基於語音指令的中文 ASR 錯誤校正系統設計與實現
Speech-Driven Editing System for Chinese ASR Errors

丁仕杰 **Sji-Jie Ding**, 張嘉惠 **Chia-Hui Chang**, 簡資烜 **Zi-Xuan Jian**
國立中央大學資訊工程學系
Department of Computer Science and Information Engineering, National Central University
shijied675@gmail.com, chiahui@g.ncu.edu.tw, 114522129@cc.ncu.edu.tw

摘要

儘管人工智慧技術近年進步顯著，但 ASR 系統仍難以應對現實中由發音和同音詞引起的錯誤。爲解決此問題，本研究提出一種基於語音指令的校正方法。系統由三個模組組成：輸入分類器、命令分類器和校正標註器。爲了支援訓練和評估，我們透過 TTS 和 ASR 流程模擬 ASR 錯誤，以模擬潛在的錯誤，並基於漢字形聲字語言特徵及大型語言模型產生校正命令。實驗顯示基於兩種校正指令的混合模型，整體校正準確率超過 80% 且性能穩定。與手動校正相比，雖然較慢但仍有一定競爭力，顯示其實際部署的可行性。

Abstract

Despite recent advances in AI, ASR systems still struggle with real-world errors from pronunciation and homophones. To solve this issue, we propose a verbal-command-based correction system that enables users to utter natural-language instructions to refine recognition outputs with minimal effort. The system consists of three modules: an input classifier, a command classifier, and a correction labeler. To support training and evaluation, we simulate ASR errors via TTS and ASR pipelines to simulate the potential errors, followed by verbal correction commands issued based on linguistic features or LLMs. Experiments show that the overall system achieves over 80% correction accuracy and delivers stable performance. Compared to manual correction, this system also demonstrates highly competitive correction speed, which sufficiently indicates its feasibility for practical deployment.

關鍵字：語音辨識、錯誤修正、語音指令、自動修正模組、中文自然語言處理

Keywords: Automatic Speech Recognition, Voice Command, Error Correction

1 緒論

隨著自動語音辨識（ASR）技術的發展，語音輸入已逐漸融入日常生活，從智慧家電、語音助理到訊息傳遞，都能看見它的身影。

然而，儘管中文 ASR 在公開測試集的字元錯誤率（CER）已降至 3.05% (Xu et al., 2025)，但因個人發音差異及大量同音異字，實際錯誤率遠高於預期。例如，用户說「這個程式很棒」，可能被誤識別爲「這個城市很棒」。

爲此，許多相關研究提出了自動錯誤修正的方法。早期方法透過額外特徵（如聲學資訊 (Zhang et al., 2021)、多模態輸入 (Jiang et al., 2024a)）增強端到端模型。爲緩解序列到序列模型常見的過度修正問題（即引入新錯誤），有研究提出如 PGCC (Dong et al., 2024) 等框架。近年來，大型語言模型（LLM）已被應用於此任務，透過融合拼音等方式提升表現 (Li et al., 2025; Wei et al., 2024)。

與此相關的拼寫錯誤校正（SEC）任務，也發展出預測編輯操作（如 LASERTAGGER (Malmi et al., 2019)）或整合語言聲學線索（如 D^2C (Jiang et al., 2024b), ReLM (Liu et al., 2024)）的方法。近期 LLM 雖可透過提示詞技術 (Yang et al., 2023; Li et al., 2024) 應用於此，但仍需手動驗證。

綜上所述，雖然自動修正技術已取得顯著進展，但當修正失敗時，使用者仍需依賴鍵盤輸入，對於不擅長打字或雙手不便的情境並不友善。爲此，我們提出一套基於語音指令的 ASR 錯誤修正系統，允許使用者直接透過口說指令修改辨識錯誤，以減少鍵盤操作並提升使用體驗。

同時爲了模擬眞實場景，我們利用文句轉語音（TTS）與 ASR 建構資料集以生成潛在錯誤，並設計了符合中文使用者習慣的修正指令。本研究主要貢獻如下：

- 提出以口說指令修正 ASR 錯誤的系統，減少手動編輯，提升互動效率。

Proceedings of the 37th Conference on Computational Linguistics and Speech Processing (ROCLING 2025), pages 277–285
November 20-22, 2025

- 系統支援多種指令類型（新增、修改、刪除），並整合分類、解析、標註到自動編輯的統一流程。
- 利用 TTS 與 ASR 建構了多樣化的語音指令與場景資料，模擬辨識錯誤，產出高品質資料集以強化系統。

