Would LLMs be Good Historical Linguists and Chinese Dialect Learners?

Yicheng Liu **Shumin Shi*** **Youchao Zhou** **Xingchen Zhang**
School of Computer Science & Technology, Beijing Institute of Technology
Beijing, China
{lyc2024, bjssm}@bit.edu.cn

Abstract

Large language models (LLMs) perform well on Standard Chinese but struggle with low-resource Chinese dialects due to substantial phonological divergence. We investigate whether incorporating *Middle Chinese*, the common historical ancestor of most of the modern Chinese dialects, can improve dialectal pronunciation modeling in a linguistically interpretable manner. We focus on two specific task variants: (1) *conditional sound change rule induction* (a variant of Sound Law Induction, SLI), where models infer executable phonological transformation rules from Middle Chinese to modern dialects, and (2) *sentence-level dialectal pronunciation transcription* (a variant of Grapheme-to-Phoneme, G2P), requiring dialect-specific International Phonetic Alphabet (IPA) generation. We construct a multi-source dataset covering Middle Chinese and 12 modern Chinese dialects, including character-level correspondences, rule exemplars, and sentence-level IPA transcription. We adopt a parameter-efficient training framework combining LoRA-based supervised fine-tuning and reinforcement learning via Group Relative Policy Optimization (GRPO) for the first task. Across both tasks and a wide range of dialects and evaluation metrics, our approach achieves overall improvements over strong baselines, including DeepSeek-V3.2 and ChatGPT-5.2, while revealing variation across dialects. These results demonstrate the value of leveraging historical linguistic knowledge for modeling low-resource Chinese dialects.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable performance across a variety of natural language processing tasks, showcasing strong capabilities in language understanding and generation. Multimodal LLMs such as Qwen 3 Omni (Xu et al., 2025) even support speech synthesis and comprehension in several major Chinese dialects, including Cantonese and Sichuanese. Despite these advances, there are 17 types of Chinese dialects[1] and 101 dialect subgroups (Xiong et al., 2012), many of which are low-resource and even endangered, and LLMs still struggle to model their phonological systems and pronunciations reliably.

Although these dialects share a common logographic writing system, i.e. Chinese characters, the phonological divergence across the Chinese family is substantial. Crucially, characters that are homophonous in Standard Chinese may be distinct in other dialects, and vice versa. For example, the characters 搬 (meaning *to move*) and 斑 (meaning *spotted*) are homophonous in Standard Chinese (both pronounced as /pan^{55}/) but not in Cantonese (pronounced as /pun^{55}/ and /pan^{55}/, respectively), whereas 酸 (meaning *sour*) and 宣 (meaning *to declare*) are homophonous in Cantonese (both pronounced as /syn^{55}/) but not in Standard Chinese (pronounced as /suan^{55}/ and /ɕyan^{55}/, respectively). Such asymmetries make it difficult to directly project phonological information from Standard Chinese to the other Chinese dialects. In practice, LLMs frequently hallucinate dialectal pronunciations due to an overreliance on Standard Chinese as a proxy for all Chinese dialects.

A primary source of these divergences lies in the historical evolution from *Middle Chinese* to different modern Chinese dialects. As shown in Figure 1, despite the different evolutionary paths, both Standard Chinese and Cantonese pronunciations can be inferred from the phonological system of Middle Chinese according to certain linguistic rules.

Motivated by these historical linguistic rules, we

* Corresponding author.

[1] There are two perspectives in linguistics regarding Chinese: one views it as a language family, with its various variants considered distinct languages; the other views Chinese as a single language, treating its variants as different dialects. This paper adopts the latter perspective, which is more widely accepted by the general Chinese public.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 18231–18256

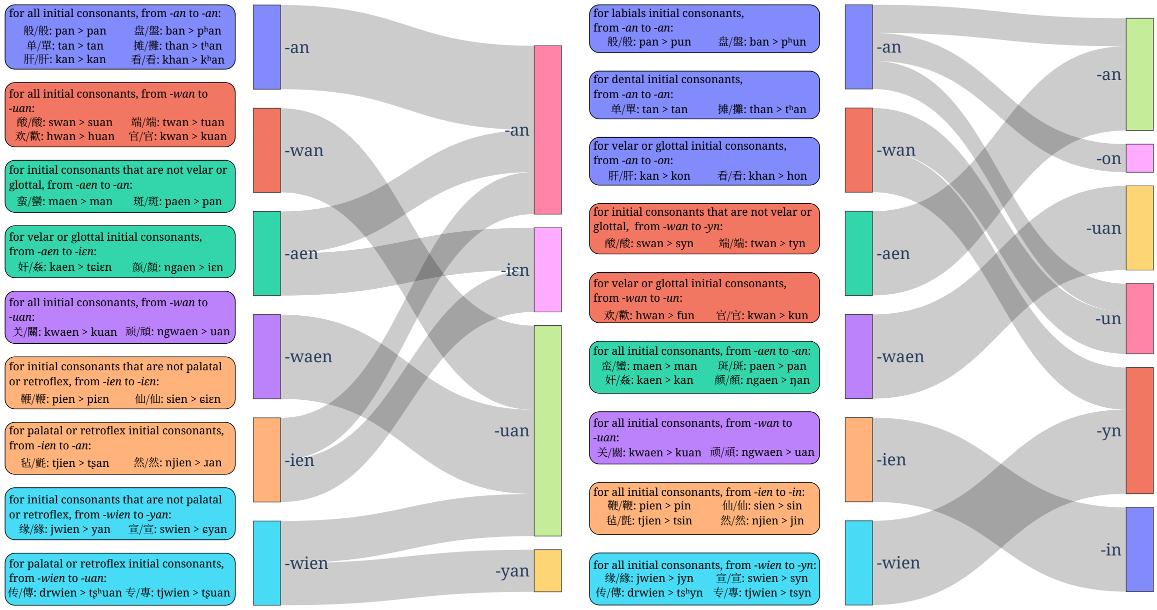

Figure 1: Example of the different sound change paths of some Middle Chinese characters in the "*shan* rhyme group" (山摄/山攝, *shan she*, where "*she*" or "rhyme group" refers to a set of characters with similar finals) to Standard Chinese and Cantonese in Guangzhou. (In this paper, the style "Chinese scripts A/Chinese scripts B" denotes "Simplified Chinese/Traditional Chinese.")

investigate whether LLMs can leverage Middle Chinese as an explicit intermediate representation to infer sound change rules and generate modern dialectal pronunciations.

To achieve this, we propose a systematic pipeline comprising data construction and two-stage model training (as intuitively summarized in Figure 2). We first build an automated alignment and polyphonic normalization pipeline to clean and enhance the existing datasets MCPDict and Polyphone. Building on this multi-source data, we adopt a parameter-efficient training framework. It begins with LoRA-based supervised fine-tuning to equip the model with basic historical reasoning and surface generation skills, followed by reinforcement learning via Group Relative Policy Optimization (GRPO). The RL reward is carefully designed to optimize the generated Python rules.

Our contributions are summarized as follows:

- We formulate two specific task variants of Sound Law Induction (SLI) and Grapheme-to-Phoneme (G2P) that operationalize historical sound change in LLMs: (i) *conditional sound change rule induction* from Middle Chinese to modern dialects, producing executable phonological rules; and (ii) *sentence-level dialectal pronunciation transcription*, requiring dialect-specific IPA generation.
- We present an automated normalization and alignment pipeline for polyphonic characters to clean the existing MCPDict database, yielding a multi-source dataset spanning Middle Chinese and 12 modern dialects.
- We propose a parameter-efficient, linguistically informed training framework combining LoRA-based supervised fine-tuning and GRPO-based reinforcement learning (only for the SLI task), achieving overall improvements over strong LLM baselines.

2 Related Work

2.1 Machine Learning and Historical Linguistics

Sims-Williams (2018) proposed the concepts of Computerized Backward Reconstruction (CBR) and Computerized Forward Reconstruction (CFR), which refer to using computers to reconstruct historical languages from modern languages and vice versa.

In the CBR task, Meloni et al. (2021) first introduced neural models into the task of historical language reconstruction, using a GRU model to reconstruct Latin from modern Romance languages. Extending this approach to the Sinitic family, Chang et al. (2022) constructed WikiHan, a comparative dataset standardizing transcriptions into IPA, and applied an encoder-decoder architecture for protoform reconstruction. Kim et al. (2023) upgraded

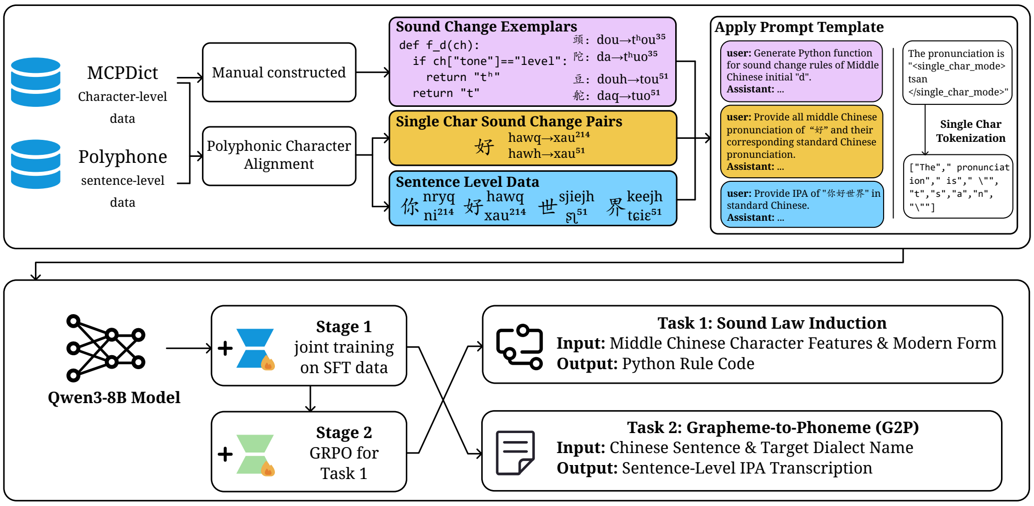

Figure 2: An overview of the pipeline proposed by this article.

the model from GRU to Transformer and conducted experiments on more languages from different language families. Huang et al. (2024) constructed the historical evolution chain of Chinese characters using a Transformer model, based on the pronunciations of different periods of Chinese and combining the glyph information of Chinese characters.

In the CFR task, Chacon and List (2016) modeled sound change patterns as directed weighted transitions to infer phylogenetic trees for Tukanoan languages. Luo (2021) proposed a reinforcement learning method based on hierarchical Monte Carlo tree search for automatically inducing sound change rules in multiple language families. Chang et al. (2023) trained a neural network to estimate pronunciation distances between phonemes and predict intermediate phonetic change steps between historical forms and their descendants.

Naik et al. (2025) adopted an SFT + programming by examples (PBE) approach to apply large language models to the task of inducing sound change rules for the first time. Unlike their work, which adds example code to the prompt context, we internalize code knowledge directly during training.

2.2 Large Language Models and Dialects

Lin et al. (2025) evaluated the reasoning performance of LLMs on the English variant AAVE, arguing that most mainstream LLMs exhibit significant vulnerability and unfairness to nonstandard dialectal inputs. Liu et al. (2023) introduced adapters to adapt large language models of standard English to different English dialect variants. Dimakis et al. (2025) proposed a method combining rule-based few-shot prompting to "standardize" dialect texts for downstream NLP systems. Liang et al. (2025) expanded upon a manually annotated dataset using LLMs to generate a bilingual dataset of Colloquial Singaporean English and standard English with explainability and fine-grained control. Jiang et al. (2025) proposed new evaluation benchmarks to test the performance of LLMs in Cantonese understanding and generation tasks.

Despite the rich diversity of research on LLMs and dialects, there remains a lack of work from a historical linguistics perspective. This highlights the necessity of our approach, which bridges LLMs with historical sound change patterns for rule induction and pronunciation prediction.

3 Task Definition

3.1 Task 1: Conditional Sound Change Rule Induction

Task 1 is a variant of SLI that requires Middle Chinese phonetic information, which evaluates whether a model can induce explicit and executable conditional sound change rules. For each character, the model receives a Middle Chinese phonological representation (a dictionary of discrete features; see Appendix A) and outputs the predicted initial, final, or tone in a target dialect d.

Concretely, given a phonological feature pair $(\mathcal{F}_1, \mathcal{F}_2)$ ((Middle Chinese initial, modern dialect initial), (Middle Chinese rhyme, modern dialect final), or (Middle Chinese tone, modern dialect tone)), a specific value v, and a set of example character pronunciation pairs L from Middle Chinese to the modern dialect satisfying $\mathcal{F}_1 = v$, the model needs to generate a rule to infer $\mathcal{F}_2$ in the modern dialect for all characters satisfying $\mathcal{F}_1 = v$.

Each rule is expressed as an executable Python function:

```
def rule(char_info):
    ...
```

which returns a single determined form or a list of possible forms for random phonemic split.

Optionally, to account for cross-component interactions, the model may generate a revision function:

```
def revise(char_info, modern_syllable):
    ...
```

Here, `modern_syllable` is a dictionary containing the initial, final, and tone of the character in the target dialect. The predicted component in this dictionary is taken from the output of the rule function `rule`, while the remaining components correspond to the character's true dialectal pronunciation, preventing the model from bypassing rule induction by directly returning the gold label.

3.2 Task 2: Dialectal Pronunciation Prediction

Task 2 evaluates sentence-level pronunciation modeling. Given a target dialect d and a Chinese sentence s, the model outputs an IPA transcription of s in dialect d, evaluated using Levenshtein distance.

For characters not in the dialect lexicon, evaluation labels are obtained by applying Task 1 rules to their Middle Chinese representations.

4 Dataset Construction

4.1 Data Sources

The data used in this work are primarily built upon two open-source resources. For character-level phonetic attributes and dialectal pronunciations, we utilize MCPDict,[2] an open-source, continuously maintained database of Chinese character pronunciations across historical periods and modern varieties. For Middle Chinese (MC), MCPDict adopts the phonological system of *Guangyun*, an 11th-century dictionary widely used in Chinese historical phonology. The dataset includes 25,335 original character entries.

