Enhancing Character-Level Understanding in LLMs through Token Internal Structure Learning

Zhu Xu♣, Zhiqiang Zhao♣†, Zihan Zhang♣, Yuchi Liu♣, Quanwei Shen♣, Fei Liu♣,
Yu Kuang♣, Jian He♣, Conglin Liu♠

♣ School of Computer Science and Technology,
Chongqing University of Posts and Telecommunications

s231231076@stu.cqupt.edu.cn, † zhaozq@cqupt.edu.cn, s2312310{91, 46, 51, 42, 31, 20}@stu.cqupt.edu.cn
♠ Baidu AI Platform & Ecosystem

liuconglin@baidu.com

Abstract

Tokenization methods like Byte-Pair Encoding (BPE) enhance computational efficiency in large language models (LLMs) but often obscure internal character structures within tokens. This limitation hinders LLMs' ability to predict precise character positions, which is crucial in tasks like Chinese Spelling Correction (CSC) where identifying the positions of misspelled characters accelerates correction processes. We propose **Token Internal Position Awareness (TIPA)**, a method that significantly improves models' ability to capture character positions within tokens by training them on reverse character prediction tasks using the tokenizer's vocabulary. Experiments demonstrate that TIPA enhances position prediction accuracy in LLMs, enabling more precise identification of target characters in original text. Furthermore, when applied to downstream tasks that do not require exact position prediction, TIPA still boosts performance in tasks needing character-level information, validating its versatility and effectiveness.

1 Introduction

Large language models (LLMs) have revolutionized natural language processing by employing tokenization methods such as Byte-Pair Encoding (BPE) (Sennrich, 2015; Wang et al., 2020) to segment text into subword units, optimizing computational efficiency. However, BPE often obscures internal character structures within tokens (Shin and Kaneko, 2024; Xu and Ma, 2024), which poses challenges for tasks requiring detailed character-level information.

For instance, LLMs frequently struggle with simple character-counting tasks. When prompted with questions like **"How many r's are in**

(a) The internal information of tokens in large language models is unclear.

(b) LLM has fully learned the internal character composition information of tokens.
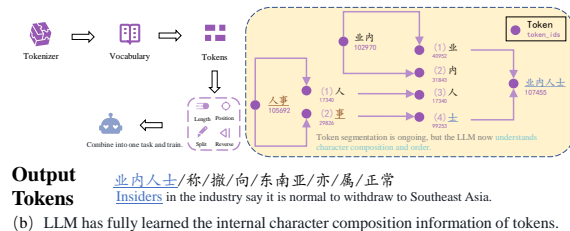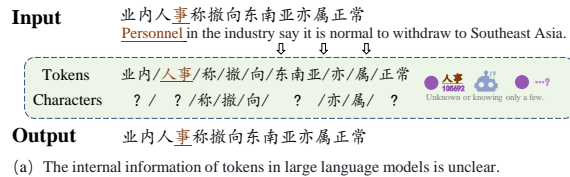
Figure 1: (a) demonstrates LLMs' inability to perform spelling correction correctly without learning token-internal character order. (b) shows TIPA's core dataset construction (left) and its character-level task enhancement through token structure understanding without architectural changes (right).

'strawberry'?" (Xu and Ma, 2024), many models fail to provide the correct answer due to their limited understanding of character positions within tokens, this limitation is more pronounced in languages like Chinese, where meaning relies heavily on character composition and sequence. Models such as GPT-4o (Hurst et al., 2024) often misidentify specific character positions in the tokenized text. For example, when asked to locate the character "阁" in the sentence "为什么总称呼对方为阁下？" (Why do you always address each other as 'Your Excellency'?) (Wu et al., 2023), they frequently provide incorrect positions.

This lack of internal character structure awareness adversely affects LLM performance in character-sensitive applications like Chinese Spelling Correction (CSC), where accurate identification of misspelled characters and their positions is crucial for efficient corrections.

Traditional Transformer-based (Vaswani, 2017) language models emphasize next-token prediction,

---

Task Type	Example
Source	(…106 chars)网 路 技术有限公司
Traditional Task	(…106 chars)网 络 技术有限公司
Position Task	[{108, 路 , 络 }]

Table 1: Accurately predicting the position of erroneous characters and providing both the incorrect and corrected characters serves two purposes: the incorrect character verifies the model's ability to precisely locate errors, while the corrected character fulfills the error correction task. This approach also reduces the number of output tokens required by the model.

focusing on sequential dependencies between tokens. This focus does not inherently encourage models to capture detailed positional relationships within tokens, leading them to rely more on token order rather than internal character composition.

To address this limitation, we propose **Token Internal Position Awareness (TIPA)**, a method designed to enhance models' ability to recognize and predict character positions within tokens. TIPA trains LLMs on reverse character prediction tasks using the tokenizer's vocabulary, compelling the model to focus on each character's position independently of sequential context.

Figure 1 illustrates the disparity in CSC performance between untrained and trained LLMs regarding token character composition and sequence.

For example, a token like "小说" (novel) would be decomposed in TIPA as a JSON structure: {2: "说", 1: "小"}, mapping each character to its position in descending order. This approach helps the model develop a structural understanding beyond typical left-to-right reading, which is crucial for tasks requiring precise character positioning.

TIPA leverages the tokenizer's own vocabulary, allowing the model to internalize character composition and structure without relying on external data, enhancing generalization in position-sensitive tasks.

Our contributions are:

1. **Enhanced Position Prediction:** Demonstrating the value of accurate character position prediction in CSC tasks, enabling faster and more precise corrections (see Table 1).

2. **Introduction of TIPA and MTIPA:** Presenting TIPA and its extension, Multi-Token Internal Position Awareness (MTIPA), which improves models' ability to capture character positions for accurate predictions.

3. **Versatility in Downstream Tasks:** Showing that TIPA enhances performance in tasks requiring character-level information, even without explicit position prediction.

2 Related Work

Tokenization methods like BPE (Sennrich, 2015; Wang et al., 2020) and WordPiece (Schuster and Nakajima, 2012) improve computational efficiency in LLMs but obscure internal character structures. Kaushal and Mahowald (2022) found that while larger models encode character-level details better, they may not explicitly understand character positions within tokens. Recent byte-level models like ByT5 (Xue et al., 2022) process raw bytes for character-level precision but require architectural changes that prevent low-cost adaptation of existing subword-based LLMs. Hybrid approaches (e.g., CANINE (Clark et al., 2022)) improve character awareness but still lack efficient positional modeling for multi-character tokens.