2 系統架構與資料集

本節將說明本研究所提出之語音辨識錯誤修正系統的整體架構與資料處理流程。該系統主要由三個核心模組組成：輸入分類器（Input Classifier）、指令分類器（Command Classifier）以及指令標註器（Command Labeler），分別負責識別輸入意圖、判斷修正類型，並產生對應的修正指令。在資料準備方面，本研究以現有文字資料集爲基礎，透過文字轉語音（Text-to-Speech, TTS）與語音辨識（Automatic Speech Recognition, ASR）技術模擬語音辨識錯誤，進一步設計多樣化的修正敘述，並依據三個模組的功能需求整理成對應的訓練資料集。以下將依序介紹各模組功能與資料處理方式。

2.1 系統架構

圖 1 爲我們系統的整體架構。我們將整個流程分成了三個模組，分別是輸入分類器、指令分類器和指令標註器。透過這三個模組，就能依照使用者提供的指令來提取各項資訊，修正原先的文字。

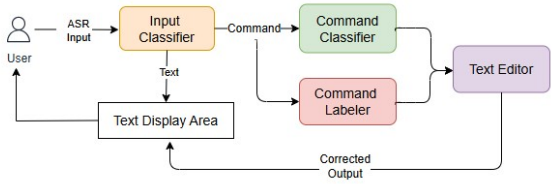

圖 1: 系統流程圖

2.1.1 輸入分類器

輸入分類器的作用爲分辨當前的語音輸入內容屬於單純的文字訊息，或是用來修改前文的指令。我們以 0 和 1 兩個標籤來表示文字和指令兩個不同的類別。當模型判斷該內容爲指令時，就會執行後續的指令分類器和指令標註器，提取指令修改需要的資訊。模型選擇的部分我們使用 BERT 的中文模型，如圖 2 所示。

2.1.2 指令分類器

我們將輸入分類器判斷爲指令的內容，當作輸入提供給指令分類器，讓模型判斷該指令

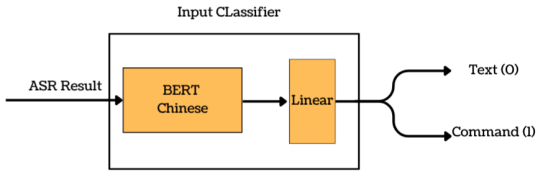

圖 2: 輸入分類器架構

屬於替換 (Replace)、新增 (Insert) 或是刪除 (Delete) 這三種類型的哪一個類別，其對應的標籤分別爲 0、1、2。和輸入分類器相同，模型我們一樣選擇使用 BERT 的中文模型，如圖 3 所示。

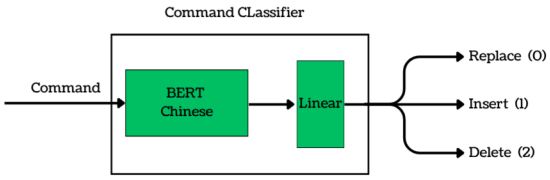

圖 3: 指令分類器架構

2.1.3 指令標註器

最後一個模組，我們將要修改的文字 (Text) 和指令 (Command) 結合，中間以 [SEP] 區隔，一起當作輸入提供給模型。模型會先對輸入進行斷詞 (tokenize)，接著判斷每一個 token 屬於"O"、"B-Modify" 或是"B-Filling" 的哪一個類別。"B-Modify" 會出現在要修改的文字中的某個位置，"B-Modify" 則會出現在指令中的某個位置，其餘無關的部分會以"O" 來表示，整體架構如圖 4 所示，與另外兩個模組不同的地方在於，模型我們選擇了 BERT 的多語言模型。

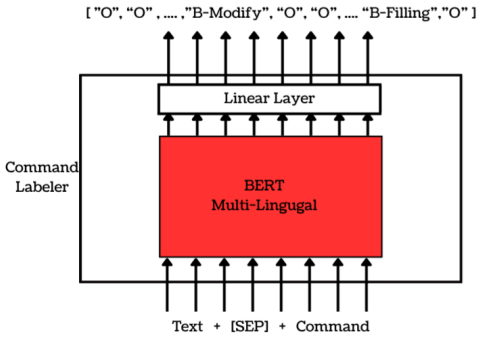

圖 4: 指令標註器架構

經過這三個模組之後，我們會得到指令類型、修改範圍和填入字這三個重要的元素，依

據不同的指令類型，我們使用不同的應對方法進行修改。舉例來說，如果是替換的指令，我們就會將修改範圍的內容和填入字直接進行替換；如果是刪除，則就單純刪除修改範圍的內容；最後針對新增的指令，我們會在修改範圍的前一個位置插入標註的填入字。對於整體架構來說，這三個模組都是不可或缺的一部分。

