國立台北大學資訊工程學系專題報告
Transformer-based 注音輸入法

專題組員： 劉晉嘉、洪紹軒、陳毓霖

專題編號： PRJ-CSIE-114-002

執行期間： 2025 年 7 月至 2026 年 5 月

摘要：本專題以繁體中文注音輸入法為應用場景，訓練一個專門處理「語境式同音字選擇」的 decoder-only LLaMA-style Transformer。傳統注音輸入法多依賴字詞表、語言模型分數或 n-gram 類型的局部統計，在長距離語境、口語化句子與同音字密集的情境中，候選字排序容易偏離使用者意圖。本研究採用受限候選字排序策略，由候選字表提供注音合法集合，再由模型依照前後文重新排列候選順序，藉此改善第一候選與前 k 名候選字品質。

模型訓練分為三個階段：第一階段自建 token table，使用大規模繁體中文語料進行下一個 token 預訓練，使模型取得基本中文語境能力；第二階段加入注音與國字對照任務，透過一字一 token 的詞表設計，讓注音 token 與中文字 token 進入同一表示空間；第三階段以接近輸入法部署格式的資料進行多任務微調，使模型習慣「前文 + 注音序列 + 預測字序列」的推論格式。最終模型約 247.7M 參數，上下文長度為 384，部署時以候選字 mask 約束輸出分數，只在合法候選字中排序。初步延遲測試顯示，常見短輸入情境可維持互動式回應，驗證此訓練方向具備實際輸入法部署潛力。

1. 簡介

注音輸入法需要在同音候選集合中選出符合語境的字。例如「ㄧㄠ ㄟ ㄨㄢ ㄏㄠ」對應到「要玩」與「藥丸」等候選；在「你今天有」這類前文後面，口語語境會明顯影響候選字排序。當排序只依賴音節與短詞頻率時，使用者往往需要頻繁按方向鍵或數字鍵修正，打斷輸入節奏。

本專題的目標是建立一套可在一般 Windows 桌面環境中實際使用的開源注音輸入法原型，並將離線模型訓練成果接上真實輸入流程。前端負責符合 Windows 輸入法習慣的操作體驗，包含 TSF 組字區、候選字視窗、標點快捷鍵與游標前文擷取；後端負責在輸入過程中維持低延遲推論；候選字表則負責限制模型輸出，確保排序結果仍符合注音讀音。

2. 模型與任務設計

模型定位為繁體中文注音輸入法專用的 decoder-only LLaMA-style Transformer。輸入法選字可表述為自回歸的下一 token 預測：模型讀取已存在的語境與目前輸入的注音序列，逐步預測最可能的中文字。這個形式符合因果注意力的計算方式，也能直接利用成熟的 LLaMA 訓練與推論工具鏈。

模型卡紀錄的主要規格為：參數量約 247.7M、上下文長度 384、20 層、16 個注意力頭、隱藏層維度 1024，使用自訂輸入法詞表，包含繁體中文字 token、注音 token、英文備援 token 與特殊 token。此規模刻意控制在可本機部署的範圍內，避免輸入法因模型過大而犧牲互動延遲。

推論提示格式如下：

<BOS> 前文字元... 注音 token ... <SEP> 預測字元...

前文提供語境，注音 token 描述目前待確定的音節序列，<SEP> 之後則是模型逐步選出的中文字。實際輸入法先由注音候選字對應表查出該注音的合法候選字，再將候選字映射到 token id，對輸出分數做候選字 mask 與正規化，最後只在合法候選集合內排序。此流程同時保留語境判斷能力與輸入法的音韻約束。

後端推論另實作快取對齊：每次請求會重新將前文與待確定音節轉成 token 序列，並與上一輪 token 序列比較共同前綴。前綴相同時只解碼新增 token；前綴縮短或改變時，系統會移除模型記憶中不再相符的序列片段。這樣的增量處理方式適合逐鍵輸入場景。

3. 訓練流程

本專題的訓練流程分成三個階段。第一階段建立繁體中文語境能力；第二階段建立注音 token 與中文字 token 的對應關係；第三階段讓模型熟悉實際輸入法會使用的提示格式。這樣的安排把語言建模、讀音對齊與部署格式適應分開處理，也讓後續錯誤分析更容易定位。