Recent studies (Xu and Ma, 2024; Shin and Kaneko, 2024) highlight LLMs' limitations in tasks requiring fine-grained character-level understanding, attributing deficiencies to tokenization and model architecture.

In CSC research, methods like ReLM (Liu et al., 2024) reframed CSC as sentence rephrasing, while self-supervised learning approaches (Jiang et al., 2024) showed that models trained on error-free data can outperform those using confusion sets. Li et al. (2024) proposed C-LLM, using character-level tokenization to enhance character-level understanding.

To enhance models' awareness of internal token structures, studies have addressed limitations like the "reversed curve phenomenon" (Berglund et al., 2023; Thawani et al., 2023) and sensitivity to text order (Chen et al., 2024). Itzhak and Levy (2022) found that while models encode orthographic information without direct character-level training, explicitly teaching spelling did not enhance performance. Our work differs by incorporating character position information and reversing character sequences during training, enabling a better understanding of internal token structures and improving tasks like Chinese Spelling Correction.

3 Methodology

We introduce two novel techniques: **Token Internal Position Awareness (TIPA)** and **Multi-Token Internal Position Awareness (MTIPA)**, designed

to enhance large language models' capacity to recognize and leverage internal character structures within tokens.

3.1 Token Internal Position Awareness (TIPA)

TIPA leverages the tokenizer's vocabulary to train the model to understand the internal structure of each token. For tokens that can be fully represented in UTF-8 (Yergeau, 2003), we apply a reverse prediction task to capture token-internal positions.

Let T denote the tokenizer, and let $V = \{t_1, t_2, \ldots, t_m\}$ be the set of tokens in the vocabulary of T. For each token $t \in V$ that can be fully represented in UTF-8, we decompose t into its constituent characters:

$$C_t = [c_1, c_2, \ldots, c_n], \qquad (1)$$

where n is the number of characters in t.

We define a reverse position mapping D_t for token t as:

$$D_t = \{(i, c_i) \mid i = n, n - 1, \ldots, 1\}. \qquad (2)$$

This mapping associates each position i (starting from n) with the character at the i-th position in t, effectively reversing the order of characters. The training prompt template used for this purpose is referenced in Table 2.

Algorithm 1 TIPA Algorithm

Require: Tokenizer T
Ensure: TIPA Dataset $\mathcal{D}_{\text{TIPA}}$

1: Initialize $\mathcal{D}_{\text{TIPA}} \leftarrow \emptyset$
2: $V \leftarrow \text{GetVocabulary}(T)$
3: **for** each token $t \in V$ **do**
4:    **if** t can be fully represented in UTF-8 **then**
5:       Decompose t into characters $C_t = [c_1, c_2, \ldots, c_n]$
6:       Create reverse position mapping $D_t = \{(i, c_i) \mid i = n, n - 1, \ldots, 1\}$
7:       Add (t, D_t) to $\mathcal{D}_{\text{TIPA}}$
8:    **end if**
9: **end for**
10: Prune irrelevant tokens from $\mathcal{D}_{\text{TIPA}}$
11: **return** $\mathcal{D}_{\text{TIPA}}$

Rationale for Reverse Ordering: By using reverse ordering, the first number output by the model corresponds to the length of the token (n). This approach integrates the token splitting task, length information, and position information into a single method. If we were to use forward ordering (i.e.,

positions starting from 1), the model might deduce the length of the token indirectly through the sequence of positions (1, 2, 3, etc.), but it wouldn't inherently know this information. Reverse ordering requires the model to output the token length as the starting position, avoiding this ambiguity.

This method retains the advantages of tokenization while enhancing the model's grasp of character composition and positional information within tokens. It slightly increases the training time due to the additional reverse prediction task but does not introduce any latency during inference.

The TIPA dataset $\mathcal{D}_{\text{TIPA}}$ is then constructed as:

$$\mathcal{D}_{\text{TIPA}} = \{(t, D_t) \mid t \in V \text{ (UTF-8)}\}. \qquad (3)$$

An overview of TIPA is illustrated in Figure 2.

TIPA Prompt Example

Instruction: 直接给出json输出，倒序给出输入的Token中包含的所有位置和字符
(Translation: Directly output the JSON, listing all characters and their positions in reverse order from the input token.)
Input: girl
Output: { "4": "**l**", "3": "**r**", "2": "**i**", "1": "**g**"}

Table 2: An example of the prompt (Zheng et al., 2024) used in TIPA($\leftarrow$) training, along with its English translation.

3.2 Multi-Token Internal Position Awareness (MTIPA)

Building upon TIPA, we propose **Multi-Token Internal Position Awareness (MTIPA)** to enhance the model's understanding of character positions within entire sentences or multi-token sequences, especially for tasks that require precise prediction of character positions.

In MTIPA, instead of focusing on individual tokens, we extend the reverse character prediction task to full sentences sampled from the training dataset. This allows the model to learn character positions in the broader context of sentences.

Specifically, we randomly sample a subset of sentences from the target task's training dataset. For each sampled sentence, we decompose it into its constituent characters and create a reverse position mapping, similar to TIPA but applied to the entire sentence.

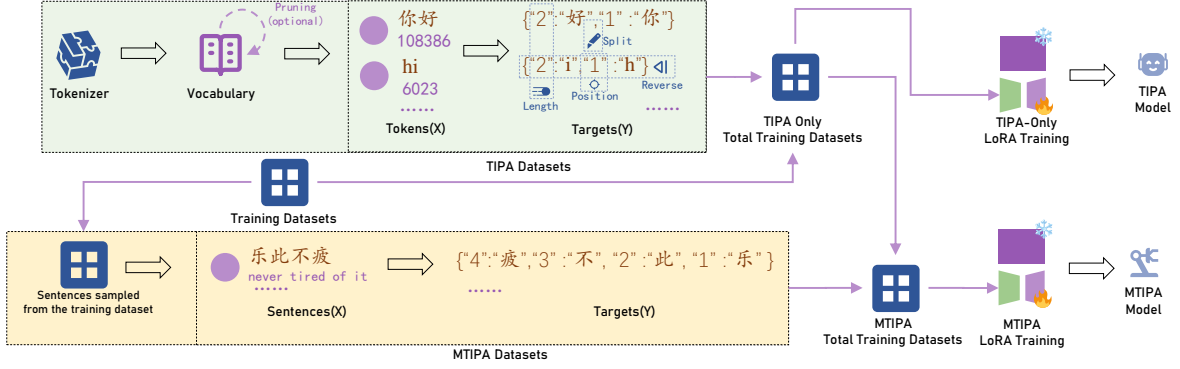An overview of MTIPA is illustrated in Figure 2.