2.2 資料集準備

對於上述三個模組的訓練資料，我們選擇了兩個現有資料集來當作基礎，分別是 SIGHAN-2015[1] 和 zh-tw-wikipedia[2]。SIGHAN 是針對中文拼字檢查任務的資料集，內容包含錯誤句子、正確句子以及修正步驟，因此被廣泛使用。zh-tw-wikipedia 則包含了截止至 2023 年 5 月，中文維基百科 2,533,212 篇條目的文字內容，我們先將每篇文章的內容進行斷句，並過濾掉中文以外語言的內容。爲了模擬語音辨識產生的輸出，我們將這些資料透過雅婷文字轉語音[3]的文字轉語音功能，將文字轉換成音訊檔案。選擇該服務的原因在於其合成語音提供了較接近於台灣本土的口音，更加貼合現實情境。得音訊檔案後，我們再使用 Google 的 Speech-to-Text AI [4] 的語音轉文字功能，將音訊檔轉換回文字，以此來模擬使用者利用語音辨識得到的文字內容，如圖5上半部所示。

對於每筆原始句子和經過 TTS 和 ASR 過後的結果，我們計算兩者之間的 Levenshtein distance (Levenshtein et al., 1966)，以此來得知我們需要對錯誤句子進行多少次 Replace, Delete 和 Insert 的動作將其還原成正確的句子。然而，分析過後，我們大多數的修正動作都是 Replace，因爲語音辨識的錯誤較少發生多字或是少字的情況，這會導致我們後續產生指令資料的比例不平均。爲了解決這個問題，我們將一部份 Replace 指令拆分成 Delete 和 Insert 兩個動作。舉例來說，原先的修正方式是將” 冰” 這個字替換成” 濱”，拆分過後變成先刪除” 冰” 字，再新增” 濱” 這個字，等同於直接進行替換的動作，解決資料不平衡的問題，如圖 5 下半部所示。

2.2.1 指令資料生成

我們希望每則指令能更貼近眞實使用情境。以中文字爲例，當需要修改某個字時，使用者可能會透過部件組合或常見詞語來描述，例如：「弓長張」、「耳東陳」、「祝福的祝」、「辛苦的苦」等。爲了模擬這樣的敘述風格，我們針對不同資料集設計了不同的指令產生方式。

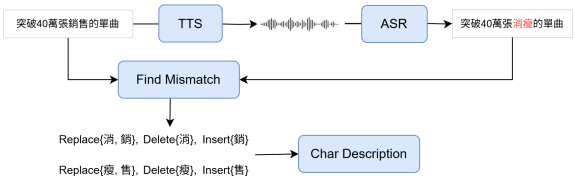

圖 5: 錯誤資料 & 刪除、新增錯誤產生流程

在 SIGHAN-2015 資料集中，我們使用 GPT-4o 並下達提示語，要求模型以常用詞語來描述修改內容，例如前述的「祝福的祝」，藉此產生完整的指令格式（詳見附錄6.1）。

至於 zh-tw-wikipedia 資料，我們依據以下優先順序，產生原字與替換字的敘述方式：

1. 部首不同但同音的字（例如：「人部的他」vs. 「女部的她」）
2. 可拆解爲具體部件的字（例如：「弓長張」）
3. 使用常用詞彙進行描述

爲實現上述規則，我們從「CNS11643 中文標準交換碼全字庫」[5]彙整了常用字的注音、部首與部件資訊，以判斷是否滿足條件一和二。至於常用詞彙描述，我們不依賴大型語言模型，而是參考標準的常用字詞列表；若單一字元對應多個候選詞彙，則隨機擇一使用。表 1 展示了由這兩種方法生成的指令範例。

Command	Type
把人部的佛改成弓部的弗	Replace
請在民主的民前面新增製造的造	Insert
刪除自心息	Delete

表 1: 指令範例

2.3 各模組資料類型

我們從上述資料中分別建立了三個不同的資料集，用於訓練我們系統的三個模組，分別是輸入分類器、指令分類器和指令標註器。

2.3.1 輸入類型資料集

我們將每筆 Wrong text 和其對應的指令分別標示爲 0 和 1，如表 2 所示，代表文字和指令兩個類別，即可用於輸入分類器模組的訓練，辨識使用者當前說的內容屬於其中哪一項。

[1] http://sighan.cs.uchicago.edu/
[2] https://huggingface.co/datasets/zetavg/zh-tw-wikipedia
[3] https://tts.yating.tw/
[4] https://cloud.google.com/speech-to-text?hl=zh-TW
[5] https://www.cns11643.gov.tw/