三個階段分別對應輸入法模型需要的核心能力。中文語境預訓練負責建立常見搭配、語氣與語意判斷；注音－國字對齊讓 < ㄧㄠ ㄟ> 這類 token 與「要、藥、耀」等候選字產生讀音關係；輸入法格式微調則讓模型習慣「前文 + 多個待確定注音 + 已預測字」的推論狀態。這些能力合在

一起，才構成可部署的候選字排序模型。

3.1 第一階段：中文語境預訓練

第一階段開始訓練自有模型，目標是讓模型學會繁體中文本身的機率分布，也就是在沒有讀音提示時估計 P(下一個字 | 前文)。同音字排序需要中文語境支撐，包含常見搭配、口語語氣、動賓關係、專有名詞前後綴等訊號。這些語言先驗會成為後續注音條件發揮作用的基礎。

此階段同時確立 token table 設計。一般大型語言模型常用 BPE 或 sentencepiece，把常見詞片段合併成 token；輸入法推論需要對單一候選國字做 mask，因此本專題採用一字一 token 的中文字詞表，讓每個候選字都能直接對應到一個輸出分數位置。英數、罕見字與未定義內容則透過 <LATIN>、<NUM>、<UNK> 等備援 token 處理。這個設計犧牲部分壓縮效率，換來輸入法所需的可控性與候選字 mask 精確性。

資料以大規模繁體中文文本為主，任務為標準下一 token 預測。原始計畫中此階段資料量約 5B tokens，用於讓模型建立中文語境能力。此階段的產物可視為「繁體中文逐字語言模型」，負責提供後續所有同音字判斷所需的語言先驗。

3.2 第二階段：注音與國字共同表示

第二階段處理中文文字與注音讀音的表徵落差。模型經過中文語境預訓練後已經掌握文字分布，接著需要學會注音 token 的意義。< ㄧㄠˋ> 代表一組同音候選，訓練目標是讓注音 token 與中文字 token 在模型內部形成可互相條件化的表示，使模型看到注音時能把機率集中到對應讀音的候選字集合附近。

資料來源包含 jieba 分詞後的文本、教育部國語辭典簡編本等注音對照資料，並在對照失敗時退回讀音頻率標註。分詞在此用於取得多音字與詞內讀音的合理標註單位。例如同一個字在不同詞中可能有不同讀音，單字表標註容易造成多音字錯誤；透過詞級資訊與字音轉換/頻率退回機制，可降低注音標註雜訊。

3.3 第三階段：輸入法格式微調

第三階段處理部署格式。實際輸入法推論時，模型輸入由「游標前文 + 待確定注音序列 + 已經逐步選出的字」組成。前半段是已提交文字，中間是注音 token，<SEP> 後面則是模型前幾步已選出的中文字。此階段透過格式微調，讓模型穩定區分注音 token、前文與輸出區段。

資料使用基於 BERT 的國字轉注音模型 G2PW 進行標註，約 0.35B tokens，並混合約 30% 第二階段資料進行多任務微調。G2PW 為大規模自然文本產生更接近上下文的讀音標註，特別是多音字；第二階段資料則用來維持基礎注音－國字對齊能力。這個多任務設計同時處理任務適應與能力保留。

微調後的提示格式為：

<BOS> 前文字元... 注音 token ... <SEP> 預測字元...

在輸入法中，每次完成一個可組字音節後，系統會把游標前文與目前組字緩衝區內的待確定注音傳給模型。模型對每個位置產生輸出分數，再由候選字表做 mask。模型負責排序，候選字表負責合法性；此分工能避免模型輸出不存在於該注音的字，也讓輸入法保有傳統候選框架的穩定性。

訓練時要求模型在 <SEP> 之後逐字輸出答案；部署時採用每個音節位置的輸出分數，套用候選字 mask 後排序。訓練讓模型學會語境與讀音如何共同決定答案，部署端再用 mask 保證輸入法不會產生非法候選。模型在整體系統將候選表中的每個字重新排序，供使用者選擇。