Figure 2: Overview of TIPA and MTIPA. TIPA enhances character-level structure awareness per token. MTIPA extends this to multi-token sequences, enabling fine-grained positional understanding.

Algorithm 2 MTIPA Algorithm

Require: Training dataset $\mathcal{D}_{\text{train}}$, sampling ratio r
Ensure: MTIPA dataset $\mathcal{D}_{\text{MTIPA}}$
1: Initialize $\mathcal{D}_{\text{MTIPA}} \leftarrow \emptyset$
2: Randomly sample a subset $\mathcal{S} \subset \mathcal{D}_{\text{train}}$ with sampling ratio r
3: **for** each sentence $s \in \mathcal{S}$ **do**
4:     Decompose s into characters $C_s = [c_1, c_2, \ldots, c_n]$
5:     Create reverse position mapping $D_s = \{(i, c_i) \mid i = n, n-1, \ldots, 1\}$
6:     Add (s, D_s) to $\mathcal{D}_{\text{MTIPA}}$
7: **end for**
8: **return** $\mathcal{D}_{\text{MTIPA}}$

In practice, we set the sampling ratio r to a small value (e.g., 10%) to balance the amount of additional data and training efficiency.

MTIPA is specifically applied in tasks that require precise character position prediction within sentences. In our experiments, MTIPA is used in Experiment 1, which involves Chinese Spelling Correction (CSC) with position prediction. In Experiment 2, which focuses on the traditional CSC task without position prediction, MTIPA is not used; only TIPA is applied. The MTIPA dataset is too long, which can lead to a long training time, and if LoRA training uses a large amount of information to learn how to infer length information, it may reduce the model's ability to perform specific tasks.

By integrating MTIPA into the training process, the model gains a deeper understanding of character positions in multi-token sequences, leading to improved performance in tasks that demand precise position awareness.

3.3 Extended Methodology with Full-Parameter SFT

The tulu-3-sft-mixture dataset(Lambert et al., 2024) developed by AI2 enables base LLM refinement into conversational AI. We extracted vocabulary from Llama-3.1(Grattafiori et al., 2024) tokenizer, integrated all tokens into TIPA, and merged this with tulu-3-sft-mixture for full-parameter supervised fine-tuning (SFT) of Llama-3.1-8B. This produced **Llama-3.1-Tulu-TIPA-8B** without requiring additional character-level task datasets, maintaining the model's inherent capabilities while enhancing character-level processing through the integrated tokenizer vocabulary.

4 Redefining the Chinese Spelling Correction Task

Dataset	Traditional	Position
Train	8,905,800	8,016,111
CSCD-Dev	188,362	55,449
CSCD-Test	188,310	54,897
Lemon	532,684	258,112

Table 3: Comparison of Output Token Counts between Traditional and Position-based Methods for Various Datasets.

We redefine the Chinese Spelling Correction (CSC) task to require the model to output the positions of incorrect characters along with their corrections. For example, the sentence "业内人事称撤向东南亚亦属正常" (translation: "Industry insiders say that withdrawing to Southeast Asia is also normal") becomes:

["position": 4, "incorrect": "事", "correction": "士"]

Here, the character at position 4, "事", is incorrect and should be corrected to "士".

As shown in Table 3, the position-based method results in fewer output tokens compared to the traditional method, highlighting its efficiency in this task.

To evaluate model performance under this new framework, we introduce several metrics.

4.1 Position Prediction Accuracy (PPA)

Let the source text be $\mathbf{x} = [x_1, x_2, \ldots, x_L]$.

When calculating this metric, the characters corrected by the model are not considered, only its ability to perceive locations is evaluated.

Let the model's predicted set of incorrect positions and characters be:

$$\hat{P} = \{\, (\hat{i}, \hat{c}) \,\}$$

where $\hat{i}$ is the predicted position, and $\hat{c}$ is the predicted incorrect character at that position.

Position Prediction Accuracy (PPA) is defined as:

$$\text{PPA} = \frac{\left| \{\, \hat{c} = x_{\hat{i}} \mid (\hat{i}, \hat{c}) \in \hat{P} \,\} \right|}{|\hat{P}|}$$

That is, PPA measures the proportion of positions predicted by the model where the predicted incorrect character $\hat{c}$ matches the character $x_{\hat{i}}$ at position $\hat{i}$ in the source text $\mathbf{x}$. The denominator $|\hat{P}|$ is the total number of positions predicted by the model.

4.2 Sentence-Level Accuracy (SA)

Sentence-Level Accuracy (SA) measures the proportion of sentences where all predicted corrections are entirely accurate, including both the positions and the corrected characters.

4.3 Sentence-Level Accuracy Ignoring Position (SAIP)

Sentence-Level Accuracy Ignoring Position (SAIP) calculates the proportion of sentences where all predicted corrections are accurate, regardless of whether the positions were correctly identified.

4.4 Non-Empty Sample Sentence Accuracy (NESSA)

Non-Empty Sample Sentence Accuracy (NESSA) evaluates the model's performance on sentences known to contain errors by calculating the proportion of such sentences where all predicted corrections are entirely accurate.

4.5 Character-Level Precision (CP), Recall (CR), and F1 Score (CF1)

At the character level (Hu et al., 2024):

- **Precision (CP)** is the proportion of incorrect characters identified by the model that is actually incorrect.

- **Recall (CR)** is the proportion of actual incorrect characters that are correctly identified by the model.

- **F1 Score (CF1)** is the harmonic mean of Precision and Recall.

5 Experiments

We conducted comprehensive experiments to validate the effectiveness of TIPA and MTIPA. Our experiments are divided into three main parts: position-aware CSC, traditional CSC, and general model training and evaluation. Below, we detail the datasets, baselines, and training protocols.

5.1 Datasets

Following the method of C-LLM (Li et al., 2024, 2022; Liang et al., 2023), we selected two new Chinese Spelling Correction (CSC) benchmarks, CSCD-NS and LEMON, to address the limitations identified (Hu et al., 2024; Yin and Wan, 2023; Li et al., 2022) in previous datasets like SIGHAN (Wu et al., 2013; Yu and Li, 2014; Tseng et al., 2015; Sun et al., 2024). Additionally, we incorporated a large-scale pseudo-data set, Wang271K, generated by ASR or OCR methods, to enhance our training set. The datasets used for training and evaluation are as follows:

- **Wang271K** (Wang et al., 2018): A dataset containing 271,329 sentences with errors introduced based on linguistic rules. This dataset was used in combination with CSCD-NS for training.