For modern Chinese dialects, we select a representative set of 12 varieties covering major Mandarin subgroups and non-Mandarin branches from MCPDict. The selected dialects and their raw data sizes are summarized in Table 1. In this context, a

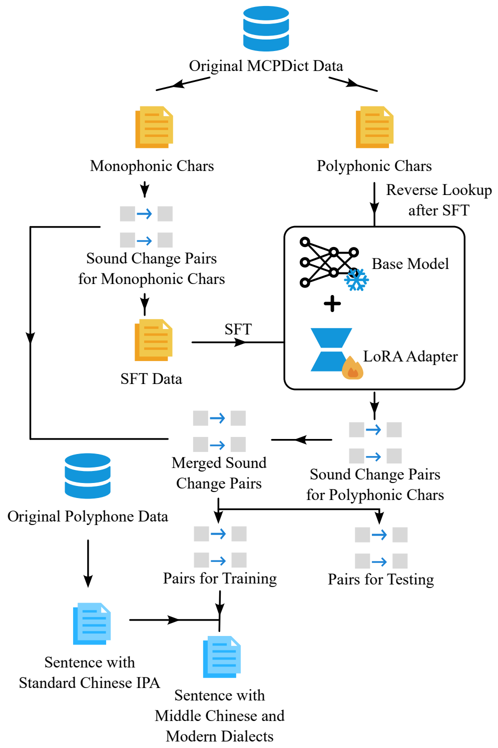

Figure 3: Construction workflow for Character-level Sound Change and Sentence-level Pronunciation Data.

monophonic character possesses exactly one pronunciation in a given linguistic system, whereas a *polyphonic* character exhibits multiple pronunciations.

For evaluating sentence-level dialectal pronunciation prediction (Task 2), we make use of a subset of the Polyphone dataset (Zhang et al., 2023),[3] which provides sentences from diverse domains specifically designed for disambiguating Mandarin polyphonic characters in context.

4.2 Structured Base Data Construction

Based on the aforementioned sources, we construct three categories of **Structured Base Data** (examples are provided in Appendix C).

Category 1: Sound Change Rule Exemplars To enable the model to generate executable sound change rules (Task 1), the authors manually constructed a set of rule exemplars for four representative dialects: Standard Mandarin, Chengdu, Shanghai, and Guangzhou. Relying on MCPDict statistics and historical linguistic knowledge, these exemplars map phonological conditions in MC to modern dialects. Each exemplar provides a generated Python script, specific linguistic rules, illustrative character

[2] https://github.com/osfans/MCPDict

[3] https://github.com/NewZsh/polyphone

Dialect	Dialect Group	#Raw Entries	#Monophonic Pairs	#Polyphonic Pairs
Putonghua	Standard Chinese	99,851	12,138	1,918
Harbin	Northeastern Mandarin	6,126	3,105	1,102
Chengdu	Southwestern Mandarin	3,094	1,953	680
Luoyang	Central Plains Mandarin	3,942	2,417	853
Jinan	Ji–Lu Mandarin	4,395	2,574	874
Yinchuan	Lan–Yin Mandarin	4,937	2,930	1,018
Changsha	Xiang	4,373	2,509	855
Guangzhou	Yue	3,324	2,105	722
Shanghai	Wu	5,962	3,408	1,148
Nanchang	Gan	3,746	2,111	739
Meixian	Hakka	5,131	2,898	973
Quanzhou	Min	7,543	3,102	1,087

Table 1: Statistics of aligned Middle Chinese–dialect character pairs. *Raw Entries* denotes the original record count in MCPDict. *Monophonic Pairs* denotes occurrences where a character has exactly one pronunciation in both Middle Chinese and the respective dialect, whereas the remaining aligned character pairs are classified as *Polyphonic Pairs*. Note that counts reflect distinct characters, not distinct pronunciations.

examples, and explicit lists of irregular characters that the function should ignore.

Category 2: Character-level Sound Change Data For characters that are monophonic in both MC and the target dialect, we directly extract them to form explicit sound change pairs. However, for polyphonic characters, MCPDict does not explicitly annotate the exact evolutionary correspondence between specific MC variants and modern dialect variants. To establish these alignments, we implement a three-step pipeline: (1) We use the unambiguous monophonic pairs as anchors. (2) For each dialect, we automatically construct 100 multiple-choice questions requiring a model to select the correct MC origin from candidate pronunciations that differ in exactly one phonological dimension (initial, final, or tone). (3) We preliminarily fine-tune a base model on this diagnostic task (using the formatting pipeline described in Section 4.3) to predict MC origins for modern pronunciations. We then use this model to disambiguate all polyphonic entries, yielding a complete set of MC-dialect sound change pairs. During main training, we allocate 95% of these pairs for training and 5% for evaluation.

Category 3: Sentence-level Pronunciation Data Original sentences from the Polyphone dataset only provide annotated Standard Mandarin readings for polyphonic characters. To construct parallel dialectal sentences, we first use the `python-pinyin`[4] library to transcribe the remaining monophonic characters, obtaining a complete Standard Mandarin IPA transcription for each sentence. Subsequently, relying on the aligned character-level sound change pairs (Category 1), we performed a reverse lookup to find the MC representations of the characters in context. Finally, we project these contextualized MC representations forward to generate sentence-level IPAs for all modern dialects. This process yields 1,652 MC–Standard Mandarin training alignments and 428 MC–modern dialect training alignments. Additionally, we randomly sample 50 sentences from the Polyphone test set, which are shared across all dialects to serve as the test set for Task 2. The overall data construction workflow of Category 2 and 3 is illustrated in Figure 3.

4.3 Construction of Supervised Fine-Tuning Data

The raw Structured Base Data constructed above cannot be directly utilized for large language model alignment. To prepare the final training set, we apply specifically designed instruction templates (detailed in Appendix D) to the three categories of Structured Base Data, transforming them into standardized instruction-response pairs. This final collection is collectively referred to as the **Supervised Fine-Tuning Data**. All subsequent supervised fine-tuning directly operates on these templates to jointly optimize the model for both historical reasoning and dialectal IPA transcription.

[4] `https://github.com/mozillazg/python-pinyin`

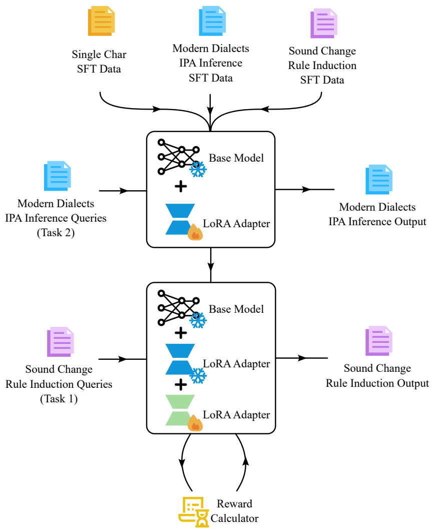

Figure 4: Workflow of our method using SFT and GRPO.

5 Method

5.1 Single Character Tokenization

In order to help the model better understand the relationships between phonemes, we adopt a single character tokenization strategy when encoding phonological components (e.g., IPA symbols and Middle Chinese phonetic categories) and input sentences in Task 2. Specifically, we wrap these parts with a <single_char_mode> tag and hook the original tokenizing function to force the tokenizer to process them character-by-character, while standard byte-pair encoding parses the rest. This preserves phoneme-level granularity (e.g., tokenizing "tsan" into "t", "s", "a", "n" instead of "ts", "an"), prevents unintended semantic combinations (e.g., tokenizing "侵入" as a single word meaning "invasion", whereas here the meaning is the rhyme "侵" and the tone "入"), and facilitates character-by-character analysis for sentence transcription.

5.2 Supervised Fine-tuning

We use supervised fine-tuning (SFT) + Low-Rank Adaptation (LoRA) (Hu et al., 2021) as the first stage of model training. Based on the Structured Base Data introduced in Section 4, we apply our predefined prompt templates to construct the final instruction–response pairs (Supervised Fine-Tuning Data). The detailed prompt format and examples can be found in Appendix D.

During supervised fine-tuning, we jointly train the model on data from Task 1 and Task 2. This joint training strategy allows the model to simultaneously acquire rule induction capabilities and surface pronunciation generation skills, facilitating knowledge sharing between the two tasks.

5.3 Reinforcement Learning with GRPO

For Task 1, after supervised fine-tuning, we further apply the GRPO (Shao et al., 2024) reinforcement learning algorithm to help the model find high-quality sound change rule functions within the search space.

The final reward score R for successfully executed rule functions is designed as follows:

$$R = Acc - P_{split} - P_{length}$$

where each term is defined as follows:

- **Accuracy Score** (Acc): For each test sample, if the executed rule function returns the exact correct unique answer, a score of 1 is given; if it returns a candidate list containing the correct answer, a score of 0.8 is assigned; otherwise, the score is 0. This partial reward of 0.8 discourages the model from lazily returning exhaustive candidate lists to avoid errors.
- **Random Split Penalty** (P_{split}): If a line in the rule function returns a candidate list of length k, the penalty for that line is defined as:

$$\alpha \left(\log_2 k + \sum_{i=1}^{k} p_i \log_2 p_i \right),$$

  where α represents the proportion of test samples covered by that line, and p_i is the empirical proportion of the true answers matching the i-th candidate. This penalty (maximized at $\alpha \log_2 k$ for uniform distributions) strongly encourages the model to deduce explicit sound change conditions rather than masking ignorance with uniform candidate lists.
- **Code Length Penalty** (P_{length}): Calculated as 0.001 times the character length of the uncommented rule code, to discourage overly verbose `if-else` structures and incentivize generalization.

Besides, if the model's generated response fails syntax checks or execution, we assign a penalty score of $R = -0.5$ to guide the generation of strictly executable Python code.

The training workflow is illustrated in Figure 4.

Model	Task 1: Conditional Sound Change Rule Induction															
	Standard Chinese				Guangzhou				Shanghai				Chengdu			
	UA	WA	US	WS	UA	WA	US	WS	UA	WA	US	WS	UA	WA	US	WS
Qwen3-8B	0.10	0.04	0.08	0.03	0.16	0.11	0.16	0.11	0.14	0.05	0.13	0.05	0.21	0.06	0.20	0.06
Our (Full)	**0.88**	**0.87**	**0.86**	**0.86**	0.84	0.79	**0.82**	**0.78**	0.84	0.81	**0.83**	**0.80**	0.89	0.89	**0.87**	**0.87**
Our (w/o GRPO)	0.74	0.73	0.72	0.72	0.82	0.75	0.80	0.73	0.81	0.73	0.80	0.72	0.85	0.84	0.83	0.82
Our (w/o SCT)	0.84	0.86	0.83	0.84	0.82	0.75	0.80	0.72	0.81	0.78	0.80	0.76	0.83	0.85	0.82	0.84
Our (w/o Joint Train)	0.74	0.68	0.70	0.65	0.80	0.66	0.77	0.63	0.80	0.78	0.77	0.73	0.81	0.75	0.79	0.74
DeepSeek-V3.2	0.81	0.85	0.62	0.61	0.83	0.81	0.71	0.68	0.80	0.66	0.67	0.53	0.84	0.77	0.71	0.49
ChatGPT-5.2	0.72	0.79	0.59	0.66	**0.87**	**0.86**	0.75	0.73	**0.88**	**0.88**	0.79	0.78	**0.93**	**0.90**	0.81	0.77
	Nanchang				Harbin				Meixian				Quanzhou			
Qwen3-8B	0.15	0.05	0.14	0.05	0.12	0.07	0.12	0.06	0.08	0.04	0.08	0.04	0.13	0.07	0.11	0.06
Our (Full)	0.83	0.82	**0.81**	**0.80**	0.83	**0.81**	**0.81**	**0.80**	0.79	**0.80**	**0.77**	**0.79**	0.82	0.85	**0.80**	**0.84**
Our (w/o GRPO)	0.78	0.71	0.75	0.68	0.77	0.73	0.74	0.70	0.69	0.58	0.67	0.57	0.77	0.75	0.75	0.73
Our (w/o SCT)	0.80	0.81	0.79	0.79	0.80	0.79	0.77	0.79	0.77	0.79	0.76	**0.79**	0.78	0.84	0.76	0.82
Our (w/o Joint Train)	0.79	0.70	0.76	0.68	0.73	0.63	0.70	0.58	0.72	0.64	0.70	0.62	0.79	0.79	0.77	0.78
DeepSeek-V3.2	0.80	0.77	0.71	0.69	0.77	0.65	0.65	0.55	0.79	0.67	0.67	0.56	0.80	0.69	0.69	0.58
ChatGPT-5.2	**0.89**	**0.85**	0.77	0.73	**0.88**	0.80	0.78	0.71	**0.83**	**0.80**	0.73	0.64	**0.87**	**0.87**	**0.80**	0.82
	Luoyang				Jinan				Yinchuan				Changsha			
Qwen3-8B	0.10	0.05	0.09	0.05	0.12	0.05	0.11	0.05	0.11	0.07	0.10	0.07	0.10	0.04	0.10	0.04
Our (Full)	0.86	0.85	**0.85**	**0.84**	0.84	0.72	**0.82**	0.71	0.85	**0.87**	**0.83**	**0.85**	0.78	0.82	0.77	**0.81**
Our (w/o GRPO)	0.78	0.79	0.77	0.77	0.78	0.68	0.75	0.65	0.77	0.78	0.74	0.75	0.77	0.76	0.74	0.74
Our (w/o SCT)	0.82	0.81	0.80	0.80	0.72	0.76	0.70	**0.75**	0.79	0.83	0.78	0.82	0.70	0.77	0.68	0.76
Our (w/o Joint Train)	0.78	0.73	0.76	0.71	0.76	0.69	0.74	0.67	0.73	0.70	0.70	0.68	0.69	0.68	0.67	0.65
DeepSeek-V3.2	0.80	0.82	0.66	0.58	0.79	0.65	0.69	0.53	0.81	0.63	0.70	0.54	0.81	0.61	0.69	0.51
ChatGPT-5.2	**0.90**	**0.86**	0.79	0.77	**0.91**	**0.86**	0.80	0.74	**0.90**	**0.87**	0.78	0.77	**0.88**	**0.84**	**0.78**	0.67

Table 2: Task 1 results on conditional sound change rule induction. UA / WA denote unweighted and weighted accuracy, while US / WS denote unweighted and weighted score. All data are the results of a single run.

6 Experiments

6.1 Experimental Setup

6.1.1 Implementations

We adopt Qwen3-8B (Yang et al., 2025) as the backbone. Model training is implemented using the Hugging Face ecosystem, including `Trainer`, `Accelerate`, and `TRL`. More details about the training configurations are provided in Appendix B.

6.1.2 Ablation Setup

To understand the contribution of different components, we designate **Our (Full)** as the complete method described in Section 5. Based on this, we evaluate task-specific ablation settings. For Task 1, we evaluate **Our (w/o GRPO)** (using only SFT without reinforcement learning), **Our (w/o SCT)** (disabling single-character tokenization in favor of standard BBPE), and **Our (w/o Joint Train)** (training exclusively on rule exemplars without using character-level pronunciation pairs). For Task 2, we evaluate **Our (w/o MC)** (omitting all Middle Chinese information and only training with modern pronunciation) and **Our (w/o SCT)**.

