0016
2	**0.3**	0.0008	0.5	0.2	0.0008	0.0016	0.5	0.2	0.0008	0.0016
2	**0.4**	0.0008	0.5	0.2	0.0008	0.0016	0.5	0.2	0.0008	0.0016
2	**0.5**	0.0008	0.5012	0.5000	0.0033	0.0065	0.5012	0.5000	0.0033	0.0065
2	**0.6**	0.0016	0.5062	0.7391	0.0140	0.0274	0.5053	0.6522	0.0123	0.0242
2	**0.7**	0.0345	0.5119	0.4931	0.0583	0.1043	0.5082	0.4306	0.0509	0.0911
2	**0.8**	0.1487	0.4799	0.2449	0.1085	0.1503	0.4725	0.2115	0.0937	0.1298
3	**0.1**	0.0008	0.5	0.2	0.0008	0.0016	0.5	0.2	0.0008	0.0016
3	**0.2**	0.0008	0.5	0.2	0.0008	0.0016	0.5	0.2	0.0008	0.0016
3	**0.3**	0.0008	0.5045	0.75	0.0098	0.0194	0.5041	0.6875	0.0090	0.0178
3	**0.4**	0.0262	0.5086	0.4045	0.0435	0.0786	0.5028	0.29771	0.0320	0.05786
3	**0.5**	0.0863	0.5016	0.3053	0.0896	0.1385	0.4930	0.2465	0.0723	0.1118
3	**0.6**	0.2851	0.4199	0.1706	0.1249	0.1442	0.4014	0.1201	0.0879	0.1015
3	**0.7**	0.4067	0.3574	0.1262	0.1216	0.1238	0.3406	0.0912	0.0879	0.0895
3	**0.8**	0.4963	0.3196	0.1163	0.1356	0.1252	0.3003	0.0832	0.0970	0.0895

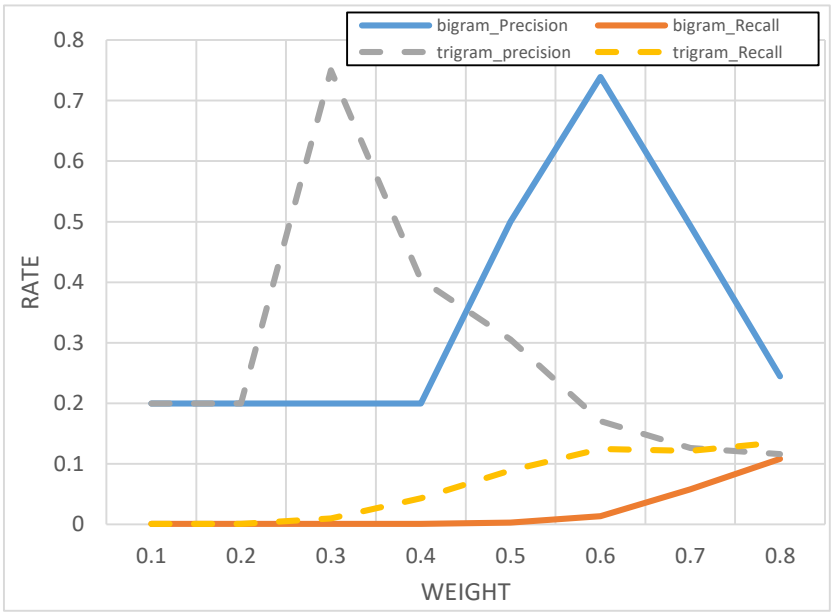
從圖表 12 的分析圖可看出 Trigram 在權重 0.5, 0.6 有較好的 F1-Score，而 Bigram 整體偏低，但隨著權重的上升，F1-score 也隨著上升。

doi:10.6342/NTU201701112



圖表 12 Beam Search 在 Bigram, Trigram 的 F1 Score 比較

圖表 13 則是觀察 Bigram, Trigram 在偵測階段時的 Precision/Recall，從圖中可看出結果跟 HMM 非常的相似，Trigram 在權重較小時有較高的 Precision。而 Bigram 則是整體的 Recall 值偏低。