- **CSCD-NS** (Hu et al., 2024): A high-quality CSC dataset where the primary source of character errors stems from pinyin input methods. It contains a significant amount of homophonic and word-level errors, making it superior to SIGHAN. The validation data from CSCD-NS was used as our validation set.

- **LEMON** (Wu et al., 2023): A novel, large-scale, multi-domain CSC dataset featuring

various real-world spelling errors. It spans seven different sub-domains, including game (GAM), encyclopedia (ENC), contract (COT), medical care (MEC), car (CAR), novel (NOV), and news (NEW), typically testing the model's domain correction capabilities in a zero-shot setting.

The training set was composed of the combined data from CSCD-NS and Wang271K. The validation set was derived from CSCD-NS, and the models were tested on both the CSCD-NS test data and LEMON.

For **Experiment 1**, we combined the training sets of Wang271K and CSCD-NS as our training data. The validation set was the CSCD-NS validation set. We tested our models on the CSCD-NS test set. (Li et al., 2024)

For **Experiment 2**, we followed the same data split but focused on the traditional CSC task without explicit position prediction. In addition, we also tested the model on the Lemon dataset.

For **general model** training, we used **tulu-3-sft-mixture**, an open-source conversational AI fine-tuning dataset for the supervised fine-tuning (SFT) stage(Lambert et al., 2024). We performed full-parameter supervised fine-tuning of Llama-3.1-8B using the combined tulu-3-sft-mixture and TIPA dataset. The TIPA dataset was constructed by extracting vocabulary from the Llama-3.1 tokenizer and integrating all tokens into the TIPA format.

For general model evaluation, we used the following benchmarks: **IFEVAL**(Zhou et al., 2023) (instruction following evaluation with strict/loose scoring), **GSM8K** (Cobbe et al., 2021) (math word problem-solving), **MMLU**(Hendrycks et al., 2020) (massive multitask language understanding), **AExams**(Hardalov et al., 2020) (Arabic exam question answering), **KoBEST**(Jang et al., 2022) (Korean language understanding benchmark), and **HumanEval**(Chen et al., 2021) (Python code generation), **TyDi QA**(Clark et al., 2020) (Multilingual QA benchmark). Inspired by LLM The Genius Paradox(Xu and Ma, 2024), we developed multilingual datasets covering three tasks: Occurrence, Length, and Distinct.

5.2 Baseline Models

We compared our methods against several baseline models, including Pure-SFT, GPT-4o (Hurst et al., 2024), DeepSeek v2.5, ERNIE-4.0, and GLM-4-Plus, to assess the relative performance improve-

ments. For general model evaluation, we used Llama-3.1-Tulu as a baseline.

5.3 Training Details

In Experiment 1 and Experiment 2, we fine-tuned the open-source Qwen2.5-7B (Yang et al., 2024) model using LoRA (Hu et al., 2021) to incorporate TIPA and MTIPA. LoRA allows efficient adaptation of large models without modifying the original model weights, making it suitable for our experiments.

For TIPA integration, we generated a TIPA dataset by performing set deduplication on tokens appearing in Wang271K, CSCD-NS, and LEMON to reduce the number of tokens for TIPA operations. This does not mean that preference learning has been applied to the dataset, because the result of pruning more than 300,000 pieces of data is only 24,994 tokens, which is sufficient to include all Chinese character tokens. This can be considered as a preference for Chinese characters, which is also a learning strategy. In practical applications, we can directly perform TIPA on all tokens of the tokenizer.

For MTIPA integration, we randomly sampled 10% of the training set and constructed the MTIPA dataset using the same method as TIPA but applied to multi-token sequences.

In Experiment 3, we extracted all tokens from the Llama-3.1 tokenizer vocabulary, performed TIPA operations to generate a dataset, and conducted mixed training with the tulu-3-sft-mixture dataset. Additional training details, hardware configurations, and training time can be found in Appendix A.2.

5.4 Experiment 1: CSC with Position Prediction

In this experiment, we evaluated the models on the redefined CSC task requiring position prediction.

5.4.1 Results

Table 4 presents the performance comparison. Our TIPA-7B model outperformed the baseline Pure-SFT-7B across all metrics, demonstrating the effectiveness of TIPA in enhancing position awareness. The MTIPA-7B model further improved performance, indicating that incorporating multi-token sequences benefits the model's understanding of character positions.

Model	PPA (%)	SA (%)	SAIP (%)	NESSA (%)	CP (%)	CR (%)	CF1 (%)
Qwen2.5-7B	4.03	32.76	37.62	0.61	0.86	1.42	1.07
GPT-4o	11.14	43.76	48.68	2.56	3.20	4.31	3.68
DeepSeek v2.5	6.67	49.60	50.88	0.65	0.63	1.98	0.95
GLM-4-Plus	13.53	38.52	43.46	4.39	1.18	7.00	2.02
ERNIE-4.0	4.06	40.10	43.07	0.72	1.60	3.64	2.22
Pure-SFT-7B	79.45	69.58	74.64	49.33	56.61	50.12	53.17
TIPA-7B	$84.72^{\uparrow 5.27}$	$70.70^{\uparrow 1.12}$	$75.90^{\uparrow 1.26}$	$51.63^{\uparrow 2.30}$	$58.72^{\uparrow 2.11}$	$51.54^{\uparrow 1.42}$	$54.90^{\uparrow 1.73}$
MTIPA-7B$_{(r=10\%)}$	$\mathbf{87.52}^{\uparrow 8.07}$	$\mathbf{72.40}^{\uparrow 2.82}$	$\mathbf{77.00}^{\uparrow 2.36}$	$\mathbf{54.67}^{\uparrow 5.34}$	$\mathbf{63.25}^{\uparrow 6.64}$	$\mathbf{54.95}^{\uparrow 4.83}$	$\mathbf{58.81}^{\uparrow 5.64}$

Table 4: (Experiment 1)Results on position-based CSC using CSCD-NS test dataset, showing that TIPA-7B and MTIPA-7B outperform the baseline in all evaluated metrics. GPT-4o's advantage stems from its character-level tokenization simplifying position identification, while other models face challenges with subword tokenization. Cross-tokenizer comparisons are less meaningful, yet previous LLMs performed poorly on this task.