Input	Type	Label
我知道，你很久以前找工作很辛苦。	Text	0
請把幸福的幸改成辛苦的辛。	Command	1

表 2: 輸入類別範例

2.3.2 指令類型資料集

指令則分成三個類別，分別是替換、新增和刪除，對應的標籤爲 0, 1, 2，如表 3。此資料集將使用於指令分類器的訓練，因爲我們需要確認指令的類型爲何，才能在最後修改的步驟根據其類型來做出對應的修正。

Command	Type	Label
請把發財的發改成爬山的爬。	Replace	0
請在民主的民前面新增製造的造。	Insert	1
刪除週末的週。	Delete	2

表 3: 指令類別範例

2.3.3 指令標註器資料集

當我們得到生成錯誤句子和對應的指令資料後，就能得知錯誤句子中的修改範圍，還有指令中的填入字爲何。因此我們將錯誤句子和指令串接在一起，中間使用 [SEP] 符號作爲分隔。接著我們使用 google-bert/bert-base-multilingual-cased [6]的 Tokenizer 將整合的句子分割成 token，將修改範圍的位置標示爲 B-Modify，填入字的位置標示爲 B-Filling，其他部分則標示爲 O。選擇該 Tokenizer 的原因在於後續訓練我們皆會使用 BERT 系列的模型，其中 multilingual 的模型對於數字以及英文的斷詞有較好的能力。在指令標註器的訓練過程，我們將使用 Sequence2Tag 的模型，幫助我們成功標記出這兩個關鍵的元素。以下爲資料範例，Input 前半段的內容，「你的回答我受到了，謝謝。」，爲需要修改的句子；後半段的句子，「刪除受到的受」，則爲修正指令，兩者中間加入 [SEP] 區分。Output Label 則表示輸入進行斷詞後，經由模型輸出的標記結果，B-Modify 對應的位置爲「你的回答我受到了，謝謝。」中的「受」字，B-Filling 則對應指令中「刪除受到的受」最後的「受」字。

- **Input:** 你的回答我受到了，謝謝。[SEP] 刪除受到的受
- **Output Label:** ['O', 'O', 'O', 'O', 'O', 'B-Modify', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-Filling']

[6]https://huggingface.co/google-bert/bert-base-multilingual-cased

3 實驗

3.1 實驗設置

在實驗中，我們依據 Section 2.2 所述的方法，爲每個模組建立所需的訓練資料，並針對 SIGHAN 與 zh-tw-wikipedia 兩個資料集分別訓練模型。如表 4 所示，我們亦採用兩者混合的 Mix dataset，以增強模型對不同指令敘述方式的適應能力。

本研究選用 BERT（Bidirectional Encoder Representations from Transformers）(Devlin et al., 2019) 系列模型進行訓練。BERT 採用基於 Transformer 的雙向編碼器架構，透過遮蔽語言建模（MLM）與下一句預測（NSP）進行預訓練，具備優秀的語意理解能力，並能靈活應用於分類與序列標記等多項 NLP 任務。在本研究中，BERT 能有效處理錯誤位置辨識與指令類型分類兩種任務。

我們使用的模型爲 google-bert/bert-base-chinese [7] 以及 google-bert/bert-base-multilingual-cased，因爲兩者已在大規模中文語料上預訓練，很適合處理繁體中文文本。

Task	SIGHAN-15		zh-tw-wikipedia		Mix dataset	
	Train	Test	Train	Test	Train	Test
Input Type Classifier	4264	474	5202	578	9826	1052
Command Classifier	1990	222	3321	369	5311	591
Command Labeler	2194	244	4510	501	6704	745

表 4: 各模組訓練&測試資料數量

3.2 輸入分類器

在輸入分類器的實驗中，我們使用基於 BERT 的中文預訓練模型進行訓練。資料總量如表 4 第一列所示。實驗結果如表 5 所示。模型名稱以訓練資料集命名，例如 Model-SIGHAN 表示使用 SIGHAN-15 資料集訓練的模型。

模型在不同資料集上皆能準確地判斷輸入內容爲純文字輸入或語音指令。儘管在混合資料集上的 F1 分數略降至 0.99，整體表現仍維持高準確度。此分類器的判斷能力對於後續系統的正確運作至關重要。

Model	Model-SIGHAN			Model-Wiki			Model-Mix		
	P	R	F1	P	R	F1	P	R	F1
BERT Chinese	1.0	1.0	1.0	1.0	1.0	1.0	0.99	0.99	0.99

表 5: 輸入分類器校能 (Performance of the Input Classifier)

[7]https://huggingface.co/google-bert/bert-base-chinese

3.3 指令分類器

在指令分類器的實驗中，我們需進一步判斷指令類型爲「替換」、「新增」或「刪除」，以便後續根據類型執行對應處理。資料總量如表 4 第二列所示。本實驗同樣使用基於 BERT 的中文預訓練模型進行訓練，實驗結果如表 6 所示。