圖表 13 Beam Search 在 Bigram, Trigram 的 Recall/Precision 比較

修改堆疊的大小

縮小堆疊大小到 5 來加速整體的系統效率，其結果沒有任何改變，結果如表格 29，針對兩組不同堆疊的偵測階段的 Precision 及 Recall 比較。兩者之間的數

doi:10.6342/NTU201701112

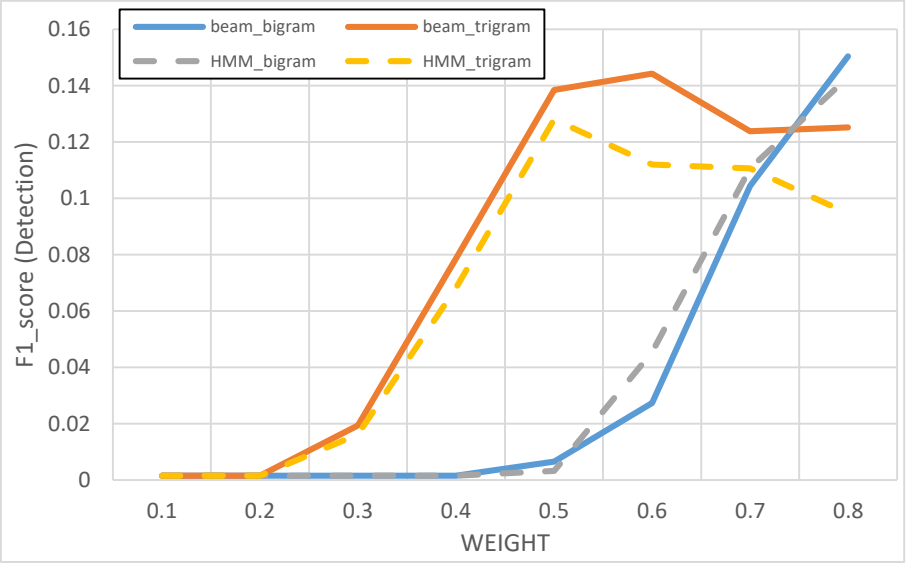
據，不論是 bigram 或 Trigram，亦或是權重的變化，彼此的結果非常相近，代表著正確的更正字在堆疊大小大於 5 以上，都可以獲得相同的結果。

表格 29 不同的堆疊大小比較

堆疊大小		5	10	5	10	5	10
Ngram	Weight	Precision	Precision	Recall	Recall	F1	F1
2	0.5	0.5000	0.5000	0.0033	0.0033	0.0065	0.0065
2	0.6	0.7391	0.7391	0.0140	0.0140	0.0274	0.0274
2	0.7	0.4931	0.4931	0.0583	0.0583	0.1043	0.1043
2	0.8	0.2472	0.2449	0.1085	0.1093	0.1515	0.1503
3	0.5	0.3053	0.3053	0.0896	0.0896	0.1385	0.1385
3	0.6	0.1706	0.1706	0.1249	0.1249	0.1442	0.1442
3	0.7	0.1269	0.1262	0.1216	0.1216	0.1242	0.1238
3	0.8	0.1171	0.1163	0.1356	0.1356	0.1256	0.1251