5.5 Experiment 2: Traditional CSC Task

In the second experiment, we assessed the models on the traditional CSC task, which does not involve explicit position prediction. We also compared the impact of using forward ($\rightarrow$) and reverse ($\leftarrow$) TIPA constructions. This task is based on single-character to single-character mappings, and the evaluation methods are similarly structured. Due to the inherent difficulty large models like GPT-4o face in producing outputs with equal character lengths, we excluded any non-equal length data from the evaluation metrics. This exclusion ensures that length inconsistencies do not skew the experimental results. Detailed information on the models' output length consistency is available in the appendix. Some models prefer to convert half-width symbols to full-width ones, and we treat them as correct by establishing a mapping between half-width and full-width characters.

5.5.1 Results

Table 5 shows that TIPA improves model performance even when position prediction is not required. The reverse TIPA construction ($\leftarrow$) consistently outperforms the forward version ($\rightarrow$), suggesting that reverse ordering better enhances the model's understanding of internal character structures.

5.5.2 Evaluation on LEMON Dataset

To further assess the generalization ability of our models, we evaluated them on the LEMON dataset, which contains longer and more complex sentences. Table 6 shows that our TIPA-7B model achieves higher character-level F1 scores across various subsets of the LEMON dataset, indicating improved performance on difficult cases.

5.6 Experiment 3: General Model Evaluation

To validate TIPA's effectiveness beyond Chinese-specific tasks and LoRA fine-tuning, we conducted full-parameter supervised fine-tuning experiments on the Llama-3.1-8B model.

5.6.1 Results

Table 7 shows that TIPA-enhanced models maintain comparable performance on benchmarks while improving in character-sensitive tasks. The TIPA model achieves particularly strong gains in IFE-VAL (strict and loose) and AExams, demonstrating enhanced instruction following capabilities.

Table 8 demonstrates TIPA's effectiveness in multilingual settings, with the average F1 score improving from 47.85 to 52.81 across nine languages. The most significant gains appear in Finnish, Indonesian, and Korean, suggesting TIPA helps with non-Latin scripts.

Table 9 confirms TIPA's benefits for character-level tasks, with the TIPA model outperforming baseline by 3.95% on character occurrence counting, 7.11% on sentence length prediction, and achieving double the performance on distinct character counting (9.29% vs 18.74%).

6 Analysis

We conducted an in-depth analysis to understand the impact of TIPA and MTIPA on model performance.

6.1 Position Prediction Accuracy

Figure 3 compares position prediction accuracy across different character positions. The MTIPA-7B model consistently outperforms others, especially at higher character positions, indicating its enhanced ability to handle longer sequences.

Model	Sentence Level						Character Level					
	Detection			Correction			Detection			Correction		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Qwen2.5-7B	45.69	57.99	51.11	42.16	53.51	47.16	31.50	68.11	43.08	27.44	59.33	37.52
GPT-4o	35.88	65.19	46.28	33.36	60.62	43.03	34.10	86.64	48.94	31.14	79.10	44.68
Pure-SFT-1.5B	50.06	43.59	46.60	42.89	37.34	39.92	53.05	49.32	51.12	42.93	39.91	41.36
TIPA($\rightarrow$)-1.5B	**75.58**	62.11	68.18	**71.55**	**58.80**	**64.55**	**77.27**	64.38	**70.24**	**71.91**	59.91	**65.36**
TIPA($\leftarrow$)-1.5B	75.43	**62.21**	**68.19**	71.19	58.72	64.36	75.60	**64.58**	69.66	70.21	**59.98**	64.69
Pure-SFT-3B	78.39	66.96	72.22	74.87	63.96	68.99	80.59	69.25	74.49	75.54	64.91	69.82
TIPA($\rightarrow$)-3B	77.80	67.73	72.42	75.10	65.38	69.91	79.49	69.80	74.33	75.63	66.41	70.72
TIPA($\leftarrow$)-3B	**78.78**	**68.47**	**73.26**	**75.83**	**65.90**	**70.52**	**80.88**	**70.88**	**75.55**	**76.52**	**67.06**	**71.47**
Pure-SFT-7B	78.46	69.07	73.47	75.44	66.42	70.64	80.73	70.94	75.52	77.05	67.70	72.07
TIPA($\rightarrow$)-7B	78.04	69.56	73.55	75.32	67.14	70.99	79.98	72.63	76.13	76.76	69.70	73.06
TIPA($\leftarrow$)-7B	**81.79**	**70.95**	**75.98**	**78.84**	**68.40**	**73.25**	**83.33**	**73.34**	**78.01**	**79.64**	**70.09**	**74.56**

Table 5: (Experiment 2)Traditional CSC results on CSCD-NS test dataset. TIPA improves character-level detection and correction, with reverse-order TIPA showing greater gains at larger model scales.

Model	Character-Level F1 Score (%)								
	CAR	COT	ENC	GAM	MEC	NEW	NOV	CSCD-NS	AVG
Qwen2.5-7B	34.04	50.35	46.73	25.34	53.51	34.3	28.99	37.52	38.85
GPT-4o	41.87	44.11	47.98	31.62	51.48	47.11	37.52	44.68	43.3
Pure-SFT-1.5B	44.2	52.73	44.11	27.95	**51.21**	48.59	29.31	41.36	42.43
TIPA($\rightarrow$)-1.5B	**45.20**	**52.95**	**46.19**	28.4	50.03	47.41	**29.68**	**65.36**	**45.65**
TIPA($\leftarrow$)-1.5B	44.92	50.81	44.69	**28.49**	50.81	48.46	29.09	64.69	45.24
Pure-SFT-3B	49.18	59.34	46.93	26.13	55.15	**56.25**	32.68	69.82	49.44
TIPA($\rightarrow$)-3B	**49.72**	59.47	48.54	32.71	55.12	55.44	32.85	70.72	50.57
TIPA($\leftarrow$)-3B	49.22	**60.51**	**48.56**	**34.67**	56.02	55.8	**33.32**	**71.47**	**51.20**
Pure-SFT-7B	52.47	58.77	53.28	32.32	61.62	60.27	35.41	72.07	53.28
TIPA($\rightarrow$)-7B	**56.07**	**64.02**	52.91	35.56	62.56	60.24	38.96	73.06	55.42
TIPA($\leftarrow$)-7B	53.69	59.79	**55.65**	**36.46**	**63.15**	**61.16**	**39.65**	**74.56**	**55.51**

Table 6: (Experiment 2)Character-level F1 scores on LEMON and CSCD-NS for various domains. TIPA-7B consistently achieves higher F1, indicating better generalization.