與輸入分類器結果相同，模型在三種類型的指令分類任務中皆達到極高準確度，於三種資料集（SIGHAN、Wiki、Mix）上之 Precision、Recall 與 F1-score 均爲 1.0，顯示模型能穩定且正確地辨識不同類型的指令。值得注意的是，混合資料集（Model-Mix）包含不同的指令敘述方式，但模型在此仍維持完美表現，反映其在異質資料上的良好泛化能力。

這樣的結果顯示，模型不僅能學習清楚區分指令類型的語言特徵，也具備跨語域資料的適應能力，有助於提升後續修正的準確度。

Model	Model-SIGHAN			Model-Wiki			Model-Mix		
	P	R	F1	P	R	F1	P	R	F1
BERT Chinese	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

表 6: 指令分類器效能

3.4 指令標註器

指令標註器的任務需同時精準標示兩類位置資訊：原文中需修改的字詞（B-Modify），以及指令中欲替換的新字詞（B-Filling）。資料總量如表 4 第三列所示。本任務使用 mBERT（Multilingual BERT）模型進行訓練，主因是需標註的句子中可能包含非中文字詞，例如英數字或特殊符號，相較於前兩項使用的中文 BERT 模型，多語言模型在此具更佳的語言適應性。

實驗結果如表 7 所示，模型在三組資料集上皆展現穩定表現，F1 值介於 0.92 至 0.94 之間。SIGHAN 資料集上 F1 爲 0.92，略低於 Wiki 資料集的 0.94，推測可能與 SIGHAN 中錯字形式更爲多樣，標註任務較具挑戰有關；而混合資料集 Model-Mix 亦能維持 F1 score 0.93 的水準，顯示模型具一定泛化能力。

Model	Model-SIGHAN			Model-Wiki			Model-Mix		
	P	R	F1	P	R	F1	P	R	F1
mBERT	0.91	0.92	0.92	0.95	0.92	0.94	0.93	0.93	0.93

表 7: 指令標註器效能

3.5 系統整體效能

我們遵循圖1所示之流程，並使用「指令標註器」(Command Labeler) 的測試集來評估此框架。如表8所示，Model-SIGHAN 與 Model-Wiki 兩個模型在其領域內皆修正了超過 80% 的句子，但在交叉測試中則表現出較差的泛化能力。錯誤分析指出，大部分的失敗案例源於標註問題，以及對不熟悉的指令描述存在理解上的困難。相比之下，Model-Mix 在不同資料集上均取得了穩定的準確率，這表明混合資料的訓練方式能有效提升模型的泛化能力。這些發現凸顯了爲改善未來系統，收集更多樣化指令描述的重要性。

Model	SIGHAN (244)		Wiki (501)	
	Match	Acc	Match	Acc
Model-SIGHAN	198	0.81	191	0.39
Model-Wiki	145	0.59	420	0.84
Model-Mix	200	0.82	419	0.84

表 8: 不同模型在兩個資料集上的修正準確率與匹配數（Match / Acc），資料集後面的數字爲測試資料總比數。

3.6 語音指令辨識錯誤的影響

在資料集建構時，我們針對文字輸入進行錯誤模擬，建立了目前的 SIGHAN-15 與 zh-tw-wikipedia 兩個訓練資料集。然而，在實際應用中，使用者以語音下達指令的時候，可能因語音辨識錯誤而影響系統判斷。爲此，我們將上述兩個資料集中 Command Labeler 的指令，同樣透過文字轉語音（TTS）與語音轉文字（ASR）處理，產生模擬語音輸入錯誤的資料。我們將這些資料當作新的測試集，並依照上一小節的實驗設計，進一步分析在加入可能含有語音辨識錯誤的指令（Error Cmd）後，對整體流程準確率的影響。

實驗結果如表9 所示。可觀察到，在處理語音辨識錯誤的情況下，原始 Model-Mix 模型效能明顯下降；在 SIGHAN 測試集的準確率僅爲 0.64，而在 Wiki 測試集更低至 0.02，顯示語音辨識錯誤對部件／部首描述的指令影響尤爲顯著。

Model	SIGHAN		Wiki	
	Match / Acc	ΔAcc	Match / Acc	ΔAcc
Model-Mix	156 / 0.64	–	12 / 0.02	–
Model-Mix + NoisyCmd	151 / 0.62	-0.02	222 / 0.44	+0.42