第 3 項 HMM 與 Beam Search 的比較

兩個方法出來的結果極其相似。



圖表 14 HMM 與 Beam Search 的 F1-score 比較

第 4 項 重排序加入

不論是 Beam Search 或 HMM，我們皆使用以字為基礎的語言模型，所以我

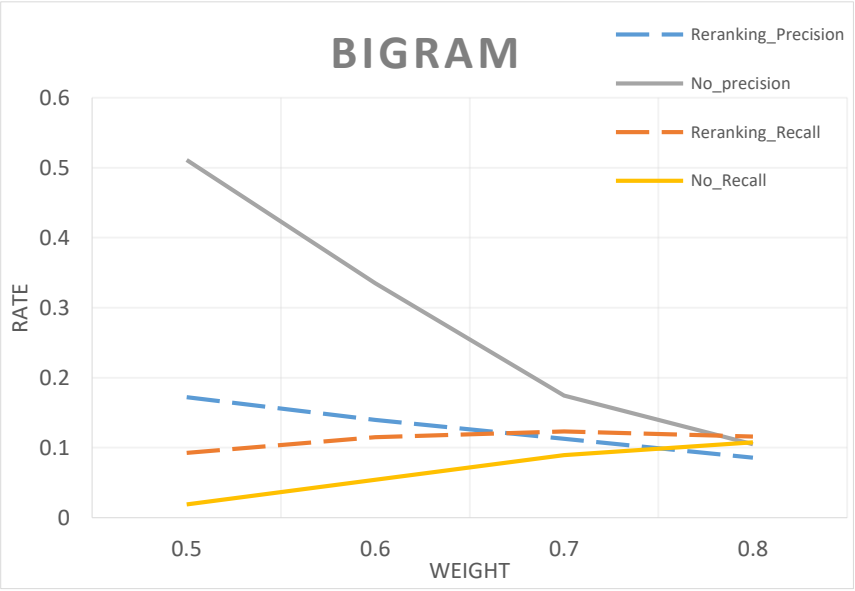
doi:10.6342/NTU201701112

們嘗試在結果的時候改用以詞為基礎的語言模型重新計分。我們這邊使用的斷詞器是用 JIEBA。

我們將 HMM 的每個字記錄多條路徑，並將最後的所有路徑重新排序，結果如表格 30。從表中可看到重排序後，F1-Score 皆有優化，舉 Bigram 的 Precision 與 Recall 比較為例子(圖表 15)，可發現原因在於 Recall 的上升，但同時 Precision 卻有所下降。

表格 30 在 HMM 上外加重排序

HMM		偵測階段 F1-Score		更正階段 F1-Score	
Ngram	Weight	No	Reranking	No	Reranking
2	0.5	0.03645	0.120662	0.031696	0.100374
2	0.6	0.093352	0.126183	0.077793	0.100045
2	0.7	0.11835	0.117878	0.094463	0.09666
2	0.8	0.106288	0.098601	0.082759	0.08042
3	0.5	0.134021	0.136585	0.090206	0.104065
3	0.6	0.111421	0.130208	0.077066	0.098958
3	0.7	0.105882	0.11088	0.075294	0.083853
3	0.8	0.093794	0.088624	0.069817	0.070767



圖表 15 Bigram 在加入重排序前後的 Precision 與 Recall 比較

整體而言，在加入重排序後的結果不如預期，可能的原因在於我們使用 JIEBA 斷詞器無法正確的斷詞，導致後續以詞為基礎的語言模型計分連帶著不準。舉句

doi:10.6342/NTU201701112

子”幸虧我會說德問”為例，如表格 31，從表中可看出 JIEBA 的斷詞相較於 CKIP 不是很理想，這句若是改用 CKIP 斷詞，在重排序的排名上，會成功的改正這句，代表著較好的斷詞器應該可以優化我們在這邊的結果。

表格 31 斷詞系統比較

排名	**原始堆疊**	**JIEBA 斷詞**	**CKIP 斷詞**
1	幸虧我會說得問	幸虧/我會/說德問	幸虧/我/會/說/德文
2	幸虧我會說得文	幸虧/我會/所的問	幸虧/我/會/說/得/問
3	幸虧我話說得問	幸虧/我會話/的問	幸虧/我/會/說/的/問
4	幸虧我會說德文	幸虧/我會話/得問	幸虧/我/會/所得文
5	幸虧我會說的文	幸虧/我會/說/德文	幸虧/我/會/說/得/文

doi:10.6342/NTU201701112

第 5 章 結論及未來展望

第 1 節 結論

從聯合報的改稿記錄，可看到專業寫作者與一般寫作者的錯字差異性，在於更細微的誤用，亦或是形狀非常相近的異體字，也看到許多報章雜誌才會出現的修改。相較於其他的錯字測試集，錯字類型不只有侷限在形音字上，還有簡體字的錯誤也會出現，所以在改錯上面要加入更多的處理。