Metric	TULU	TULU-TIPA
IFEVAL$_{0\text{-shot, strict}\uparrow}$	64.88	**67.84**
IFEVAL$_{0\text{-shot, loose}\uparrow}$	68.02	**70.24**
GSM8K$_{8\text{-shot}\uparrow}$	**74.53**	**74.53**
MMLU$_{5\text{-shot}\uparrow}$	**65.30**	**65.30**
AExams$_\uparrow$	38.92	**40.22**
KoBEST$_{F1\uparrow}$	51.09	**52.37**
HumanEval$_{pass@1\uparrow}$	**53.05**	51.83

Table 7: Standard benchmark evaluation of **Llama-3.1-Tulu** and **Llama-3.1-Tulu-TIPA-8B**, showing comparable performance with TIPA-enhanced model.

Language	F1	
	TULU	TULU-TIPA
Arabic	**67.22**	67.02
Bengali	30.44	**31.13**
English	61.16	**64.83**
Finnish	49.74	**60.99**
Indonesian	56.76	**66.64**
Korean	46.16	**56.37**
Russian	44.07	**44.95**
Swahili	47.73	**53.10**
Telugu	27.39	**30.28**
Average	47.85	**52.81**

Table 8: TyDi QA multilingual evaluation showing improved performance with TIPA.

6.2 Training Dynamics

Figure 4 shows the comparison of character-level metrics and position accuracy across different epochs for Pure-SFT-7B, TIPA-7B, and MTIPA-7B. MTIPA-7B achieves the highest performance ceiling.

6.3 Impact on Downstream Tasks

Our analysis confirms that TIPA and MTIPA significantly enhance models' understanding of internal character structures, leading to improved performance in both position prediction and traditional CSC tasks. The models no longer need to "guess"

Metric	TULU	TULU-TIPA
Occurrence	34.62	**38.57**
Length	24.48	**31.59**
Distinct	9.29	**18.74**

Table 9: Average character-level comparison between TULU and TIPA systems across eight languages (CN, EN, JP, KR, AR, EN-Hard, FR, RU).
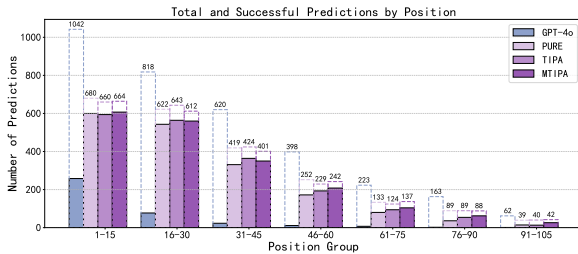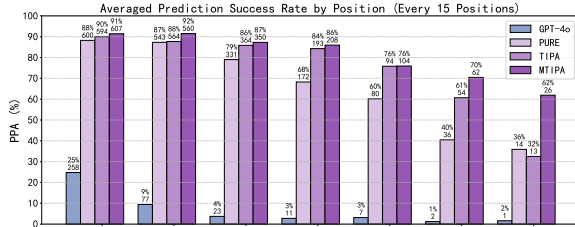

Figure 3: (Experiment 1)Comparison of position prediction accuracy by character position. MTIPA-7B achieves consistently higher accuracy, especially at longer positions.

the composition of tokens, as the training includes explicit character structure information.

Furthermore, by not altering the tokenizer or model architecture, our methods maintain compatibility with existing systems and do not introduce additional inference latency.

6.4 General Model Findings

Our general model experiments reveal several key insights:

- TIPA integration through full-parameter SFT maintains model performance on benchmarks while improving character-level capabilities.

- The multilingual evaluation shows TIPA's benefits extend beyond Chinese to other scripts and languages.

- Character-level tasks demonstrate significant improvements, particularly in counting distinct characters.

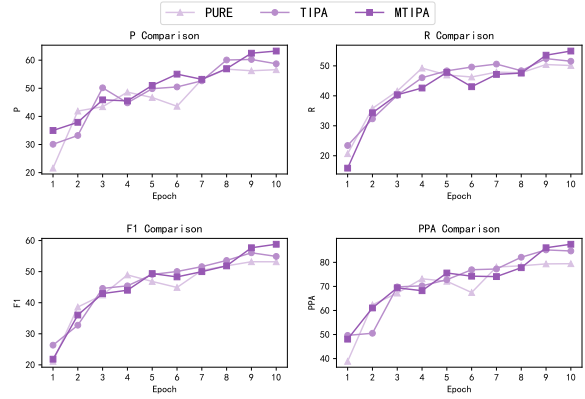- The approach scales effectively to larger models (8B parameters) without performance

Figure 4: (Experiment 1)Training dynamics showing character-level metrics and position accuracy. MTIPA-7B achieves the highest performance ceiling.

degradation.

These results suggest that TIPA's benefits are not limited to Chinese-specific tasks or LoRA fine-tuning, but represent a general improvement in LLMs' character-level understanding.

7 Conclusion

We introduced **Token Internal Position Awareness (TIPA)** and **Multi-Token Internal Position Awareness (MTIPA)** to enhance large language models' (LLMs) ability to accurately predict character positions within tokens. Our experiments demonstrate TIPA's effectiveness across three scenarios: position-aware Chinese spelling correction, traditional CSC tasks, and both training and evaluation of general models. The method shows particular strength in:

1. Improving position prediction accuracy in Chinese text (up to 8.07% absolute gain) 2. Enhancing traditional CSC performance (up to 5.64% F1 improvement) 3. Boosting multilingual and character-level capabilities without sacrificing benchmark performance 4. Scaling effectively to larger models through both LoRA and full-parameter fine-tuning

By training LLMs on reverse character prediction tasks using the tokenizer's vocabulary, TIPA, and MTIPA effectively address the limitations imposed by tokenization methods like Byte-Pair Encoding (BPE) while maintaining compatibility with existing systems.

8 Limitations

While our study presents promising results, there are several limitations:

3847

- The methods' ability to predict positions for out-of-vocabulary (OOV)(Sennrich, 2015; Wang et al., 2020) words, which do not directly appear in the tokenizer's vocabulary, requires further investigation.

- Directly mixing TIPA datasets during general model SFT training may induce a bias toward shorter text sequence generation when using excessive data proportions. While this can be mitigated by setting minimum token constraints, more fundamental solutions through pretraining or reinforcement learning stages warrant further exploration.