6.2 Baselines

To comprehensively evaluate the effectiveness of the proposed approach, we compare our models against the following baselines:

- **Qwen3-8B (Base)**: the original Qwen3-8B model without any fine-tuning, serving as a reference for the backbone's out-of-the-box performance.
- **DeepSeek-V3.2**(DeepSeek-AI et al., 2025): a strong open-source large language model, accessed via its official API.
- **ChatGPT-5.2**(OpenAI, 2025): a closed-source general-purpose large language model, accessed via its official API.

We exclude previous methods for related tasks (e.g., Luo (2021) and Naik et al. (2025)) from our baselines because their approaches cannot accommodate the explicit Middle Chinese phonological representations required by our task formulation.

All baseline models are evaluated on the same test sets at the level of Structured Base Data. For prompt formats, we provide more detailed information to the baseline models to further clarify the task requirements to mitigate evaluation mismatches for semantically equivalent model outputs.

	Task 2: Sentence-to-Dialect IPA Generation															
	Standard Chinese				Guangzhou				Shanghai				Chengdu			
Model	Full	Train	Test	Pred.	Full	Train	Test	Pred.	Full	Train	Test	Pred.	Full	Train	Test	Pred.
Qwen3-8B	10.62	3.36	3.23	3.50	29.11	3.81	3.51	3.79	33.43	3.62	3.54	3.51	22.90	3.80	3.96	3.56
Our (Full)	**0.10**	**0.02**	0.03	1.50	**0.48**	**0.10**	0.22	**0.93**	**0.69**	**0.08**	**0.11**	1.82	**0.68**	**0.12**	**0.01**	**1.26**
Our (w/o MC)	**0.10**	0.03	0.03	1.92	3.49	0.54	0.59	1.24	1.21	0.57	0.75	2.04	0.80	0.23	0.22	1.42
Our (w/o SCT)	**0.10**	**0.02**	**0.00**	1.45	0.64	0.25	0.20	1.11	0.80	0.22	0.51	**1.50**	0.78	0.17	0.07	**1.26**
DeepSeek-V3.2	0.28	0.09	0.03	**1.43**	0.81	0.38	**0.03**	1.07	1.37	0.59	0.36	2.10	0.98	0.39	0.45	1.51
ChatGPT-5.2	0.21	0.13	0.11	1.60	0.72	0.31	0.06	1.08	3.47	2.64	3.03	3.03	2.11	1.69	1.18	2.46
	Nanchang				Harbin				Meixian				Quanzhou			
Qwen3-8B	39.16	6.30	4.65	3.80	18.67	3.62	3.66	3.47	29.65	3.93	4.03	3.94	28.73	4.74	3.95	4.25
Our (Full)	**0.62**	**0.14**	**0.06**	**1.47**	**0.32**	**0.07**	**0.07**	**0.87**	**0.48**	**0.13**	**0.05**	**0.85**	**0.37**	**0.09**	**0.16**	**1.25**
Our (w/o MC)	0.87	0.39	0.60	1.93	0.38	0.15	0.16	1.03	0.76	0.40	0.54	1.20	0.72	0.41	1.00	1.74
Our (w/o SCT)	0.80	0.30	0.47	1.62	0.45	0.13	0.19	0.98	0.64	0.23	0.64	1.11	0.73	0.35	1.09	1.41
DeepSeek-V3.2	1.41	0.93	1.00	1.48	0.73	0.48	0.61	1.11	0.83	0.23	0.47	0.95	1.40	1.16	0.86	2.16
ChatGPT-5.2	4.92	4.25	4.59	4.08	1.44	0.86	0.83	1.24	2.13	1.71	2.05	1.80	2.93	2.40	2.28	2.81
	Luoyang				Jinan				Yinchuan				Changsha			
Qwen3-8B	29.46	3.55	3.46	3.47	20.57	3.84	4.23	3.86	22.98	3.39	3.92	3.23	43.80	4.25	4.17	4.29
Our (Full)	**0.46**	**0.06**	**0.16**	**1.00**	**0.46**	**0.06**	**0.13**	2.30	**0.53**	**0.08**	**0.06**	0.87	**0.38**	**0.05**	**0.01**	**1.44**
Our (w/o MC)	0.57	0.17	0.53	1.36	0.53	0.14	0.32	2.47	**0.53**	0.12	0.14	**0.75**	0.60	0.26	0.26	1.66
Our (w/o SCT)	0.57	0.15	0.62	1.09	0.81	0.35	0.22	**2.26**	0.65	0.19	0.16	0.81	1.13	0.64	0.57	1.85
DeepSeek-V3.2	1.39	1.05	1.65	1.72	0.98	0.56	0.17	2.58	1.10	0.65	0.57	1.56	1.41	0.91	0.83	2.06
ChatGPT-5.2	3.44	2.57	2.46	2.93	2.58	2.11	2.20	3.04	1.99	1.41	1.12	2.31	6.23	4.92	4.82	4.96

Table 3: Task 2: Sentence-to-Dialect IPA Generation. Full denotes the constrained edit distance per character on the entire test set; Train and Test denote the constrained edit distances per character on the subsets of characters seen and unseen in the training set, respectively; Pred. indicates the proportion of characters that are out of vocabulary for the target dialect and thus require pseudo labels generated by applying the sound change rules from Task 1. All data are the results of a single run.

Specifically, for Task 1, we specify the values of each field in the dictionary in the prompt; for Task 2, we provide the phonology of the target dialect, including all possible initials, finals, and tones. The detailed format and examples can be found in Appendix D.

6.3 Evaluation Metrics

6.3.1 Metrics for Task 1: Conditional Sound Change Rule Induction

For Task 1, we evaluate the generated sound change rules from both execution correctness and rule complexity perspectives. We report two metrics: Accuracy and Score.

Accuracy Accuracy is defined as the proportion of test instances for which the predicted outcome is correct. The prediction is considered correct if the generated function's output matches the gold dialect pronunciation or is included in the returned candidate list.

Score The same as the reward function used in the GRPO stage (see Section 5.3).

Because sound change rules cover varying numbers of characters (from less than 10 to hundreds), we report two versions of the above metrics: unweighted (UA/US) metrics and weighted (WA/WS) metrics by the number of test instances covered by each rule. The unweighted metrics reflect the model's general rule induction capability by averaging rule-level results equally, while the weighted metrics reflect the practical application quality weighted by the number of test instances covered by each rule. Meanwhile, Score incorporates rule conciseness penalties over Accuracy to evaluate whether the generated rules are overfitted.

6.3.2 Metrics for Task 2: Dialectal IPA Pronunciation Prediction

For Task 2, due to the existence of legitimate pronunciation variants for certain characters, we adopt a modified Levenshtein edit distance as the metric, which is defined as the **minimum number of insertion, deletion, and substitution operations** required to transform the model's output string into **one of the candidate pronunciations**. The detailed implementation algorithm is provided in Appendix B.3.

For characters not present in the dialect lexicon, we use the sound change rules generated in Task 1 (consistently using the rules generated by our full method to ensure fairness) to generate their pronunciations from Middle Chinese phonological representations, treating the results as pseudo labels.

During the distance calculation, we can also obtain the alignment between the predicted data and the gold pronunciations. After obtaining the alignment, we categorize the characters in the sentences into three groups: those appearing in the training set, those appearing in the test set, and those not present in the dialect lexicon. We then compute the edit distances for each group separately to analyze the model's generalization ability. To better capture phonetic naturalness and perceptual similarity, we additionally compute Panphon feature edit distance (Mortensen et al., 2016) to replace the standard Levenshtein operations, detailed in Appendix E.

6.4 Experimental Results

6.4.1 Task 1

Table 2 presents the main results for Task 1. Our full method achieves superior Score metrics compared to DeepSeek-V3.2 and ChatGPT-5.2 on most dialects, while its Accuracy is second only to ChatGPT-5.2, indicating strong rule generation capabilities.

Notably, ChatGPT-5.2 tends to generate more complex rules and incorporates judgments for irregular characters, effectively "overfitting" to the data, which results in high Accuracy but lower Score. A more detailed analysis is provided in Appendix E.3.1.

Removing any component from our full method, whether GRPO reinforcement learning, single-character tokenization strategy, or joint training, leads to a decline in overall model performance, indicating that each component positively contributes to enhancing the model's capabilities.

We also conducted significance tests to further support the reliability of the experimental results, detailed in Appendix E.2.

6.4.2 Task 2

The main results for Task 2 are presented in Table 3. For most of the dialects, our approach significantly outperforms all baseline models.

If Middle Chinese phonological information is not incorporated during training (Our (w/o MC)), the model's performance degrades, indicating that including Middle Chinese phonological information helps enhance the model's generalization ability, especially for handling unseen characters in the test set. The single-character tokenization strategy also positively impacts model performance.

Also, we observe that except for the Qwen3-8B base model, which performs poorly, other models exhibit relatively high edit distances in the out-of-vocabulary sets, indicating that there remains a gap between the pseudo labels generated from the sound change rules and the actual pronunciations, reflecting the limitations of the models.

It is worth noting that although ChatGPT-5.2 performed well in Task 1, its performance in Task 2 is suboptimal, with edit distances greater than 1 for most dialects except for relatively resource-rich Guangzhou dialect (Cantonese) and Harbin dialect, which is relatively close to Standard Chinese. This further confirms the inadequacies of existing models in understanding and generating low-resource dialects. A more detailed analysis is provided in Appendix E.3.2.

7 Conclusion

This paper explores the capability of LLMs to summarize historical sound change patterns and leverages ancestor language information to enhance their performance on Chinese dialects.

We propose a two-stage training framework that combines supervised fine-tuning and reinforcement learning, enabling the model to explicitly generate sound change rules while implicitly predicting dialectal pronunciations based on these rules.

Empirical results on a diverse set of dialects demonstrate consistent improvements over strong baselines, highlighting the value of ancestor language information in dialect modeling.

Future work can extend from character-level transcription to word-level dialect translation and transcription, as well as integrate the work into multimodal speech models to convert IPA-based representations into human-perceivable audio. Moreover, designing reinforcement learning rewards based on linguistic plausibility is a promising direction.

We hope this research encourages further integration of historical linguistic knowledge with LLMs to support dialect understanding and preservation.

Limitations

We assume dialectal pronunciations derive from Middle Chinese via regular sound changes. However, this strict tree model overlooks archaic strata, loanwords, and mutual influence between geographically proximate dialects.

Some dialects exhibit extensive literary and colloquial reading differences (i.e., the same Chinese character has different pronunciations for spoken and written vocabulary). We did not specifically distinguish these, treating them as random divergences.

Some dialects exhibit tone sandhi phenomena in sentences (i.e., when two characters combine into a word, the tones may change). We did not specifically address this phenomenon in our IPA inference.

Acknowledgments

We would like to thank the anonymous reviewers and the meta-reviewer for their constructive feedback and helpful suggestions. This work was supported by the National Natural Science Foundation of China (Grant No. 61671064). We also thank all the contributors to MCPDict for their efforts in building and maintaining this valuable resource for Chinese historical phonology and dialectology.

References

Thiago Costa Chacon and Johann-Mattis List. 2016. Improved computational models of sound change shed light on the history of the tukanoan languages. *Journal of Language Relationship*, 13(3-4):177–204.

Kalvin Chang, Chenxuan Cui, Youngmin Kim, and David R. Mortensen. 2022. WikiHan: A new comparative dataset for Chinese languages. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3563–3569, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Kalvin Chang, Nathaniel Robinson, Anna Cai, Ting Chen, Annie Zhang, and David Mortensen. 2023. Automating sound change prediction for phylogenetic inference: A tukanoan case study. In *Proceedings of the 4th Workshop on Computational Approaches to Historical Language Change*, pages 129–142, Singapore. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Antonios Dimakis, John Pavlopoulos, and Antonios Anastasopoulos. 2025. Dialect normalization using large language models and morphological rules. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 23696–23714, Vienna, Austria. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Zhige Huang, Haoan Jin, Mengyue Wu, and Kenny Q. Zhu. 2024. Automatic reconstruction of Ancient Chinese pronunciations. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5689–5698, Miami, Florida, USA. Association for Computational Linguistics.

Jiyue Jiang, Pengan Chen, Liheng Chen, Sheng Wang, Qinghang Bao, Lingpeng Kong, Yu Li, and Chuan Wu. 2025. How well do LLMs handle Cantonese? benchmarking Cantonese capabilities of large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4464–4505, Albuquerque, New Mexico. Association for Computational Linguistics.

Young Min Kim, Kalvin Chang, Chenxuan Cui, and David R. Mortensen. 2023. Transformed protoform reconstruction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 24–38, Toronto, Canada. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Jinggui Liang, Dung Vo, Yap Hong Xian, Hai Leong Chieu, Kian Ming A. Chai, Jing Jiang, and Lizi Liao. 2025. Colloquial singaporean English style transfer with fine-grained explainable control. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26962–26983, Vienna, Austria. Association for Computational Linguistics.

Fangru Lin, Shaoguang Mao, Emanuele La Malfa, Valentin Hofmann, Adrian de Wynter, Xun Wang, Si-Qing Chen, Michael J. Wooldridge, Janet B. Pierrehumbert, and Furu Wei. 2025. Assessing dialect fairness and robustness of large language models in reasoning tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6317–6342, Vienna, Austria. Association for Computational Linguistics.

Yanchen Liu, William Held, and Diyi Yang. 2023. DADA: Dialect adaptation via dynamic aggregation of linguistic rules. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13776–13793, Singapore. Association for Computational Linguistics.

Jiaming Luo. 2021. Automatic methods for sound change discovery.

Yun Mai. 2009. *Introduction to Chinese Phonology*. Jiangsu Education Press. In Chinese.

Carlo Meloni, Shauli Ravfogel, and Yoav Goldberg. 2021. Ab antiquo: Neural proto-language reconstruction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4460–4473, Online. Association for Computational Linguistics.

David R. Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. PanPhon: A resource for mapping IPA segments to articulatory feature vectors. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3475–3484, Osaka, Japan. The COLING 2016 Organizing Committee.

Atharva Naik, Darsh Agrawal, Hong Sng, Clayton Marr, Kexun Zhang, Nathaniel Romney Robinson, Kalvin Chang, Rebecca Byrnes, Aravind Mysore, Carolyn Rose, and David R. Mortensen. 2025. Programming by example meets historical linguistics: A large language model based approach to sound law induction. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29628–29647, Vienna, Austria. Association for Computational Linguistics.