表 9: 不同模型架構於語音錯誤模擬資料（SIGHAN 與 Wiki）上的表現比較，包含 Match 數、準確率（Acc）與其變化量（ΔAcc）。

爲改善上述問題，我們將這些含語音辨識誤差的資料加入原訓練集，重新訓練出 Model-Mix + NoisyCmd 模型。從表中結果可見，該模型在 Wiki 測試集的準確率從 0.02 顯著提升至 0.44，雖然在 SIGHAN 測試集略降至 0.62，

但整體而言對模型的實用性與穩定性有正向幫助。這說明適當擴增模擬語音輸入錯誤的訓練資料，有助於提升系統在實際語音應用場景中的效能。

綜合上述觀察，若希望系統能更有效地處理語音輸入所帶來的誤差，未來應進一步研究並調整錯誤處理策略，包括擴增訓練資料、設計錯誤修復機制，或結合語音辨識的置信分數等方法，以提升模型在實際語音應用場景中的整體效能。

4 眞實使用情境

4.1 建置 API

從上一章的實驗結果，我們已驗證系統架構的可行性。因此我們將本系統設計爲後端 API，允許任何具備語音辨識功能的應用串接使用。同時，我們記錄每次請求的 log，以蒐集更多樣的指令表達方式，協助日後擴充訓練資料。

目前提供兩個開放的 API，其中之一爲 Input Classifier，其流程如圖 6。當使用者透過語音辨識進行輸入時，互動端 (i.e. PC / 手機) 會辨識出語音輸入的結果，接著便將其送至 Input Classifier API，我們會預測出該段內容的類型，並回傳給互動端。如果該段文字內容爲純文字，即可直接顯示在輸入欄，讓使用者判斷該辨識結果是否正確，若該段敘述爲指令，即須呼叫下一個 Error Correction API。

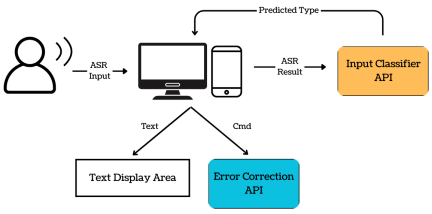

圖 6: Input Classifier API 流程

圖 7 爲 Error Correction API 的整體流程，互動端會將欲修改的句子以及對應的修正指令提供給該 API，我們會先使用 Command Classifier 分類接收到的 command 爲替換、新增和刪除中哪個類別，再透過 Command Labeler 標示出 text 和 command 中，修改位置 (B-Mofify) 以及填入字 (B-Filling) 兩個部分。最後透過 Text Editor 進行修改，直接回傳修改後的內容 (corrected text)。

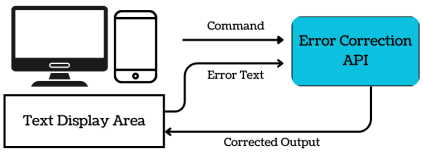

圖 7: Error Correction API 流程

4.2 實際應用

成功建立 API 後，接下來就是實際投入使用。這裡我們以 EduACT [8] 爲例，這是一個提供教師創作對話機器人的平台，協助學生課後學習以及進度追蹤。我們將系統串接至該平台的語音辨識，當使用者使用語音辨識產生內容後，便會將辨識結果傳送給 Input Classifier API，判斷使用者的意圖。舉例來說，某篇文章的名稱爲「數位勞動蟻民」，但使用語音辨識的話，結果會變成「數位勞動移民」。從表 10 的 log 結果也能看出，Input Classifier 的判斷並沒有錯誤。

項目	內容
輸入文字	數位勞動移民
預測結果	0
回傳	`POST /input_classifier/ → 200`

表 10: Case Study: 系統 log (Text)

這就是中文語音辨識實際容易遇到的錯誤情況之一，爲了應對這個狀況，使用者即可透過語音辨識下達指令，例如: 「把移民的移改成螞蟻的蟻」。可以看到表 11 的 log 紀錄，我們的系統也能正確的辨認出該文字內容屬於指令，因此回傳結果爲 1，接著便會透過 Error Correction API，判斷指令類型以及進行錯誤標記。

項目	內容
輸入文字	把移動的移改成螞蟻的蟻。
預測結果	1
回傳	`POST /input_classifier/ → 200`

表 11: Case Study: 系統 log (指令分類器)

接收到請求之後，我們會先利用 Command Classiifer 判斷該指令的類型，如表 12 所示，Command Classiifer 判斷這是一個替換指令，因此 command type 爲 0。接著便是串接欲修改的句子以及指令之後，由 Command Labeler 標記出錯誤位置 (B-Modify) 以及修改字