在 Bigram 與 Trigram 的比較下，通常 Bigram 可以獲得好的 Precision，Trigram 則獲得好的 Recall，不論是在我們的 HMM 或者是 Beam Search 上。

加入重排序後，結果不如預期的好，可能在於我們使用 JIEBA 做為我們的斷詞系統，改用其他的斷詞系統應該會有更好的結果。

第 2 節 未來展望

關於改稿記錄的截取，此篇論文只要專注在一個字的修改，但有時書寫時，會發生兩個字皆有錯的可能，或是有些是單純新增或刪除的單字，透過這些資訊的理解，或許可讓我們更能理解專業編輯者常見的修改，不會只侷限在字的上面，而可以擴展到整個句子的理解分析上。

在此篇論文的雜訊通道模型的機率為假設生成，但若我們可藉由計算聯合報的字詞轉換替代，此資料應該可更貼近實際的狀況，但因資料取得時間稍晚，未能將這個機率安插在我們的雜訊通道模型上面。

doi:10.6342/NTU201701112

參考文獻

[1] Yih-Ru Wang, Liang-Chun Chang, Yeh-Kuang Wu and Yuan-Fu Liao (2013), "Conditional Random Field-based Parser and Language Model for Traditional Chinese Spelling Checker", The 7th SIGHAN Workshop on Chinese Language Processing (SIGHAN-7).

[2] Shuiyuan Zhang, Jinhua Xiong, Jiapeng Hou, Qiao Zhang and Xueqi Cheng (2015), "HANSpeller++: A Unified Framework for Chinese Spelling Correction", Eighth SIGHAN Workshop on Chinese Language Processing.

[3] Chao-Lin Liu, Min-Hua Lai, Kan-Wen Tien, Yi-Hsuan Chuang, Shih-Hung Wu and Chia-Ying Lee (2011), "Visually and phonologically similar characters in incorrect Chinese words: Analyses, identification, and applications", ACM Transactions on Asian Language Information Processing, volume 10, pages 39.

[4] Yih-Jeng Lin, Feng-Long Huang and Ming-Shing Yu (2002), "A CHINESE SPELLING ERROR CORRECTIONS SYSTEM", Processings of the Seventh Conference on Artificial Intelligence and Applications.

[5] Chuan-Jie Lin and Wei-Cheng Chu (2015), "A Study on Chinese Spelling Check Using Conufision Sets and N-grams Statistics.", International Journal of Computational Linguistics and Chinese Language Processing. Volume 20, pages 23-47.

[6] Shih-Hung Wu, Yong-Zhi Chen, Ping-Che Yang, Tsun Ku and Chao-Lin Liu (2010), "Reducing the False Alarm Rate of Chinese Character Error Detection", Proceedings of CIPS-SIGHAN Joint Conference on Chinese Language Processing, pages 54-61

[7] Hsun-Wen Chiu and Jason S. Chang (2014), "Chinese Spell Checking Based on

doi:10.6342/NTU201701112

Noisy Channel Model", Master Thesis on National Tsing Hua University.

[8] Yih-Ru Wang and Yuan-Fu Liao (2014), "NCTU and NTUT's Entry to CLP-2014 Chinese Spelling Check Evaluation", Association for Computational Linguistics, In Proceedings of the 3rd CIPS-SIGHAN Joint Conference on Chinese Language Processing, pages 216-219.

[9] Hsun-Wen Chiu, Jian-Cheng Wu and Jason S. Chang (2013), "Chinese Spelling Checker Based on Statistical Machine Translation", Proceedings of SIGHAN-7, pages 49-53.

[10] Yih-Ru Wang and Yuan-Fu Liao (2015), "Word Vector/Conditional Random Field-based Chinese Spelling Error Detection for SIGHAN-2015 Evaluation", Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing (SIGHAN-8), pages 46-49.

[11] Jinhua Xiong, Qiao Zhang, Shuiyuan Zhang, Jianpeng Hou and Xueqi Cheng (2015), Computational Linguistics and Chinese Language Processing, Volume 20, No.1, pages 1-22.

doi:10.6342/NTU201701112