OpenAI. 2025. Chatgpt-5.2. `https://chat.openai.com/`. Accessed: December 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Patrick Sims-Williams. 2018. Mechanising historical phonology. *Transactions of the Philological Society*, 116(3):555–573.

Li Wang. 1985. *The History of Chinese Phonology*. China Social Sciences Press. In Chinese.

Zhenghui Xiong, Zhenxing Zhang, Xing Huang, Bu Dao, and Jiayan Zou. 2012. *Language Atlas of China: 2nd Edition. Volume on Chinese Dialects*. Commercial Press. In Chinese.

Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, Yuanjun Lv, Yongqi Wang, Dake Guo, He Wang, Linhan Ma, Pei Zhang, Xinyu Zhang, Hongkun Hao, Zishan Guo, and 19 others. 2025. Qwen3-omni technical report. *Preprint*, arXiv:2509.17765.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Siheng Zhang, Xingjun Tan, Yanqiang Lei, Xianxiang Wang, Zhizhong Zhang, and Yuan Xie. 2023. Cvte-poly: A new benchmark for chinese polyphone disambiguation. In *Proc. Interspeech 2023*, pages 5526–5530.

A Chinese Syllables and the Representation of Middle Chinese

In Chinese phonology, both Middle Chinese and modern Chinese dialects can be abstractly analyzed as syllables consisting of three basic components: an **initial**, a **final**, and a **tone**. Figure 5 shows the initials, finals, and tones of the character “文字” (meaning “script”) in Middle Chinese and modern dialects, represented in different ways.

For modern dialects, the International Phonetic Alphabet (IPA) is used to transcribe the syllables.

For Middle Chinese, due to the existence of multiple reconstructed pronunciations with varying IPA details, we do not use IPA to represent Middle Chinese syllables. Instead, we use the following methods to represent Middle Chinese syllables:

Phonological Status A set of discrete categorical features:

- **Initial** (声母/聲母): consonant at the onset of the syllable.
- **Roundedness** (开合/開合): whether the final involves lip rounding, can be empty in some cases.
- **Division** (等/等): one of the four traditional categories defined by vowel quality and structure.
- **Chongniu** (重纽/重紐, lit. “repeated button”): subdivision of certain third-division finals, can be empty in some cases.
- **Rhyme** (韵/韻): broad class of finals that can rhyme.
- **Tone** (调/調): tonal category.

It can be loosely assumed that the combination of **roundedness**, **division**, **chongniu**, and **rhyme** jointly determines the final of a character in Middle Chinese.

For each of the initials, rhymes, and tones, a conventional representative character is used to denote each category; for example, the initial “幫” represents the initial of the character “幫”, the rhyme “

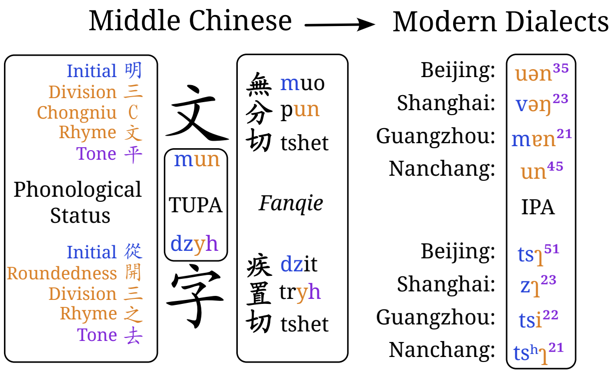

Figure 5: Chinese syllables of the character "文字" (meaning "script") in Middle Chinese and modern dialects, showing initials, finals, and tones with different representations.

東" represents the rhyme of the character "東", and the tone "平" represents the tone of the character "平" in Middle Chinese.

In addition to the above attributes, the following auxiliary features are also commonly used:

- **Voicing** (清浊/清濁): manner of articulation of the initial.
- **Initial group** (组/組): grouping of initials by place of articulation.
- **Rhyme group** (摄/攝): higher-level grouping of rhymes by phonological proximity.

These auxiliary features are not strictly required to identify Middle Chinese syllables, but they often influence sound change patterns from Middle Chinese to modern dialects. For additional details and examples of each feature, see standard references in Chinese historical phonology.

We therefore provide them as additional conditioning information to the model.

Tshet-uinh Phonetic Alphabet (切韵拼音/切韻拼音, TUPA) One of the default transcription methods used in MCPDict, a Latin transcription scheme for Middle Chinese phonology that does not depend on specific reconstructions.

***Fanqie* (反切/反切)** A traditional method of indicating pronunciation commonly used in ancient Chinese dictionaries, which uses two characters with known pronunciations to represent the pronunciation of the target character. The first character indicates the initial, while the second character indicates the combination of the final and tone. For example, the fanqie for the character "文" is "無分切", indicating that in Middle Chinese, its initial is the same as that of "無", and its final and tone are the same as those of "分".

More details can be found in works on Chinese historical phonology such as Wang (1985) and Mai (2009).

B Experiment Details

B.1 Training Configuration

All experiments are conducted on a single NVIDIA A800 GPU with 80GB of memory.

All hyperparameters not explicitly specified below use default values of the Hugging Face ecosystem.

The SFT training configuration is as follows:

```
per_device_train_batch_size = 8
gradient_accumulation_steps = 1
num_train_epochs = 5
learning_rate = 2e-4
BF16 = True
```

The hyperparameters used during the GRPO training stage are listed below:

```
learning_rate = 2e-5
max_grad_norm = 0.05
num_train_epochs = 1
save_steps = 100
per_device_train_batch_size = 8
gradient_accumulation_steps = 1
max_completion_length = 2500
num_generations = 8
max_prompt_length = 2500
BF16 = True
```

B.2 Inference Configuration

For local models, we use vLLM(Kwon et al., 2023) for inference acceleration, with the following parameters:

```
temperature=0.6
top_p=0.95
max_tokens=5000
top_k=20
```

For online models accessed via API, we set `max_completion_tokens=10000` and enable reasoning, while keeping other parameters at their default values for inference.

B.3 Details for Modified Levenshtein edit distance

Let the predicted IPA sequence be $s = [s_1, s_2, \ldots, s_n]$, and the gold reference be a sequence of candidate lists $l = [L_1, L_2, \ldots, L_m]$, where each L_j is a set of valid string candidates.

The goal is to transform s into a form that matches one of the candidates in l through insertion, deletion, and substitution operations, while minimizing the total cost.

The operations and their associated costs are defined as follows:

- **Insertion**: inserting a string t into s, with cost equal to the length of t.
- **Substitution**: replacing s_i with a string t, with cost equal to the edit distance between s_i and t.
- **Deletion**: deleting a string s_i, with cost equal to the length of s_i.

In task 2 of this paper, s denotes the IPA sequence generated by the model, and l denotes the gold sequence of candidate IPA lists. Note that the alignment output was solely for the purpose of computing and minimizing the Levenshtein distance on different data distribution subsets (train, test, and out-of-vocabulary) as described in Section 6.3.2.

Algorithm 1 Constrained Edit Distance

Input: Sequence of strings $S = [s_1, s_2, \dots, s_n]$, Sequence of candidate lists $\mathcal{L} = [L_1, L_2, \dots, L_m]$
Output: Minimum edit cost $D_{n,m}$ and alignment array A

$n \leftarrow \text{length}(S)$
$m \leftarrow \text{length}(\mathcal{L})$
Initialize Cost Functions:
$C_{\text{ins}}[j] \leftarrow \min\{|t| : t \in L_j\}$ for $j \in \{1 \dots m\}$
Function $CostSub(i, j)$: **return** $\min\{\text{EditDist}(s_i, t) : t \in L_j\}$
Function $CostDel(i)$: **return** $|s_i|$
Initialize DP Tables:
Let D be a matrix of size $(n+1) \times (m+1)$ initialized to ∞
Let P be a matrix of size $(n+1) \times (m+1)$ for backtracking
$D[0][0] \leftarrow 0$
▷ Base Case: Insertions only
for $j \leftarrow 1$ to m **do**
    $D[0][j] \leftarrow D[0][j-1] + C_{\text{ins}}[j]$
    $P[0][j] \leftarrow (\text{“ins”}, 0, j-1)$
end for
▷ Base Case: Deletions only
for $i \leftarrow 1$ to n **do**
    $D[i][0] \leftarrow D[i-1][0] + CostDel(i)$
    $P[i][0] \leftarrow (\text{“del”}, i-1, 0)$
end for
Fill DP Table:
for $i \leftarrow 1$ to n **do**
    for $j \leftarrow 1$ to m **do**
        $v_{\text{sub}} \leftarrow D[i-1][j-1] + CostSub(i, j)$
        $v_{\text{ins}} \leftarrow D[i][j-1] + C_{\text{ins}}[j]$
        $v_{\text{del}} \leftarrow D[i-1][j] + CostDel(i)$
        $D[i][j] \leftarrow \min(v_{\text{sub}}, v_{\text{ins}}, v_{\text{del}})$
        if $D[i][j] = v_{\text{sub}}$ **then**
            $P[i][j] \leftarrow (\text{“sub”}, i-1, j-1)$
        else if $D[i][j] = v_{\text{ins}}$ **then**
            $P[i][j] \leftarrow (\text{“ins”}, i, j-1)$
        else
            $P[i][j] \leftarrow (\text{“del”}, i-1, j)$
        end if
    end for
end for
Backtracking:
Initialize alignment array A of size n with -1
$i \leftarrow n, \quad j \leftarrow m$
while $i > 0$ **or** $j > 0$ **do**
    $(op, p_i, p_j) \leftarrow P[i][j]$
    if $op =$ “sub” **then**
        $A[i] \leftarrow j$ ▷ Align s_i to L_j
        $i \leftarrow p_i, \quad j \leftarrow p_j$
    else if $op =$ “del” **then**
        $A[i] \leftarrow -1$ ▷ s_i is deleted
        $i \leftarrow p_i, \quad j \leftarrow p_j$
    else ▷ $op =$ “ins”
        $i \leftarrow p_i, \quad j \leftarrow p_j$
    end if
end while
return $D[n][m], A$

C Examples of Structured Base Data

In this section, we provide detailed examples of the three categories of Structured Base Data discussed in Section 4.2. These structured data serve as intermediate representations before being transformed into supervised fine-tuning (SFT) prompts via predefined templates.

C.1 Category 1: Sound Change Rule Exemplars

Rule exemplars are manually constructed by the authors using data from MCPDict to provide the model with the necessary format for deriving and generating executable Python rule functions (Task 1).

Example of a sound change rule exemplar (translated and simplified for illustration)

```
{
  "dialect": "Chengdu",
  "rule": "def initial_d(ch):  if ch[\"tone
\"] == \"level\":    return \"t \"  return
\"t\"\n",
  "examples": [
    {
      "case": "if the tone is level, the
initial consonant is aspirated",
      "target": "t ",
      "chars": [
        ["臺", "deoj", "t ai21"],
        ["停", "deng", "t in21"]
      ]
    },
    {
      "case": "otherwise, the initial
consonant is unaspirated",
      "target": "t",
      "chars": [
        ["豆", "douh", "təu213"],
        ["電", "denh", "tian213"]
      ]
    }
  ],
  "irregular_examples": [
    ["膻", "danq", "san213"]
  ]
}
```

C.2 Category 2: Character-level Sound Change Data

This category allows the model to master dialectal phonology and historical evolutionary patterns from Middle Chinese (MC) to modern dialects. It contains two subcategories: MC pronunciation data and sound change pair data. All MC character pronunciations participate in training, so the model can learn as many MC phonetic structures as possible. Conversely, the sound change pairs are split into 95% training and 5% test sets to evaluate the model's derivation capabilities on unseen data.

Example of Middle Chinese pronunciation data

```
{
  "char": "好",
  "pronunciations": [
    {"tupa": "hawq", "fanqie": "呼皓切",
"status": "曉一開豪上"},
    {"tupa": "hawh", "fanqie": "呼到切",
"status": "曉一開豪去"}
  ]
}
```

Example of a sound change pair

```
{
  "char": "好",
  "dialect": "Standard Chinese",
  "pairs": [
    ["hawh", ["xau51"]],
    ["hawq", ["xau214"]]
  ]
}
```

C.3 Category 3: Sentence-level Pronunciation Data

This data category enables the model to master the reasoning format of deriving contextual pronunciations combined with MC information. We ensure that all constituent characters in the training sentences belong to the training subset of the character-level sound change pairs.

Example of Middle Chinese pronunciation data

```
{
  "dialect": "Guangzhou",
  "chars": ["管", "道", "由", ... ],
  "tupas": ["kwanq", "dawq", "ju", ... ],
  "ipas": ["kun35", "tou22", "jɐu21", ... ]
}
```

D Prompt Template

In this section, we present the prompt templates used for training and inference. The templates include both the original Chinese version and its English translation to facilitate readers' understanding of the structure and content of the prompts. All prompts are in Traditional Chinese, which is the primary form used in MCPDict as well as commonly used in Chinese historical phonology research.

D.1 Query for Single Character Pronunciation

D.1.1 Query for Sound Change from Middle Chinese to Modern Dialects

The prompt templates with example and translation are shown below.