[8] https://eduact.csie.ncu.edu.tw/

詞 (B-Filling) 的位置。最後結合這些預測結果，讓 Text Editor 直接進行修正。

項目	內容
輸入文字	數位勞動移民
使用者指令	把移動的移改成螞蟻的蟻。
預測指令類型	0（替換）
BIO 標註預測	['O', ..., 'B-Modify', ..., 'B-Filling', 'O']
修正後文字	數位勞動蟻民
回傳	POST /error_correction/ → 200

表 12: Case Study: 系統 log (指令標註器)

結果如表12中「修正後文字」欄位所示，修正爲「數位勞動蟻民」，成功完成我們的語音修正任務。透過蒐集每個模型預測的 log 資料，也能擴充我們現有資料集的多元性，對於未來的訓練有很大的幫助。

4.3 修正速度比較

爲了更全面地評估本系統在實際應用中的效益，本節將聚焦於「修正速度」的比較。我們關注的不僅是修正的正確性，也包括修正所需的時間成本。因此，我們設計了一項實驗，針對人爲使用鍵盤修正與透過語音指令修正兩種方式，進行平均耗時的比較。

我們從 SIGHAN-15 以及 zh-tw-wikipedia 的測試集中修正成功的案例中各挑選了 10 筆資料，共 20 筆來進行測試。每筆資料的內容包括:(1) 錯誤句子 (2) 正確句子 (3) 修正指令，如表 13 所示，錯誤句子中最後的「化」字，應該改成「花」。

項目	內容
錯誤句子	對不起，我不能參加你開的慶祝會，因爲我有事情。但是我一定送給你一把很大的化。
修正指令	請把化學的化改成花朵的花
修正結果	對不起，我不能參加你開的慶祝會，因爲我有事情。但是我一定送給你一把很大的花。

表 13: 修正速度測試資料範例

我們分別比較了兩種修正方式。第一種爲人工修正：使用者會看到每筆資料中的錯誤句子與正確修正結果，並透過滑鼠與鍵盤移動至錯誤位置進行編輯。爲聚焦於修正所需時間，我們事先標示出錯誤句與正確句之間的差異，使實驗參與者能直接定位錯誤並執行修正，無須額外花費時間尋找錯誤位置。

第二種方法爲本研究提出的語音指令修正系統。我們提供錯誤句子與對應的修正指令，並記錄從實驗參與者以語音講述指令，到系統完成修正並返回結果的總耗時。儘管部分情況下修正可能無法一次完成，本實驗僅考量成功修正所需的時間，並排除其他可能干擾準確性的因素。

實驗結果如表 14 所示，使用鍵盤輸入進行修正的平均耗時爲約 3 秒，而本研究所提出之語音修正系統的平均耗時爲約 5 秒。儘管語音修正在時間上略高，但差距相對有限，顯示本系統在修正效率上具備實用性與競爭力。

此外，在行動裝置等輸入條件受限的情境中，鍵盤輸入所需的操作時間可能進一步增加。對於不熟悉鍵盤輸入的使用者而言，透過語音指令進行修正可提供更直覺且便利的互動方式，進一步提升系統的可用性。

方法	平均修正時間 (秒)
人工修正	3
語音修正（Model-Mix）	5

表 14: 不同修正方式的平均時間比較

5 結論和未來展望

總結而言，本研究提出一套創新的語音辨識錯誤修正方法，透過口語指令的方式，提升修正彈性與人機互動的自然度。整體系統架構簡潔，僅需輸入分類器、指令分類器與指令標註器三個模組，皆可基於 BERT 系列模型完成訓練，有效降低開發與部署成本。實驗結果亦驗證了各模組的優異表現，展現本系統的實用性與擴展潛力。

此外，我們也利用現有的文字資料集，透過 TTS 以及 ASR 產生語音辨識的錯誤資料集，並建立不同敘述風格的指令類型。我們也以 API 的方式，提供我們的系統給其他具有語音辨識的服務使用，並將蒐集到的眞實資料整理成日後進一步提高模型效能的訓練資料。除此之外，大語言模型的加入也可以讓系統辨識出一些操作類型的指令，擴展使用語音辨識進行互動的靈活性。

儘管如此，系統仍存在若干限制。例如訓練資料多聚焦於單一詞語的單次修改，當句子中錯誤較多時，使用者需連續輸入多筆指令，可能影響修正效率；另一方面，若語音辨識本身輸出錯誤指令內容，亦可能影響最終的修正結果。針對此類問題，未來可嘗試導入其他自動錯誤修正技術，以提升整體系統的穩定性。

針對多語言環境的應用，若能針對各語言的常見辨識錯誤與語句特性，建立專屬的訓練資

料集，則本系統架構亦有望擴展至其他語種，實現跨語言的語音錯誤修正能力。

展望未來，隨著語音互動技術的持續普及，使用者逐漸由鍵盤輸入轉向語音控制。如何自然且有效地修正語音辨識錯誤，將成爲關鍵挑戰之一。本研究所提出之系統，提供語音操作使用者一種新穎且直覺的互動方式，未來在教育、醫療、智慧助理等多元場域皆具備廣泛應用潛力。

參考資料

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Ling Dong, Wenjun Wang, Zhengtao Yu, Yuxin Huang, Junjun Guo, and Guojiang Zhou. 2024. Pronunciation guided copy and correction model for asr error correction. *International Journal of Machine Learning and Cybernetics*, 15(10):4787–4799.