4. 系統整合與部署

系統實作主要承載訓練成果並驗證互動式輸入可行性。前端以 Windows TSF/COM 實作，負責注音鍵盤映射、組字緩衝區、候選字介面與前文擷取；後端為常駐服務程式，啟動時載入模型，透過 named pipe 接收前文與待確定注音。模型以 GGUF/Q4_K_M 形式部署到 llama.cpp，主要目的是降低本機推論延遲與記憶體需求。

5. 主要成果與評估

本專題的主要成果是完成一套面向注音輸入法任務的模型訓練流程，並將訓練後模型接上可實際使用的 Windows 輸入法原型以驗證部署可行性。目前量化結果聚焦在「推論與管線通訊延遲」，準確率評估仍需建立更完整的測試集。

延遲結果顯示，主要成本會隨待確定音節數量增加而上升，前文長度在快取命中良好時影響較可控。對日常輸入而言，1 至 4 個待選音節多落在十數到二十多毫秒範圍，符合互動式輸入的需求；極端長前文與 16 個待確定音節則會上升到百毫秒等級，未來需透過更細的增量更新、批次策略或候選數量裁切改善。

本系統適合改善同音詞或口語短句選字，例如「你今天有要玩嗎」、「可以幫我畫成圖表嗎」、「等一下一起吃飯嗎」等。這些例子僅作為行為觀察，說明候選字 mask 加語境模型的設計確實對注音輸入的痛點有直接對應。

Table 1: 模型訓練階段摘要

階段	目標	重點
1	中文語境預訓練	以自訂一字一 token 的詞表訓練繁體中文下一 token 模型，使輸出分數能直接對應單字候選，並建立語意、搭配與口語先驗。
2	注音－國字對齊	透過注音標註資料讓模型學會讀音 token 與中文字 token 的共同表示，補上一般中文預訓練較少涵蓋的音韻條件。
3	輸入法格式微調	使用實際部署格式進行多任務微調，使模型適應前文、待確定注音序列與逐步輸出字元混合出現的推論狀態。

Table 2: 暖機後服務延遲測試摘要（30 筆樣本；僅統計有效樣本）

情境	待確定音節	平均	p95
短輸入、單音節	1	8.41	12.89
短輸入、4 音節	4	20.88	22.85
短前文、8 音節	8	40.40	49.34
中等前文、12 音節	12	63.91	75.75
長前文、16 音節	16	130.10	174.97

6. 結語與展望

本專題完成了從語料處理、token table 設計、分階段訓練到輸入法格式微調的模型流程，並以 Windows 輸入法原型驗證其實際互動可能性。

後續工作包含：

1. 建立正式準確率資料集，分別評估第一候選、前 k 名候選、長前文、口語句與專有名詞情境。
2. 改善增量推論，使游標移動、刪字與候選字改選時能更精準重用 KV 快取。
3. 研究 NPU、DirectML 或更小量化格式，以降低純 CPU 部署延遲與耗電。
4. 加入本地個人化機制，並以隱私優先方式設計，因本模型的大小非常小，所以使用者可以在本地完成 LoRA 微調來從根本個人化輸入法模型。
5. 強化退回策略，讓零前文、罕見字、英文與符號輸入在模型信心不足時仍維持穩定體驗。

7. 銘謝

感謝指導老師在題目發想與系統實作方向上的建議，也感謝組員在模型、資料處理、Windows TSF、部署與測試上的分工合作。本專題同時受惠於 llama.cpp、utf8cpp、Asio、vcpkg 以及多個開源繁體中文資料與工具專案；這些基礎建設讓大型語言模型整合進桌面輸入法成為可行方向。

8. 參考文獻

[1] A. Vaswani et al., “Attention Is All You Need,” Advances in Neural Information Processing Systems, 2017.

[2] Meta AI, “LLaMA: Open and Efficient Foundation Language Models,” 2023.

[3] ggml-org, “llama.cpp and GGUF model format,” https://github.com/ggml-org/llama.cpp.

[4] Microsoft Learn, “Text Services Framework,” Windows application development documentation.

[5] McBopomofo project notes, “注音輸入法現況與 DAG 相關介紹,” https://mcbopomofo.substack.com/p/dag.

[6] Bopomofo IME LLaMA 250M model card and GGUF deployment notes, project repository artifacts.