Original Chinese Template

User:
分析漢字“{char}”的中古漢語音韻地位，以JSON格式給出“{char}”的音韻信息以及{dialect name}讀音。
Assistant Reasoning:
Case 1: single Middle Chinese pronunciation, single modern dialect pronunciation
“{char}”在廣韻中的反切是“{fanqie}”，音韻地位是“{status}”，切韻拼音是“{tupa}”，因此{dialect name}讀音是“{dialect

pronunciation}”。
Case 2: single Middle Chinese pronunciation, multiple modern dialect pronunciations
“{char}”在廣韻中的反切是“{fanqie}”，音韻地位是“{status}”，切韻拼音是“{tupa}”，但該讀音在{dialect name}中發生了分化，可能的讀音有“{dialect pronunciations}”。
Case 3: multiple Middle Chinese pronunciations
“{char}”在廣韻中有{number of the pronunciations}個讀音，分別是:
for i from 1 to number of the pronunciations:
{i}. 反切是“{fanqie i}”，音韻地位是“{status i}”，切韻拼音是“{tupa i}”，
Case 3.1: no corresponding pronunciation in the modern dialect
但在{dialect name}中沒有對應的讀音。
Case 3.2: one corresponding pronunciation in the modern dialect
因此{dialect name}讀音是“{dialect pronunciation}”。
Case 3.3: multiple corresponding pronunciations in the modern dialect
但該讀音在{dialect name}中發生了分化，可能的讀音有“{dialect pronunciations}”。
接下來，我會根據上述分析，以JSON格式給出“{char}”的音韻信息以及{dialect name}讀音。
Assistant:
{JSON format Middle Chinese information and dialect pronunciations}

English Translation of the Template

User:
Analyze the Middle Chinese phonetic status of the character “{char}” and provide the phonetic information of “{char}” and its pronunciation in {dialect name} in JSON format.
Assistant Reasoning:
Case 1: single Middle Chinese pronunciation, single modern dialect pronunciation
The fanqie of “{char}” in Guangyun is “{fanqie}”, the phonetic status is “{status}”, and the Tshet-uinh Phonetic Alphabet is “{tupa}”. Therefore, the pronunciation in {dialect name} is “{dialect pronunciation}”.
Case 2: single Middle Chinese pronunciation, multiple modern dialect pronunciations
The fanqie of “{char}” in Guangyun is “{fanqie}”, the phonetic status is “{status}”, and the Tshet-uinh Phonetic Alphabet is “{tupa}”. However, this pronunciation has diverged in {dialect name}, and the possible pronunciations are “{dialect pronunciations}”.
Case 3: multiple Middle Chinese pronunciations
The character “{char}” has {number of the pronunciations} pronunciations in Guangyun, which are as follows:
for i from 1 to number of the pronunciations:
{i}. The fanqie is “{fanqie i}”, the phonetic status is “{status i}”, and the Tshet-uinh Phonetic Alphabet is “{tupa i}”.
Case 3.1: no corresponding pronunciation in the modern dialect
However, there is no corresponding pronunciation in {dialect name}.
Case 3.2: one corresponding pronunciation in the modern dialect
Therefore, the pronunciation in {dialect name} is “{dialect pronunciation}”.
Case 3.3: multiple corresponding pronunciations in the modern dialect
However, this pronunciation has diverged in {dialect name}, and the possible pronunciations are “{dialect pronunciations}”.
Next, based on the above analysis, I will provide the phonetic information of “{char}” and its pronunciation in {dialect name} in JSON format.
Assistant: {JSON format Middle Chinese information and dialect pronunciations}

Example

User:
分析漢字“更”的中古漢語音韻地位，以JSON格式給出“更”的音韻信息以及普通話讀音。
Assistant Reasoning:
“更”在廣韻中有2個讀音，分別是:
1. 反切是“古行切”，音韻地位是“見開二庚平”，切韻拼音是“kaeng”，因此普通話讀音是“kəŋ^{55}”。
2. 反切是“古孟切”，音韻地位是“見開二庚去”，切韻拼音是“kaengh”，因此普通話讀音是“kəŋ^{51}”。
接下來，我會根據上述分析，以JSON格式給出“更”的音韻信息以及普通話讀音。
Assistant:
```
{"char": "更", "pronunciations": [{"tupa": "kaeng", "status": "見 開 二 庚 平", "dialect_pronunciations": ["kəŋ55"]}, {"tupa": "kaengh", "status": "見開二庚去", "dialect_pronunciations": ["kəŋ51"]}]
```

English Translation of the Example

User:
Analyze the Middle Chinese phonetic status of the character “更” and provide the phonetic information of “更” and its pronunciation in Standard Chinese in JSON format.
Assistant Reasoning:
The character “更” has 2 pronunciations in Guangyun, which are as follows:
1. The fanqie is “古行切”, the phonetic status is “見開二庚平”, and the Tshet-uinh Phonetic Alphabet is “kaeng”. Therefore, the pronunciation in Standard Chinese is “kəŋ^{55}”.
2. The fanqie is “古孟切”, the phonetic status is “見開二庚去”, and the Tshet-uinh Phonetic Alphabet is “kaengh”. Therefore, the pronunciation in Standard Chinese is “kəŋ^{51}”.
Next, based on the above analysis, I will provide the phonetic information of “更” and its pronunciation in Standard Chinese in JSON format.
Assistant:
```
{"char": "更", "pronunciations": [{"tupa": "kaeng", "status": "見 開 二 庚 平", "dialect_pronunciations": ["kəŋ55"]}, {"tupa": "kaengh", "status": "見開二庚去", "dialect_pronunciations": ["kəŋ51"]}]
```

D.1.2 Query for Middle Chinese Pronunciation

The prompt templates with example and translation are shown below.

Original Chinese Template

User:
以JSON格式給出"{char}"的所有反切、音韻地位和切韻拼音。
Assistant:
{JSON format Middle Chinese information}

English Translation of the Template

User:
Provide all the fanqie, phonological status and Tshet-uinh Phonetic Alphabet information of "{char}" in JSON format.
Assistant: {JSON format Middle Chinese information}

Example

User:
以JSON格式給出"更"的所有反切、音韻地位和切韻拼音。
Assistant:
{"char": "更", "pronunciations": [{ "fanqie": "古 行 切", "status": "見 開 二 庚 平","tupa": "kaeng"}, { "fanqie": "古孟切", "status": "見開二庚去","tupa": "kaengh"}]

English Translation of the Example

User:
Provide all the fanqie, phonological status and Tshet-uinh Phonetic Alphabet information of "更" in JSON format.
Assistant:
{"char": "更", "pronunciations": [{ "fanqie": "古 行 切", "status": "見 開 二 庚 平","tupa": "kaeng"}, { "fanqie": "古孟切", "status": "見開二庚去","tupa": "kaengh"}]

D.1.3 Query for Modern Dialect Pronunciation

This template is only used for the ablation experiment without Middle Chinese information. The prompt templates with example and translation are shown below.

Original Chinese Template

User:
以JSON格式給出"{char}"的所有{dialect name}讀音。
Assistant:
{JSON format Modern Dialect information}

English Translation of the Template

User:
Provide all the {dialect name} pronunciations of "{char}" in JSON format.
Assistant: {JSON format Modern Dialect information}

Example

User:
以JSON格式給出"更"的所有普通話讀音。
Assistant:
{"char": "更", "pronunciations": ["kəŋ^{55}", "kəŋ^{51}"]}

English Translation of the Example

User:
Provide all the Standard Chinese pronunciations of "更" in JSON format.
Assistant:
{"char": "更", "pronunciations": ["kəŋ^{55}", "kəŋ^{51}"]}

D.2 SFT Data For Reverse Lookup

The prompt templates with example and translation are shown below.

Original Chinese Template

User:
已知一個漢字的{dialect name}讀音爲{dialect pronunciation}，其切韻拼音可能是：{list of candidates}中的哪一個?
Assistant:
{dialect name}讀音爲{dialect pronunciation}的漢字，其可能的切韻拼音是{correct choice}。

English Translation of the Template

User:
Given that the pronunciation of a Chinese character in {dialect name} is {dialect pronunciation}, which one of the following Tshet-uinh Phonetic Alphabet candidates is it:
{list of candidates}
Assistant:
The possible Tshet-uinh Phonetic Alphabet for the character with {dialect name} pronunciation {dialect pronunciation} is {correct choice}.

Example

已知一個漢字的普通話讀音爲tɕʰiaŋ^{55}，其切韻拼音可能是:
A. khyang
B. gyang
C. dzyang
D. kyang
中的哪一個?
Assistant:
普通話讀音爲tɕʰiaŋ^{55}的漢字，其可能的切韻拼音是A. khyang。

English Translation of the Example

User:
Given that the pronunciation of a Chinese character in Standard Chinese is tɕʰiaŋ^{55}, which one of the following Tshet-uinh Phonetic Alphabet candidates is it:

A. khyang
B. gyang
C. dzyang
D. kyang
Assistant:
The possible Tshet-uinh Phonetic Alphabet for the character with Standard Chinese pronunciation $t\text{ɕ}^h\text{iaŋ}^{55}$ is A. khyang.
Note: The pronunciations of the four options are $t\text{ɕ}^h\text{iaŋ}^{55}$, $t\text{ɕ}^h\text{iaŋ}^{35}$, $t\text{ɕ}^h\text{iaŋ}^{35}$, and $t\text{ɕiaŋ}^{55}$ respectively.

D.3 Sound Change Rule Generation

D.3.1 SFT Prompt

The prompt templates with example and translation are shown below.

Original Chinese Template

User:
下面是一些中古漢語中`{phonetic part}`為“`{part name}`”的字在`{dialect name}`中的讀音，按`{phonetic part}`分類:
`{list of all possible pronunciations in the dialect with percentages}`
根據上述示例，生成一個規則函數，用於判斷在中古漢語中`{phonetic part}`為“`{part name}`”的字在`{dialect name}`中的`{phonetic part}`。要點如下:
1. 只返回Python函數代碼，函數名稱為“`{phonetic part}_{part name}`”，函數接受一個字典參數ch，包含漢字的以下音韻信息字段：'聲母'、'開合'、'等'、'重紐'、'韻'、'調'、'清濁'、'組'、'攝'，返回該字在`{dialect name}`中的`{phonetic part}`。
2. 如果需要根據{dialect name}的聲韻調進行額外的判斷，可以再返回一個名稱為“`{phonetic part}_{part name}_現`”的函數，接受上述參數ch外，還接受第二個參數syl，包含漢字在普通話中的'聲母'、'韻母'、'聲調'三個字段，其中'聲母'來自函數“`{phonetic part}_{part name}`”的輸出，另外兩項爲實際發音，你可以結合ch和syl的信息對結果進行二次修改，以提升正確率。
3. 最終評分存在長度懲罰，請保持輸出精簡，對於明顯不符合系統演變規則的個例，視爲不規則讀音，應該直接忽略，不在函數中體現。
4. 如果存在無法細分的隨機演變，你可以返回一個包含多個可能讀音的列表，但所有該行返回的測試用例計算正確率會有0.8的懲罰系數，並且在最終得分扣除真實分布與均勻分布的KL散度與在該行返回的正確測試用例占比的乘積。
Assistant Reasoning:
嗯，我現在需要生成中古漢語`{phonetic part}`為“`{part name}`”的漢字演變爲`{dialect name}{phonetic part}`的規則函數。根據上述示例，我可以總結出`{phonetic part}`“`{part name}`”的以下規則:
`{list of rules}`
此外，下面幾個示例似乎不規則，我應該忽略它們:
`{list of irregular characters}`
接下來，我將根據這些規則編寫Python函數來實現對應的判斷邏輯。
Assistant:
`{Python code for the sound change rule}`

English Translation of the Template

User:
Here are some characters with `{phonetic part}` as “`{part name}`” in Middle Chinese and their pronunciations in `{dialect name}`, categorized by `{phonetic part}`:
`{list of all possible pronunciations in the dialect with percentages}`
Based on the above examples, generate a rule function to determine the `{phonetic part}` of characters with `{phonetic part}` as “`{part name}`” in Middle Chinese in `{dialect name}`. Key points are as follows:
1. Only return the Python function code, with the function name as “`{phonetic part}_{part name}`”, which takes a dictionary parameter ch containing the following phonetic information fields of the character: '聲母'(initial), '開合'(roundedness), '等'(division), '重紐'(chongniu), '韻'(rhyme), '調'(tone), '清濁'(voicing), '組'(initial group), '攝'(rhyme group), and returns the `{phonetic part}` of the character in `{dialect name}`.
2. If additional judgments are needed based on the {dialect name} initials, finals, and tones, you can also return a function named “`{phonetic part}_{part name}_現`”, which takes the above parameter ch and a second parameter syl, containing the following three fields of the character in Standard Chinese: '聲母'(initial), '韻母'(final), '聲調'(tone), where '`{phonetic part}`'(initial) comes from the output of the function “`{phonetic part}_{part name}`”, and the other two are the actual pronunciations. You can combine the information from ch and syl to make secondary modifications to the results to improve accuracy.
3. The final score has a length penalty, so keep the output concise. For obvious exceptions that do not conform to systematic sound change rules, treat them as irregular pronunciations and ignore them, without reflecting them in the function.
4. If there are random changes that cannot be further divided, you can return a list of multiple possible pronunciations. However, all test cases returned in that line will have a penalty factor of 0.8 for accuracy calculation, and the final score will deduct the product of the KL divergence between the true distribution and the uniform distribution and the proportion of correct test cases returned in that line.
Assistant Reasoning:
Okay, now I need to generate a rule function for the sound change of characters with `{phonetic part}` as “`{part name}`” in Middle Chinese to `{dialect name}{phonetic part}`. Based on the above examples, I can summarize the following rules for `{phonetic part}` “`{part name}`”:
`{list of rules}`
Besides, the following examples seem irregular, and I should ignore them:
`{list of irregular characters}`
Next, I will write a Python function based on these rules to implement the corresponding judgment logic.
Assistant:
`{Python code for the sound change rule}`

Example

User:
下面是一些中古漢語中聲母為“心”的字在普通話中的讀音，按聲母分類:

ts^h（佔比約1%）：
- 賜：sieh(心開三支去) -> ts^{h}ɿ^{51}
s（佔比約50%）：
- 斯：sie(心開三支平) -> sɿ^{55}
- 薩：sat(心開一寒入) -> sa^{51}
- 孫：son(心合一魂平) -> suən^{55}
tɕ（佔比約2%）：
- 峻：swinh(心合三真去) -> tɕyn^{51}
ɕ（佔比約47%）：
- 性：siaengh(心開三清去) -> ɕiŋ^{51}
- 膝：sit(心開三真入) -> ɕi^{55}
- 選：swienq(心合三仙上) -> ɕyan^{214}
ts（佔比約1%）：
- 噪：sawh(心開一豪去) -> tsau^{51}

根據上述示例，生成一個規則函數，用於判斷在中古漢語中聲母為“心”的字在普通話中的聲母。要點如下:

1. 只返回Python函數代碼，函數名稱為“聲母_心”，函數接受一個字典參數ch，包含漢字的以下音韻信息字段：'聲母'、'開合'、'等'、'重紐'、'韻'、'調'、'清濁'、'組'、'攝'，返回該字在普通話中的聲母。
2. 如果需要根據普通話的聲韻調進行額外的判斷，可以再返回一個名稱為“聲母_心_現”的函數，接受上述參數ch外，還接受第二個參數syl，包含漢字在普通話中的'聲母'、'韻母'、'聲調'三個字段，其中'聲母'來自函數“聲母_心”的輸出，另外 兩項爲實際發音，你可以結合ch和syl的信息對結果進行二次修改，以提升正確率。
3. 最終評分存在長度懲罰，請保持輸出精簡，對於明顯不符合系統演變規則的個例，視爲不規則讀音，應該直接忽略，不在函數中體現。
4. 如果存在無法細分的隨機演變，你可以返回一個包含多個可能讀音的列表，但所有該行返回的測試用例計算正確率會有0.8的懲罰系數，並且在最終得分扣除真實分布與均勻分布的KL散度與在該行返回的正確測試用例占比的乘積。