Jin Jiang, Xiaojun Wan, Wei Peng, Rongjun Li, Jingyuan Yang, and Yanquan Zhou. 2024a. Cross modal training for asr error correction with contrastive learning. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12246–12250.

Lai Jiang, Hongqiu Wu, Hai Zhao, and Min Zhang. 2024b. Chinese spelling corrector is just a language learner. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6933–6943.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, pages 707–710. Soviet Union.

Yinghui Li, Shang Qin, Haojing Huang, Yangning Li, Libo Qin, Xuming Hu, Wenhao Jiang, Hai-Tao Zheng, and Philip S Yu. 2024. Rethinking the roles of large language models in chinese grammatical error correction. *arXiv preprint arXiv:2402.11420*.

Yuang Li, Xiaosong Qiao, Xiaofeng Zhao, Huan Zhao, Wei Tang, Min Zhang, and Hao Yang. 2025. Large language model should understand pinyin for chinese asr error correction. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Linfeng Liu, Hongqiu Wu, and Hai Zhao. 2024. Chinese spelling correction as rephrasing language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18662–18670.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Victor Junqiu Wei, Weicheng Wang, Di Jiang, Yuanfeng Song, and Lu Wang. 2024. Asr-ec benchmark: Evaluating large language models on chinese asr error correction. *arXiv preprint arXiv:2412.03075*.

Kai-Tuo Xu, Feng-Long Xie, Xu Tang, and Yao Hu. 2025. Fireredasr: Open-source industrial-grade mandarin speech recognition models from encoder-decoder to llm integration. *arXiv preprint arXiv:2501.14350*.

Chao-Han Huck Yang, Yile Gu, Yi-Chieh Liu, Shalini Ghosh, Ivan Bulyko, and Andreas Stolcke. 2023. Generative speech recognition error correction with large language models and task-activating prompting. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Shuai Zhang, Jiangyan Yi, Zhengkun Tian, Ye Bai, Jianhua Tao, Xuefei Liu, and Zhengqi Wen. 2021. End-to-end spelling correction conditioned on acoustic feature for code-switching speech recognition. In *Interspeech*, pages 266–270.

6 附錄

6.1 指令生成提示詞

```
你是一個文字校對助手，專門協助修改語音辨識錯誤的句子。我會提供「正確答案」和「題目」，你需要：
1. 比較「正確答案」和「題目」，根據規則決定是否需要修改。
   - 如果兩句意義相同，告訴我「不需修改」。
   - 如果有需要修改的地方，列出所有修改指令，每次只修改一個字。
2. 指令需考慮到用語音辨識輸入時的可辨識性，且包含：
   - 指令類型（替換）。
   - 修改位置（「題目」中的第幾個字開始）。
   - 被修改的字描述（使用常用詞語，如「洗碗的洗」或「希望的希」）。
   - 替換字描述（使用常用詞語，如「辛苦的辛」）。
   - 修改後的題目和答案。
3. 修改規則：
   - 忽略標點符號的錯誤。
   - 如果字不同但不影響意思，不算錯字。
   - 比較數字時，統一轉爲阿拉伯數字。
4. 如果有多處需要修改，請列出所有修改指令，並保持順序，一次一個字。

#### 範例：
正確答案：我眞的希望我可以去看你。
題目：我眞的洗碗我可以去康你。
指令：
1. 請把『洗碗的洗』改成『希望的希』。
   指令類型：替換
   修改位置：第4個字
   填入字：希
   題目：我眞的{洗}碗我可以去康你。
   答案：我眞的希碗我可以去康你。
2. 請把『洗碗的碗』改成『希望的望』。
   指令類型：替換
   修改位置：第5個字
   填入字：望
   題目：我眞的希{碗}我可以去康你。
   答案：我眞的希望我可以去康你。
3. 請把『健康的康』改成『看見的看』。
   指令類型：替換
   修改位置：第11個字
   填入字：看
   題目：我眞的希望我可以去{康}你。
   答案：我眞的希望我可以去看你。
```