9 Acknowledgments

References

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2023. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288.*

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374.*

Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. *arXiv preprint arXiv:2402.08939.*

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in ty pologically di verse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168.*

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783.*

Momchil Hardalov, Todor Mihaylov, Dimitrina Zlatkova, Yoan Dinkov, Ivan Koychev, and Preslav Nakov. 2020. Exams: A multi-subject high school examinations dataset for cross-lingual and multilingual question answering. *arXiv preprint arXiv:2011.03080.*

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300.*

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. *arXiv preprint arXiv:2106.09685.*

Yong Hu, Fandong Meng, and Jie Zhou. 2024. Cscd-ns: a chinese spelling check dataset for native speakers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–159.

John D Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276.*

Itay Itzhak and Omer Levy. 2022. Models in a spelling bee: Language models implicitly learn the character composition of tokens. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5061–5068.

Myeongjun Jang, Dohyung Kim, Deuk Sin Kwon, and Eric Davis. 2022. Kobest: Korean balanced evaluation of significant tasks. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3697–3708.

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

Ayush Kaushal and Kyle Mahowald. 2022. What do tokens know about their characters and how do they know it? *arXiv preprint arXiv:2206.02608*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. 2024. Tülu 3: Pushing frontiers in open language model post-training.

Jiahao Li, Quan Wang, Zhendong Mao, Junbo Guo, Yanyan Yang, and Yongdong Zhang. 2022. Improving chinese spelling check by character pronunciation prediction: The effects of adaptivity and granularity. *arXiv preprint arXiv:2210.10996*.

Kunting Li, Yong Hu, Liang He, Fandong Meng, and Jie Zhou. 2024. C-llm: Learn to check chinese spelling errors character by character. *arXiv preprint arXiv:2406.16536*.

Zihong Liang, Xiaojun Quan, and Qifan Wang. 2023. Disentangled phonetic representation for chinese spelling correction. *arXiv preprint arXiv:2305.14783*.

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

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE.

Rico Sennrich. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Andrew Shin and Kunitake Kaneko. 2024. Large language models lack understanding of character composition of words. *arXiv preprint arXiv:2405.11357*.

Changxuan Sun, Linlin She, and Xuesong Lu. 2024. Two issues with chinese spelling correction and a refinement solution. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 196–204.

Avijit Thawani, Saurabh Ghanekar, Xiaoyuan Zhu, and Jay Pujara. 2023. Learn your tokens: Word-pooled tokenization for language modeling. *arXiv preprint arXiv:2310.11628*.

Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. 2015. Introduction to sighan 2015 bake-off for chinese spelling check. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 32–37.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Changhan Wang, Kyunghyun Cho, and Jiatao Gu. 2020. Neural machine translation with byte-level subwords. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9154–9160.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2517–2527.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Hongqiu Wu, Shaohua Zhang, Yuchen Zhang, and Hai Zhao. 2023. Rethinking masked language modeling for chinese spelling correction. *arXiv preprint arXiv:2305.17721*.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at sighan bake-off 2013. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 35–42.

Nan Xu and Xuezhe Ma. 2024. Llm the genius paradox: A linguistic and math expert's struggle with simple word-based counting problems. *arXiv preprint arXiv:2410.14166*.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

François Yergeau. 2003. Utf-8, a transformation format of iso 10646. Technical report.

Xunjian Yin and Xiaojun Wan. 2023. A comprehensive evaluation and analysis study for chinese spelling check. *arXiv preprint arXiv:2307.13655*.

Junjie Yu and Zhenghua Li. 2014. Chinese spelling error detection and correction based on language model, pronunciation, and shape. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 220–223.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient finetuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Appendix

A.1 The limitations of GPT-4o

In Table 10, we present examples highlighting the limitations of GPT-4o in handling character-level tasks due to its lack of internal character structure awareness.

These examples illustrate how GPT-4o struggles with tasks that require precise character-level understanding. The tokenization process obscures internal character structures, leading to errors in character counting, splitting, and position identification. Our proposed methods, TIPA and MTIPA, aim to address these issues by enhancing models' awareness of internal token structures.

A.2 Implementation Details

In our experiments, we utilized PyTorch and the Hugging Face Transformers (Wolf et al., 2020) library to implement our models. We fine-tuned the open-source models using LoRA (Hu et al., 2021) (Low-Rank Adaptation of Large Language Models) with specific configurations to efficiently adapt the large models without modifying the original weights.

A.2.1 TIPA Training Methodology

We implemented two approaches for TIPA training:

- **Unpruned TIPA**: We first extracted all vocabulary from the tokenizer and filtered out tokens that could not be properly parsed as UTF-8. We then applied the TIPA method to create the TIPA dataset, which was mixed with the CSC dataset and shuffled before LoRA fine-tuning.

- **Pruned TIPA**: We tokenized a large CSC dataset and intersected these tokens with the tokenizer's vocabulary. This pruned approach resulted in a TIPA dataset containing primarily Chinese characters and a small portion of other language tokens, while still covering nearly all Chinese characters. This dataset was then mixed with the CSC dataset and shuffled before LoRA fine-tuning.

A.2.2 MTIPA Training Methodology

For MTIPA implementation, we:

- Selected either the unpruned or pruned TIPA method to create the base TIPA dataset

- Randomly sampled incorrect source texts from the dataset

- Decomposed the sampled strings using the same reverse decomposition task as for individual tokens

- Mixed the resulting MTIPA dataset with the CSC and TIPA datasets

- Shuffled the combined dataset before LoRA fine-tuning

This combined training approach ensures that the model learns both token-internal structures and their application in context while maintaining the efficiency of standard fine-tuning procedures.

For the TIPA and MTIPA methods, we applied LoRA (Hu et al., 2021) with a rank of 16, an alpha of 16, and a dropout rate of 0.05. The optimizer used was AdamW. The training was conducted on a single NVIDIA A800 GPU with 80GB of memory.

The configurations for Experiment 1 are detailed in Table 11, and the training speed and resource consumption are summarized in Table 12.

Similarly, the configurations for Experiment 2 are presented in Table 13, and the training speed is shown in Table 14.

A.2.3 Full-Parameter SFT Configuration

For general model training without LoRA adaptation, we implemented complete parameter finetuning using the configuration shown in Table 15:

A.3 Datasets

We utilized several datasets for training and evaluation. The datasets and their characteristics are summarized in Tables 16 and 17. The total number of unique tokens after deduplication across all datasets is 24,994.