Assistant Reasoning:
嗯，我現在需要生成中古漢語聲母爲“心”的漢字演變爲普通話聲母的規則函數。根據上述示例，我可以總結出聲母“心”的以下規則:
- 通常情況，變爲s（例如，斯(sie -> sɿ^{55})、孫(son -> suən^{55})、薩(sat -> sa^{51})）。
- 若普通話韻母以i或y開頭，變爲ɕ（例如，選(swienq -> ɕyan^{214})、膝(sit -> ɕi^{55})、性(siaengh -> ɕiŋ^{51})）。

此外，下面幾個示例似乎不規則，我應該忽略它們:
- 賜(sieh -> ts^{h}ɿ^{51})
- 峻(swinh -> tɕyn^{51})
- 噪(sawh -> tsau^{51})

接下來，我將根據這些規則編寫Python函數來實現對應的判斷邏輯。

Assistant:
```
def 聲母_心(ch):
    return "s"
def 聲母_心_現(ch: CharInfo, syl):
    if syl["韻母"].startswith(("i", "y")):
    # if the final starts with i or y
        return "ɕ"
    # otherwise keep the original prediction
    return syl["聲母"]
```

English Translation of the Example

User:
Here are some characters with initial as “心” in Middle Chinese and their pronunciations in Standard Chinese, categorized by initial:
ts^h (Approximately 1%):
- 賜: sieh(心開三支去) -> ts^{h}ɿ^{51}
s (Approximately 50%):
- 斯: sie(心開三支平) -> sɿ^{55}
- 薩: sat(心開一寒入) -> sa^{51}
- 孫: son(心合一魂平) -> suən^{55} tɕ (Approximately 2%):
- 峻: swinh(心合三真去) -> tɕyn^{51}
ɕ (Approximately 47%):
- 性: siaengh(心開三清去) -> ɕiŋ^{51}
- 膝: sit(心開三真入) -> ɕi^{55}
- 選: swienq(心合三仙上) -> ɕyan^{214}
ts (Approximately 1%):
- 噪: sawh(心開一豪去) -> tsau^{51}

Based on the above examples, generate a rule function to determine the initial of characters with initial as “心” in Middle Chinese in Standard Chinese. Key points are as follows:

1. Only return the Python function code, with the function name as “聲母_心”, which takes a dictionary parameter ch containing the following phonetic information fields of the character: '聲母'(initial), '開合'(roundedness), '等'(division), '重紐'(chongniu), '韻'(rhyme), '調'(tone), '清濁'(voicing), '組'(initial group), '攝'(rhyme group), and returns the initial consonant of the character in Standard Chinese.
2. If additional judgments are needed based on the Standard Chinese initials, finals, and tones, you can also return a function named “聲母_心_現”, which takes the above parameter ch and a second parameter syl, containing the following three fields of the character in Standard Chinese: '聲母'(initial), '韻母'(final), '聲調'(tone), where '{phonetic part}'(initial) comes from the output of the function “聲母_心”, and the other two are the actual pronunciations. You can combine the information from ch and syl to make secondary modifications to the results to improve accuracy.
3. The final score has a length penalty, so keep the output concise. For obvious exceptions that do not conform to systematic sound change rules, treat them as irregular pronunciations and ignore them, without reflecting them in the function.
4. If there are random changes that cannot be further divided, you can return a list of multiple possible pronunciations. However, all test cases returned in that line will have a penalty factor of 0.8 for accuracy calculation, and the final score will deduct the product of the KL divergence between the true distribution and the uniform distribution and the proportion of correct test cases returned in that line.

Assistant Reasoning:
Okay, now I need to generate a rule function for the sound change of characters with initial as “心” in Middle Chinese to Standard Chinese initial. Based on the above examples, I can summarize the following rules for initial “心”:
- Normally, it changes to s (e.g., 斯(sie -> sɿ^{55}), 孫(son -> suən^{55}), 薩(sat -> sa^{51})).
- If the Standard Chinese final starts with i or y, it changes to ɕ (e.g., 選(swienq -> ɕyan^{214}), 膝(sit -> ɕi^{55}), 性(siaengh -> ɕiŋ^{51})).

In addition, the following examples seem irregular, and I should ignore them:

- 賜(sieh -> tsʰɿ^{51})
- 峻(swinh -> tɕyn^{51})
- 噪(sawh -> tsau^{51})

Next, I will write a Python function based on these rules to implement the corresponding judgment logic.

Assistant:

```
def 聲母_心(ch):
    return "s"
def 聲母_心_現(ch: CharInfo, syl):
    if syl["韻母"].startswith(("i", "y")):
    # if the final starts with i or y
        return "ɕ"
    # otherwise keep the original prediction
    return syl["聲母"]
```

D.3.2 Query Prompt for non-SFT Baselines

The prompt templates with example and translation are shown below. Since it's not used for SFT, there is no assistant output. The main difference between this template and the SFT template is that it provides more detailed explanations of the dictionary fields; specific examples are not given here.

Original Chinese Template

User:

下面是一些中古漢語中{phonetic part}為“{part name}”的字在{dialect name}中的讀音，按{phonetic part}分類:

{list of all possible pronunciations in the dialect with percentages}

根據上述示例和你自己掌握的知識，生成一個規則函數，用於判斷在中古漢語中{phonetic part}為“{part name}”的字在{dialect name}中的{phonetic part}。要點如下:

1. 只返回Python函數代碼，函數名稱為“{phonetic part}_{part name}”，函數接受一個字典參數ch，包含漢字的以下音韻信息字段：'聲母'(Literal['幫', '滂', '並', '明', '端', '透', '定', '泥', '知', '徹', '澄', '孃', '精', '清', '從', '心', '邪', '莊', '初', '崇', '生', '俟', '章', '昌', '常', '書', '船', '見', '溪', '羣', '疑', '影', '曉', '匣', '云', '來', '日', '以'])、'開合'(Literal['開', '合', ''], 脣音聲母和部分韻母不分開合爲'')、'等'(Literal['一', '二', '三', '四'])、'重紐'(Literal['A', 'B', 'C', ''])、'韻'(['東', '冬', '鍾', '江', '支', '脂', '之', '微', '魚', '虞', '模', '齊', '祭', '泰', '佳', '皆', '夬', '灰', '咍', '廢', '真', '臻', '文', '殷', '元', '魂', '痕', '寒', '刪', '山', '先', '仙', '蕭', '宵', '肴', '豪', '歌', '麻', '唐', '陽', '庚', '耕', '清', '青', '登', '蒸', '侯', '尤', '幽', '侵', '覃', '談', '鹽', '添', '咸', '銜', '嚴', '凡'], 以平概上去入)、'調'(Literal['平', '上', '去', '入'])、'清濁'(Literal['全清', '全濁', '次清', '次濁'])、'組'(Literal['幫', '端', '知', '精', '莊', '章', '見', '影', ''])、'攝'(Literal['通', '江', '止', '遇', '蟹', '臻', '山', '效', '果', '假', '宕', '梗', '曾', '流', '深', '咸'])，返回該字在{dialect name}中的{phonetic part}。

2. 如果需要根據{dialect name}的聲韻調進行額外的判斷，可以再返回一個名稱為“{phonetic part}_{part name}_現”的函數，接受上述參數ch外，還接受第二個參數syl，包含漢字在普通話中的'聲母'、'韻母'、'聲調'三個字段，其中'聲母'來自函數“{phonetic part}_{part name}”的輸出，另外兩項爲實際發音，你可以結合ch和syl的信息對結果進行二次修改，以提升正確率。

3. 最終評分存在長度懲罰，請保持輸出精簡，對於明顯不符合系統演變規則的個例，視爲不規則讀音，應該直接忽略，不在函數中體現。

4. 如果存在無法細分的隨機演變，你可以返回一個包含多個可能讀音的列表，但所有該行返回的測試用例計算正確率會有0.8的懲罰系數，並且在最終得分扣除真實分布與均勻分布的KL散度與在該行返回的正確測試用例占比的乘積。

English Translation of the Template

User:

Here are some characters with {phonetic part} as “{part name}” in Middle Chinese and their pronunciations in {dialect name}, categorized by {phonetic part}:

{list of all possible pronunciations in the dialect with percentages}

Based on the above examples and your own knowledge, generate a rule function to determine the {phonetic part} of characters with {phonetic part} as “{part name}” in Middle Chinese in {dialect name}. Key points are as follows:

1. Only return the Python function code, with the function name as “{phonetic part}_{part name}”, which takes a dictionary parameter ch containing the following phonetic information fields of the character: '聲母'(initial, Literal['幫', '滂', '並', '明', '端', '透', '定', '泥', '知', '徹', '澄', '孃', '精', '清', '從', '心', '邪', '莊', '初', '崇', '生', '俟', '章', '昌', '常', '書', '船', '見', '溪', '羣', '疑', '影', '曉', '匣', '云', '來', '日', '以']) , '開合'(roundedness, Literal['開', '合', ''], empty string for labial initial and some finals.) , '等'(division, Literal['一', '二', '三', '四']) , '重紐'(chongniu, Literal['A', 'B', 'C', '']) , '韻'(rhyme, Literal['東', '冬', '鍾', '江', '支', '脂', '之', '微', '魚', '虞', '模', '齊', '祭', '泰', '佳', '皆', '夬', '灰', '咍', '廢', '真', '臻', '文', '殷', '元', '魂', '痕', '寒', '刪', '山', '先', '仙', '蕭', '宵', '肴', '豪', '歌', '麻', '唐', '陽', '庚', '耕', '清', '青', '登', '蒸', '侯', '尤', '幽', '侵', '覃', '談', '鹽', '添', '咸', '銜', '嚴', '凡'], the *level* tone rhymes are used to represent the four tone rhymes.), '調'(tone, Literal['平', '上', '去', '入']) , '清濁'(voicing, Literal['全清', '全濁', '次清', '次濁']), '組'(initial group, Literal['幫', '端', '知', '精', '莊', '章', '見', '影', '']), '攝'(rhyme group, Literal['通', '江', '止', '遇', '蟹', '臻', '山', '效', '果', '假', '宕', '梗', '曾', '流', '深', '咸']), and returns the {phonetic part} of the character in {dialect name}.

2. If additional judgments are needed based on the {dialect name} initials, finals, and tones, you can also return a function named “{phonetic part}_{part name}_現”, which takes the above parameter ch and a second parameter syl, containing the following three fields of the character in Standard Chinese: '聲母'(initial), '韻母'(final), '聲調'(tone), where '{phonetic part}'(initial) comes from the output of the function “{phonetic part}_{part name}”, and the other two are the actual pronunciations. You can combine the information from ch and syl to make secondary modifications to the results to improve accuracy.

3. The final score has a length penalty, so keep the output concise. For obvious exceptions that do not conform to systematic sound change rules, treat them as irregular pronunciations and ignore them, without reflecting them in the function.
4. If there are random changes that cannot be further divided, you can return a list of multiple possible pronunciations. However, all test cases returned in that line will have a penalty factor of 0.8 for accuracy calculation, and the final score will deduct the product of the KL divergence between the true distribution and the uniform distribution and the proportion of correct test cases returned in that line.

D.4 Sentence to IPA Transcription

D.4.1 SFT Data with Middle Chinese Information

The prompt templates with example and translation are shown below.

Original Chinese Template

User:
結合音韻學知識，將下面的句子中的漢字轉換為{`dialect name`}讀音的國際音標表示，保留標點符號和非中文字符不變。
句子：{`sentence`}
Assistant Reasoning:
讓我逐字列出每個漢字的切韻拼音，結合漢字和切韻拼音，給出對應的{`dialect name`}讀音，保留標點符號和非中文字符不變：
for each character in the sentence
if the character is Chinese character
{`character`}|{`MC Tupa`}|{`dialect pronunciation`}
if the character is not Chinese character
{`character`}
接下來，我將根據列出的讀音，按照用戶要求的格式輸出。
Assistant:
{IPA representation of the sentence}

English Translation of the Template

User:
Using Chinese phonological knowledge, convert the Chinese characters in the following sentence into the International Phonetic Alphabet representation of {`dialect name`} pronunciation, while keeping punctuation marks and non-Chinese characters unchanged.
Sentence: {`sentence`}
Assistant Reasoning:
Let me list out the Middle Chinese Tupa for each character, and combine the character and Middle Chinese Tupa to give the corresponding {`dialect name`} pronunciation, while keeping punctuation marks and non-Chinese characters unchanged:
for each character in the sentence
if the character is Chinese character
{`character`}|{`MC Tupa`}|{`dialect pronunciation`}
if the character is not Chinese character
{`character`}
Next, I will output according to the format requested by the user based on the listed pronunciations.
Assistant:
{IPA representation of the sentence}

Example

User:
結合音韻學知識，將下面的句子中的漢字轉換為廣州話讀音的國際音標表示，保留標點符號和非中文字符不變。
句子：管道由多節圓形陶管連接構成。
Assistant Reasoning:
讓我逐字列出每個漢字的切韻拼音，結合漢字和切韻拼音，給出對應的廣州話讀音，保留標點符號和非中文字符不變：
管|kwanq|kun^{35}
道|dawq|tou^{22}
由|ju|jɐu^{21}
多|ta|tɔ^{55}
節|tset|tsit^{3}
圓|uen|jyn^{21}
形|gheng|jeŋ^{21}
陶|daw|tʰou^{21}
管|kwanq|kun^{35}
連|lien|lin^{21}
接|tsiep|tsip^{3}
構|kouh|kʰɐu^{33}
成|djiaeng|seŋ^{21}
。
接下來，我將根據列出的讀音，按照用戶要求的格式輸出。
Assistant:
kun^{35} tou^{22} jɐu^{21} tɔ^{55} tsit^{3} jyn^{21} jeŋ^{21} tʰou^{21} kun^{35} lin^{21} tsip^{3} kʰɐu^{33} seŋ^{21}。

English Translation of the Example

User:
Using Chinese phonological knowledge, convert the Chinese characters in the following sentence into the International Phonetic Alphabet representation of Guangzhou dialect (Cantonese) pronunciation, while keeping punctuation marks and non-Chinese characters unchanged.
Sentence: 管道由多節圓形陶管連接構成。
Assistant Reasoning:
Let me list out the Middle Chinese Tupa for each character, and combine the character and Middle Chinese Tupa to give the corresponding Guangzhou dialect (Cantonese) pronunciation, while keeping punctuation marks and non-Chinese characters unchanged:
管|kwanq|kun^{35}
道|dawq|tou^{22}
由|ju|jɐu^{21}
多|ta|tɔ^{55}
節|tset|tsit^{3}
圓|uen|jyn^{21}
形|gheng|jeŋ^{21}
陶|daw|tʰou^{21}
管|kwanq|kun^{35}
連|lien|lin^{21}
接|tsiep|tsip^{3}
構|kouh|kʰɐu^{33}
成|djiaeng|seŋ^{21}
。
Next, I will output according to the format requested by the user based on the listed pronunciations.

Assistant:
kun^{35} tou^{22} jɐu^{21} tɔ^{55} tsit^{3} jyn^{21} jeŋ^{21} tʰou^{21} kun^{35} lin^{21} tsip^{3} kʰɐu^{33} seŋ^{21}。

D.4.2 SFT Data without Middle Chinese Information

The prompt templates with example and translation are shown below.

Original Chinese Template

User:
將下面的句子轉換為{`dialect name`}讀音的國際音標表示，保留標點符號和非中文字符不變。
句子：{`sentence`}
Assistant:
{IPA representation of the sentence}

English Translation of the Template

User:
Convert the Chinese characters in the following sentence into the International Phonetic Alphabet representation of {`dialect name`} pronunciation, while keeping punctuation marks and non-Chinese characters unchanged.
Sentence: {`sentence`}
Assistant:
{IPA representation of the sentence}

Example

User:
將下面的句子轉換為廣州話讀音的國際音標表示，保留標點符號和非中文字符不變。
句子：管道由多節圓形陶管連接構成。
Assistant:
kun^{35} tou^{22} jɐu^{21} tɔ^{55} tsit^{3} jyn^{21} jeŋ^{21} tʰou^{21} kun^{35} lin^{21} tsip^{3} kʰɐu^{33} seŋ^{21}。

English Translation of the Example

User:
Convert the Chinese characters in the following sentence into the International Phonetic Alphabet representation of Guangzhou dialect (Cantonese) pronunciation, while keeping punctuation marks and non-Chinese characters unchanged.
Sentence: 管道由多節圓形陶管連接構成。
Assistant:
kun^{35} tou^{22} jɐu^{21} tɔ^{55} tsit^{3} jyn^{21} jeŋ^{21} tʰou^{21} kun^{35} lin^{21} tsip^{3} kʰɐu^{33} seŋ^{21}。

D.4.3 Query Prompt for non-SFT Baselines

The prompt templates with example and translation are shown below. Since it's not used for SFT, there is no assistant output.

Original Chinese Template

User:
已知{`dialect name`}的音系如下：
聲母：{`initials`}
韻母：{`finals`}
聲調：{`tones`}
請給出下列句子{`dialect name`}讀音的國際音標，音標間用空格分隔，非漢字的數字、標點等保持不變，僅考慮單字調，不考慮連讀變調，不要返回除此以外的任何內容：
{`sentence`}

English Translation of the Template

User:
Given the phonology of {`dialect name`} as follows:
Initials: {`initials`}
Finals: {`finals`}
Tones: {`tones`}
Please provide the International Phonetic Alphabet transcription of the following sentence in {`dialect name`} pronunciation, with spaces between syllables, keeping non-Chinese characters such as numbers and punctuation unchanged. Only consider single-character tones, without considering tone sandhi, and do not return any content other than that:
{`sentence`}

Example

User:
已知廣州話的音系如下：
聲母： j, t, tsʰ, ts, s, h, p, kʰ, l, k, n, w, f, ŋ, pʰ, kʷ, m, kʷʰ, tʰ
韻母： ɐt, eŋ, œŋ, am, a, ɔi, yn, ɛ, ɐi, ɐu, ip, oŋ, i, im, ɔŋ, ɐm, an, y, øy, ai, u, ɔk, yt, ap, in, iu, aŋ, ɐŋ, ŋ̍, ɛŋ, au, ek, ɐn, ɐp, ɐk, ei, ok, ui, øn, ak, un, ɔ, ou, at, øt, it, ɔn, œk, ɛk, ɔt, ut, m̩, œ
聲調： ^{5}, ^{55}, ^{22}, ^{23}, ^{35}, ^{33}, ^{2}, ^{21}, ^{3}
請給出下列句子廣州話讀音的國際音標，音標間用空格分隔，非漢字的數字、標點等保持不變，僅考慮單字調，不考慮連讀變調，不要返回除此以外的任何內容：
管道由多節圓形陶管連接構成。

English Translation of the Example

User:
Given the phonology of Guangzhou dialect (Cantonese) as follows:
Initials: j, t, tsʰ, ts, s, h, p, kʰ, l, k, n, w, f, ŋ, pʰ, kʷ, m, kʷʰ, tʰ
Finals: ɐt, eŋ, œŋ, am, a, ɔi, yn, ɛ, ɐi, ɐu, ip, oŋ, i, im, ɔŋ, ɐm, an, y, øy, ai, u, ɔk, yt, ap, in, iu, aŋ, ɐŋ, ŋ̍, ɛŋ, au, ek, ɐn, ɐp, ɐk, ei, ok, ui, øn, ak, un, ɔ, ou, at, øt, it, ɔn, œk, ɛk, ɔt, ut, m̩, œ
Tones: ^{5}, ^{55}, ^{22}, ^{23}, ^{35}, ^{33}, ^{2}, ^{21}, ^{3}
Please provide the International Phonetic Alphabet transcription of the following sentence in {`dialect name`} pronunciation, with spaces between syllables, keeping non-Chinese characters such as numbers and punctuation unchanged. Only consider single-character tones, without considering tone sandhi, and do not return any content other than that:
管道由多節圓形陶管連接構成。

E More Results

E.1 Panphon Feature Edit Distance

To better reflect phonetic naturalness and perceptual similarity between the generated pronunciations and the gold references in Task 2, we also compute the feature edit distance based on Panphon. The modified evaluation metric incorporates 22 articulatory features of the International Phonetic Alphabet, where closer articulatory features yield smaller substitution distances. The detailed evaluation results are presented in Table 4.

E.2 Statistical Significance

Table 5 and Table 6 present the statistical significance test results for Task 1 and Task 2, respectively, comparing our method with other ablated methods and baselines across different metrics.

For Task 1, our method shows significant advantages over most ablated methods in terms of "Accuracy per Rule," "Score per Rule," and "Accuracy per Character." It also significantly outperforms DeepSeek-V3.2 and ChatGPT-5.2 in the "Score" metric, consistent with the results in the main text. Although our method does not significantly outperform DeepSeek-V3.2 in "Accuracy per Rule," it does so in "Accuracy per Character." This indicates that the main reason for our superior performance over DeepSeek-V3.2 in the main text is that DeepSeek-V3.2 failed to generate some rules, leading to poor character coverage, although the rules it did generate performed well in terms of accuracy.

For Task 2, our method demonstrates significant advantages across different character sets using the edit distance to the standard answer as the metric for statistical significance testing. This is consistent with the results presented in the main text.

E.3 Case Study

E.3.1 Case Study for Task 1

Below are two case studies in Task 1, the *italicized parts* are translations of code comments:

Tone "入" (*ru sheng*, or checked tone) in Meixian Dialect

Example Chars

^{1}(Approximately 59%):
- 蟀: sruit(生合三真入) -> sut^{1}
- 壹: qit(影開三A真入) -> it^{1}
- 稷: tsyk(精開三蒸入) -> tsit^{1}
- 嚼: dzyak(從開三陽入) -> tsiok^{1}
- 迹: tsiaek(精開三清入) -> tsiak^{1}

^{5}(Approximately 38%):
- 逐: druk(澄三東入) -> tsʰuk^{5}
- 惑: ghweok(匣合一登入) -> fet^{5}
- 冽: liet(來開三仙入) -> liet^{5}
- 曆: lek(來開四青入) -> lak^{5}
- 顎: ngak(疑開一唐入) -> ŋok^{5}

^{53}(Approximately 2%):
- 蚱: tsraek(莊開二庚入) -> tsa^{53}
- 覡: ghek(匣開四青入) -> saŋ^{53}
- 窄: tsraek(莊開二庚入) -> tsa^{53}
- 臆: qyk(影開三C蒸入) -> i^{53}
- 笏: hot(曉合一魂入) -> fu^{53}

^{44}(Approximately 1%):
- 哈: ngop(疑開一覃入) -> ha^{44}
- 拉: lop(來開一覃入) -> la^{44}
- 砉: hwaek(曉合二庚入) -> kui^{44}
- 揖: qip(影開三A侵入) -> ia^{44}
- 凹: qeep(影開二咸入) -> au^{44}

^{31}(Approximately 1%):
- 岌: ngyip(疑開三B侵入) -> in^{31}

^{11}(Approximately 1%):
- 餄: keep(見開二咸入) -> i^{11}

ChatGPT-5.2 Generated Rule Function

```
def 調_入(ch):
    """
    判斷中古入聲字在梅縣話的入聲調類（不考慮舒化等現讀條件時的基礎規則）。
    主要依中古清濁分陰入/陽入:
      - 全清/次清 -> 1（陰入）
      - 全濁/次濁 -> 5（陽入）
    Determine the checked tone category of Middle Chinese checked tone characters in Meixian dialect (basic rules without considering current reading conditions like smoothing).
    Mainly divided into Yin Ru / Yang Ru based on Middle Chinese voicing:
      - Fully voiceless/Secondary voiceless -> 1 (Yin Ru)
      - Fully voiced/Secondary voiced -> 5 (Yang Ru)
    """
    if ch.get("清濁") in ("全濁", "次濁"):
        return "5"
    return "1"
def 調_入_現(ch, syl):
    """
    允許結合梅縣話現讀（是否仍為塞尾入聲等）對基礎結果做小幅修正。
    規則要點（偏保守）:
      1）若現讀仍有塞尾(-p/-t/-k/ʔ)，保持入聲基礎調(1/5)。
      2）若現讀已失去塞尾（入聲舒化），多數情況:
         - 陰入(1) -> 44
         - 陽入(5) -> 53
      3）少量較穩定的類型化微調:
         - 咸攝舒化：傾向 44
         - 深攝舒化且基礎為陽入：可見 31
         - 梗攝舒化：依韻母高元音(i/ui/ei等)略偏 44，否則偏 53
    Allow minor modifications to the base result based on the current Meixian reading (e.g., whether it still retains a checked coda).
    Rule highlights (conservative):
      1）If the current reading still retains a checked coda (-p/-t/-k/ʔ), keep the base checked tone (1/5).
```

Model	Task 2: Panphon Feature Edit Distance															
	Standard Chinese				Guangzhou				Shanghai				Chengdu			
	Full	Train	Test	Pred.	Full	Train	Test	Pred.	Full	Train	Test	Pred.	Full	Train	Test	Pred.
Qwen3-8B	10.23	1.82	1.88	1.86	28.30	1.80	1.76	1.82	31.77	1.82	1.83	1.85	21.49	2.18	2.11	2.09
Our (Full)	0.10	**0.01**	0.01	0.51	**0.21**	**0.01**	0.09	0.15	**0.39**	**0.03**	**0.04**	**0.31**	**0.41**	**0.04**	**0.01**	**0.39**
Our (w/o MC)	0.09	**0.01**	0.01	0.54	2.88	0.12	0.03	0.21	0.57	0.14	0.15	0.40	0.46	0.08	0.06	0.46
Our (w/o SCT)	0.11	**0.01**	**0.00**	0.47	0.28	0.05	**0.01**	0.16	0.42	0.06	0.11	0.40	0.46	0.09	0.04	0.41
DeepSeek-V3.2	0.14	**0.01**	0.01	**0.29**	0.39	0.12	**0.01**	**0.14**	0.69	0.13	0.06	0.39	0.50	0.09	0.13	**0.39**
ChatGPT-5.2	0.13	0.05	0.04	0.43	0.36	0.09	**0.01**	0.19	2.11	1.29	1.68	1.36	0.83	0.40	0.34	0.63
	Nanchang				Harbin				Meixian				Quanzhou			
Qwen3-8B	37.62	3.54	2.06	1.89	17.29	1.90	2.20	1.56	27.95	2.23	2.13	2.07	27.10	2.35	1.88	2.25
Our (Full)	**0.32**	**0.05**	**0.02**	**0.35**	**0.18**	**0.02**	**0.03**	**0.26**	**0.32**	**0.04**	**0.01**	0.20	**0.18**	**0.04**	**0.02**	0.43
Our (w/o MC)	0.37	0.09	0.11	0.51	0.18	**0.02**	0.08	0.27	0.38	0.08	0.21	0.31	0.27	0.11	0.14	0.55
Our (w/o SCT)	0.39	0.09	0.09	0.42	0.26	0.05	0.12	0.27	0.38	0.06	0.22	0.29	0.35	0.12	0.26	0.48
DeepSeek-V3.2	0.53	0.23	0.17	0.42	0.20	0.04	0.12	**0.26**	0.37	0.08	0.04	**0.19**	0.35	0.20	0.15	**0.40**
ChatGPT-5.2	3.31	2.67	3.04	2.46	0.84	0.30	0.24	0.43	1.14	0.72	1.08	0.71	1.34	0.80	0.83	0.93
	Luoyang				Jinan				Yinchuan				Changsha			
Qwen3-8B	28.97	1.72	1.79	1.85	18.98	1.94	2.05	1.99	22.48	1.75	2.09	1.75	41.72	2.03	2.13	1.97
Our (Full)	**0.28**	**0.01**	**0.01**	**0.29**	**0.23**	**0.02**	**0.03**	**0.49**	0.33	**0.02**	**0.01**	0.20	**0.29**	**0.02**	**0.01**	**0.43**
Our (w/o MC)	0.28	0.02	0.16	0.30	0.24	0.03	0.05	0.51	**0.31**	**0.02**	**0.01**	**0.19**	0.34	0.06	0.06	0.48
Our (w/o SCT)	0.30	0.03	0.18	0.27	0.42	0.11	0.08	0.61	0.38	0.07	0.04	0.24	0.66	0.22	0.12	0.52
DeepSeek-V3.2	0.35	0.10	0.25	0.36	0.41	0.15	**0.03**	0.61	0.41	0.09	0.07	0.23	0.69	0.24	0.23	0.56
ChatGPT-5.2	1.83	0.93	0.87	1.14	1.10	0.67	0.84	0.83	1.01	0.54	0.38	0.64	4.26	3.10	3.19	3.04

Table 4: Task 2 results evaluated by Panphon Feature Edit Distance. Lower is better. The data represents perceptual similarity.

```
        2) If the current reading has lost the checked coda (checked tone smoothing), in most cases:
            - Yin Ru (1) -> 44
            - Yang Ru (5) -> 53
        3) A few stable typological fine-tunings:
            - Xian rhyme group smoothing: tends towards 44
            - Shen rhyme group smoothing with Yang Ru base: can be 31
            - Geng rhyme group smoothing: slightly favors 44 if the final has a high vowel (i/ui/ei, etc.), otherwise favors 53
    """
    base = syl.get("聲調") or 調_入(ch)
    ym = (syl.get("韻母") or "").strip()
    def is_checked(rime: str) -> bool:
        if not rime:
            return True  # 無信息時不改(Do not change if there is no information)
        last = rime[-1]
        return last in ("p", "t", "k", "ʔ")
    if is_checked(ym):
        return base
    # 舒化（失塞尾）後的調類調整(Tone category adjustment after smoothing (loss of checked coda))
    she = ch.get("攝")
    # 咸攝舒化多歸高平一類（例：哈、拉、凹）(Xian rhyme group smoothing mostly falls into the high-level category (e.g., 哈、拉、凹))
    if she == "咸":
        return "44"
    # 深攝：陽入舒化可見較低曲折（例：岌）(Shen rhyme group: Yang Ru smoothing shows lower contours (e.g., 岌))
    if she == "深" and base == "5":
        return "31"
    # 梗攝：按韻母高元音/介音略分（例：硅 vs 蚱/窄/覫）(Geng rhyme group: slightly divided by high vowels/medials in the final (e.g., 硅 vs 蚱/窄/覫))
    if she == "梗":
        if ym.endswith(("i", "ui", "ei", "e", "ɿ", "ʅ")):
            return "44"
        return "53"
    # 曾攝（蒸等）舒化常見轉曲折（例：臆）(Zeng rhyme group (Zheng, etc.) smoothing commonly turns into contour tone (e.g., 臆))
    if she == "曾":
        return "53"
    # 臻攝（魂等）舒化可見轉曲折（例：笏）(Zhen rhyme group (Hun, etc.) smoothing can show contour changes (e.g., 笏))
    if she == "臻":
        return "53"
    # 一般情況：陰入舒化 -> 44；陽入舒化 -> 53(General case: Yin Ru smoothing -> 44; Yang Ru smoothing -> 53)
    return "44" if base == "1" else "53"
```

DeepSeek-V3.2 Generated Rule Function

```
def 調_入(ch):
    # 根据中古清浊和摄判断入声调(Determine checked tone based on Middle Chinese voicing and rhyme group)
    清浊 = ch['清濁']
    摄 = ch['攝']
    if 清浊 in ['全清', '次清']:
        return '1'
    elif 清浊 == '全濁':
        return '5'
```

Comparison	Accuracy per Rule		Score per Rule		Accuracy per Character	
	p-value	statistic	p-value	statistic	p-value	statistic
Our > Base	**5.97e-181**	6.08e+05	**2.48e-192**	7.04e+05	**0.0**	3.62e+09
Our > w/o Single Char	**9.53e-20**	9.86e+04	**4.27e-18**	1.26e+05	**1.43e-77**	4.98e+07
Our > w/o Joint Train	**1.84e-41**	1.66e+05	**2.02e-24**	4.68e+05	**0.0**	2.51e+08
Our > w/o GRPO	**2.48e-23**	1.16e+05	**3.29e-28**	1.62e+05	**0.0**	1.57e+08
Our > DeepSeek	1.00e+00	1.45e+05	**2.10e-47**	5.24e+05	**0.0**	2.31e+08
Our > ChatGPT	1.00e+00	1.00e+05	**1.57e-18**	4.57e+05	3.80e-01	8.94e+07

Table 5: Statistical significance test results for Task 1 using Wilcoxon signed-rank test between our full method and other ablated methods and baselines on different metrics. p-value 0.0 indicates that the value is too small to be represented with IEEE-754 double precision. With Bonferroni correction, p-values that less than $\alpha = \frac{0.05}{m} = \frac{0.05}{m} = 0.0083$ indicate that our method is significantly better than the compared method. Here, $m = 6$ is the number of comparisons.

Comparison	All Chars		Test Chars		Pred. Chars	
	p-value	statistic	p-value	statistic	p-value	statistic
Our (Full) > Base	**0.0**	8.54e+6	**3.36e-112**	6.75e+1	**6.48e-217**	2.67e+4
Our (Full) > Our (w/o MC)	**2.68e-182**	1.11e+6	**3.74e-19**	2.11e+3	**9.83e-28**	4.15e+4
Our (Full) > Our (w/o Single Char)	**7.61e-119**	1.01e+6	**1.80e-17**	1.89e+3	**1.47e-4**	6.11e+4
Our (Full) > DeepSeek-V3.2	**0.0**	2.43e+6	**1.22e-28**	3.32e+3	**3.15e-24**	8.47e+4
Our (Full) > ChatGPT-5.2	**0.0**	4.16e+6	**4.06e-77**	3.51e+3	**1.28e-119**	5.49e+4

Table 6: Significance Test Results for Task 2 (p-value and Significance Score) using Wilcoxon signed-rank test between our full method and other ablated methods and baselines on different character sets. p-value 0.0 indicates that the value is too small to be represented with IEEE-754 double precision. With Bonferroni correction, p-values that less than $\alpha = \frac{0.05}{m} = \frac{0.05}{5} = 0.01$ indicate that our method is significantly better than the compared method. Here, $m = 5$ is the number of comparisons.

```
    else:  # 次浊
        if 摄 in ['梗', '宕', '山']:
            return '5'
        else:
            return '1'
def 調_入_現(ch, syl):
    # 先获取基础声调(Get base tone first)
    基础调 = syl['聲调']
    韵母 = syl['韻母']
    # 检查是否保留塞音韵尾(Check if the checked
coda is retained)
    if 韵母 and 韵母[-1] in 'ptk':
        return 基础调
    else:  # 舒化(Smoothing)
        摄 = ch['攝']
        if 摄 in ['咸', '深']:
            return '44'
        else:
            return '53'
```

Rule Function Generated by Our Method

```
def 調_入(ch):
    if ch["清濁"] in ["全濁", "次濁"]:
        return "5"
    return "1"
```

Initial “透” (*tou*) in Luoyang Dialect

Example Chars

t^h(Approximately 92%):
- 捅: thoungq(透一東上) -> $t^hʊŋ^{54}$
- 眺: thewh(透開四蕭去) -> $t^hiɔ^{31}$
- 退: thojh(透合一灰去) -> t^huei^{31}
- 毯: thamq(透開一談上) -> t^han^{54}
- 妥: thwaq(透合一歌上) -> t^huo^{54}

t(Approximately 7%):
- 搭: thap(透開一談入) -> ta^{34}
- 貸: theojh(透開一咍去) -> fet^5
- 叨: thaw(透開一豪平) -> $tɔ^{34}$

x(Approximately 3%):
- 瘓: thwanq(透合一寒上) -> $xuan^{31}$

ChatGPT-5.2 Generated Rule Function

```
def 聲母_透(ch):
    """
    判斷中古「透」母在洛陽話的聲母。
    以塞送氣 tʰ 為主；少量在合口一等山攝寒韻出
現擦化 x（如 xuan）。
    Determine the initial of Middle Chinese ”Tou” ini-
tial in Luoyang dialect.
    Mainly aspirated stop tʰ; a small amount of frica-
tivization x appears in closed mouth, division I, Shan
rhyme group, Han rhyme (e.g., xuan).
    """
    if ch.get('聲母') != '透':
        return None
    if (ch.get('開合') == '合'
            and ch.get('等') == '一'
```

Example of Chengdu Dialect Inference

Sentence	今	日	西	泠	拚	一	慟	，
Standard Chinese Reference	tɕin^{55}	ɻ̩^{51}	ɕi^{55}	liŋ^{35}	pʰin^{55}	i^{55}	tʰoŋ^{51}	，
Middle Chinese	kyim	njit	sej	leng	pheng	qit	doungh	，
Gold	tɕin^{45}	zɿ^{21}	ɕi^{45}	nin^{21}	pʰin^{45}	i^{21}	tʰoŋ^{213}	，
Our	**tɕin^{45}**	**zɿ^{21}**	**ɕi^{45}**	**nin^{21}**	pʰin^{21}	**i^{21}**	**tʰoŋ^{213}**	，
DeepSeek-V3.2	**tɕin^{45}**	zɿ^{213}	**ɕi^{45}**	lin^{21}	pʰan^{42}	ȵi^{213}	**tʰoŋ^{213}**	，
ChatGPT-5.2	**tɕin^{45}**	zɿ^{42}	**ɕi^{45}**	**nin^{21}**	**pʰin^{45}**	i^{45}	tʰoŋ^{42}	，
Sentence	不	堪	重	唱	寶	刀	歌	。
Standard Chinese Reference	pu^{51}	kʰan^{55}	tʂuŋ^{35}	tʂʰaŋ^{51}	pau^{214}	tau^{55}	kɤ^{55}	。
Middle Chinese	put	khom	druong	tjhyangh	pawq	taw	ka	。
Gold	pu^{21}	kʰan^{45}	tsʰoŋ^{21}	tsʰaŋ^{213}	pau^{42}	tau^{45}	ko^{45}	。
Our	**pu^{21}**	**kʰan^{45}**	**tsʰoŋ^{21}**	**tsʰaŋ^{213}**	**pau^{42}**	**tau^{45}**	**ko^{45}**	。
DeepSeek-V3.2	pu^{213}	**kʰan^{45}**	**tsʰoŋ^{21}**	**tsʰaŋ^{213}**	**pau^{42}**	**tau^{45}**	**ko^{45}**	。
ChatGPT-5.2	pu^{42}	**kʰan^{45}**	**tsʰoŋ^{21}**	tsʰaŋ^{42}	pau^{213}	**tau^{45}**	**ko^{45}**	。

Example of Nanchang Dialect Inference

Sentence	黃	庭	堅	曾	批	評	蘇	軾	說
Standard Chinese Reference	xuaŋ^{35}	tʰiŋ^{35}	tɕiɛn^{55}	tsʰəŋ^{35}	pʰi^{55}	pʰiŋ^{35}	su^{55}	ʂɿ^{51}	ʂuo ^{55}
Middle Chinese	ghwang	deng	ken	dzeong	phej	byaeng	so	sjyk	sjwiet
Gold	uɔŋ^{45}	tʰin^{24}	tɕiɛn^{42}	tsʰɛn^{24}	pʰi^{42}	pʰin^{24}	su^{42}	sɨt^{5}	sot^{5}
Our	fɔŋ^{45}	**tʰin^{24}**	**tɕiɛn^{42}**	**tsʰɛn^{24}**	**pʰi^{42}**	pʰin^{42}	**su^{42}**	**sɨt^{5}**	**sot^{5}**
DeepSeek-V3.2	uɔŋ^{24}	**tʰin^{24}**	**tɕiɛn^{42}**	tsɛn^{42}	**pʰi^{42}**	**pʰin^{24}**	**su^{42}**	**sɨt^{5}**	ɕyot^{5}
ChatGPT-5.2	huaŋ^{24}	**tʰin^{24}**	tɕiɛn^{5}	tsʰon^{24}	pʰi^{5}	**pʰin^{24}**	su^{5}	ɕɿ^{42}	ɕuo^{5}

Table 7: Case study of Task 2 (Middle Chinese to Chengdu Dialect pronunciation).

```
        and ch.get('攝') == '山'
        and ch.get('韻') == '寒'):
    return "X"
return "tʰ"
```

DeepSeek-V3.2 Generated Rule Function

```
def 聲母_透(ch):
    """
    根據中古音韻信息判斷透母在洛陽話中的聲
母。默認返回tʰ，少數例外根據音韻條件處理。
    Determine the initial of the Tou initial in Luoyang
dialect based on Middle Chinese phonological informa-
tion. Default returns tʰ, a few exceptions are handled
based on phonological conditions.
    """
    # 提取音韻特徵(Extract phonological features)
    開合 = ch.get('開合', '')
    等 = ch.get('等', '')
    韻 = ch.get('韻', '')
    調 = ch.get('調', '')
    # 例外規則（基於示例總結）(Exception rules
(summarized based on examples))
    # 1. 讀 t 的例外(Exceptions for reading t)
    if 攝 == '咸' and 開合 == '開'
            and 等 == '一' and 調 == '入'
            and 韻 == '談':
        return "t"
    if 攝 == '蟹' and 開合 == '開'
            and 等 == '一' and 調 == '去'
            and 韻 == '咍':
        return "t"
    if 攝 == '效' and 開合 == '開'
            and 等 == '一' and 調 == '平'
            and 韻 == '豪':
        return "t"
    # 2. 讀 x 的例外(Exceptions for reading x)
    if 攝 == '山' and 開合 == '合'
            and 等 == '一' and 調 == '上'
            and 韻 == '寒':
        return "X"
    # 默認規則(Default rule)
    return "tʰ"
```

Rule Function Generated by Our Method

```
def 聲母_透(ch):
    return "tʰ"
```

Despite we explicitly emphasized in the instructions to generate concise rules that ignore phonologically irregular cases, the rule function generated by ChatGPT-5.2 is still relatively complex, containing a lot of logic to handle irregular characters. This is why it performs well on the Accuracy metric, but at the same time leads to a lower Score metric compared to our complete method.

Another noteworthy point is that DeepSeek-V3.2 tends to use Simplified Chinese (e.g., "韻" → "韵", "攝" → "摄") during generation, rather than the Traditional Chinese used in the instructions, which sometimes affects the correct parsing and execution of the rules, thereby impacting the accuracy of the final results.

E.3.2 Case Study for Task 2

In Chengdu Dialect inference, DeepSeek-V3.2 considers Standard Chinese as the basis, mapping Standard Chinese tones ^{55}, ^{35}, ^{214}, and ^{51} to Chengdu Dialect tones ^{45}, ^{21}, ^{42}, and ^{213} respectively. However, for characters with checked tones (or 入聲, *ru sheng*) in Middle Chinese (i.e., characters ending with -p, -t, -k, such as "日" and "一"), the tone in Chengdu Dialect typically changes to ^{21}, which differs from the evolution rules in Standard Chinese and cannot be directly derived from the mapping. Therefore, DeepSeek-V3.2 makes errors when handling checked tone characters. In Nanchang Dialect inference, since we indicated in the prompt that there are syllables ending with -t in Nanchang Dialect, DeepSeek correctly handles checked tone characters to some extent.

ChatGPT-5.2 also primarily relies on Standard Chinese for mapping. When handling tones, it tends to choose tone contours that are as similar as possible to those in Standard Chinese, rather than following the actual dialect pronunciation for mapping. For example, in Chengdu Dialect, it maps ^{214} to ^{213} (both are falling-rising tones) and ^{51} to ^{42} (both are falling tones), and in Nanchang Dialect, it maps ^{55} to ^{5} (both are high-level tones). This is one of the reasons why ChatGPT-5.2 performs poorly on this task.