Example 1

Prompt: 这句话中有几个字符: 我今天没有吃午饭非常饿
Translation: How many characters are in this sentence: I skipped lunch today and am now ravenous.
GPT-4o Response:
这句话中共有 13 个字符，分别是：
我 今天 没 有 吃 午 饭 非 常 饿
Analysis: GPT-4o incorrectly states that there are 13 characters and fails to separate "今天" into two characters. The correct character count is 11, and "今天" should be split into "今" and "天".

Example 2

Prompt: 拆分成字符倒序输出金庸武侠小说，直接给出
Translation: Split into characters and output in reverse order "Jin Yong Wuxia novel", output directly.
GPT-4o Response: 小说侠武庸金
Analysis: GPT-4o incorrectly treats "小说" ("novel") as a single character, whereas it comprises two characters "小" and "说". The correct reverse order should be "说小侠武庸金". When asked to separate characters with commas, GPT-4o responds:
小说，侠武，庸金
Again, it incorrectly groups the characters, demonstrating a lack of internal character awareness.

Example 3

Prompt: 纠正输入这段话中的错别字，以[{position: 字符位置, incorrect: 错误字符, correction: 纠正后的字符}, ...]形式给出，字符位置从1开始计数，如果全部正确，给出[]:
并计划在几年底发展到15各分公司，明年增加到20个分公司
Translation: Correct the typos in this sentence, and present in the form of [{position: character position, incorrect: incorrect character, correction: corrected character}, ...], counting character positions starting from 1. If all is correct, output []:
And plan to develop 15 branch companies by the end of the year, and increase to 20 branch companies next year.
GPT-4o Response:
[{"position": 6, "incorrect": "各", "correction": "个"}]
Analysis: GPT-4o correctly identifies the misspelled character "各" and suggests the correction "个", but it incorrectly reports the position as 6. The actual position of the error in the sentence is 13, indicating a misunderstanding of character positions within the tokenized text.

Table 10: Examples demonstrating GPT-4o's limitations in character-level tasks due to lack of internal character structure awareness.

Configs	Values
Devices	1 NVIDIA A800 GPU (80GB)
Batch size	16
Learning rate	1×10^{-4}
Epochs	10
LoRA rank	16
LoRA alpha	16
LoRA dropout	0.05
Optimizer	AdamW

Table 11: Configurations for Experiment 1.

Method	Batches	Speed	GPU hours
Pure-SFT-7B	188,340	$\sim$ 1 s/batch	52.6 h
TIPA-7B	203,960	$\sim$ 1 s/batch	56.7 h
MTIPA-7B	222,780	$\sim$ 1.6 s/batch	99.0 h

Table 12: Training speed and resource consumption for Experiment 1.

The distribution of token lengths in the pruned token set used to construct the TIPA dataset is shown in Figure 5. We observed that most tokens have lengths of 1 or 2 characters.

A.4 Prompts Used in Experiments

We provide the prompts used in our experiments for different models and tasks. Due to the length of the prompts, we format them to allow for automatic line wrapping.

A.5 Analysis of Output Length Consistency

We analyzed the proportion of outputs where the corrected text has the same character length as the original text. This metric reflects the models' ability to maintain length consistency, which is important for certain applications.

In Table 19, Qwen2.5-7B and GPT-4o were explicitly instructed to output texts of the same length as much as possible, but their averages are relatively low, indicating that these models do not fully grasp the length information of the tokens. Our models were not instructed to output texts of the same length, as all training samples were of equal

3851

Configs	Values
Devices	1 NVIDIA A800 GPU (80GB)
Batch size	16
Learning rate	1×10^{-4}
Epochs	6 (results reported at epoch 3)
LoRA rank	16
LoRA alpha	16
LoRA dropout	0.05
Optimizer	AdamW

Table 13: Configurations for Experiment 2. The models achieved optimal performance at epoch 3, beyond which overfitting was observed.

Method	Batches	Speed	GPU hours
Pure-SFT-7B	56,502	~ 0.96 s/batch	~ 15 h
TIPA($\rightarrow$)-7B	61,188	~ 0.95 s/batch	~ 16 h
TIPA($\leftarrow$)-7B	61,188	~ 0.95 s/batch	~ 16 h
Pure-SFT-3B	56,502	~ 0.58 s/batch	~ 9 h
TIPA($\rightarrow$)-3B	61,188	~ 0.58 s/batch	~ 10 h
TIPA($\leftarrow$)-3B	61,188	~ 0.58 s/batch	~ 10 h
Pure-SFT-1.5B	56,502	~ 0.35 s/batch	~ 5.5 h
TIPA($\rightarrow$)-1.5B	61,188	~ 0.35 s/batch	~ 5.9 h
TIPA($\leftarrow$)-1.5B	61,188	~ 0.35 s/batch	~ 5.9 h

Table 14: Training speed and resource consumption for Experiment 2.

length. This demonstrates the models' ability to mimic and generalize length information through training methods. In CSCD-NS, the error patterns of tokens are present in the training set, so the Pure-SFT-3B model has good mimicking ability. In the LEMON dataset, since more new tokens may appear, TIPA has trained character composition awareness for all possible tokens, while the Pure-SFT method cannot perceive such tokens during training, so TIPA shows better generalization ability. However, this does not involve including the test set in the training set, because the tokens are known in the vocabulary, and we can train the tokenizer's vocabulary without obtaining any datasets.

A.6 Additional Results

We provide additional examples from our experiments to illustrate the impact of TIPA and MTIPA on model outputs.

A.6.1 Experiment 1: CSC with Position Prediction

Table 20 shows several examples comparing the outputs of the Pure-SFT-7B model and the MTIPA-7B model on the CSC task with position prediction.

Configs	Values
Devices	4 NVIDIA H100 GPUs (80GB)
Batch size	32 (gradient accumulation)
Learning rate	5×10^{-6}
Epochs	2

Table 15: Full-parameter SFT configurations. All unspecified parameters align with TULU training settings. Training completed in 3d 8h 10m 26s.

Dataset	Samples	Unique Tokens
Training		
CSCD-NS Train	30,000	22,012
Wang271K	271,329	20,369
Validation		
CSCD-NS Dev	5,000	16,003

Table 16: Training and validation datasets were used in our experiments. "Unique Tokens" refers to the number of unique tokens appearing in the tokenizer after deduplication.

A.6.2 Experiment 2: Traditional CSC Task

Table 21 provides examples from Experiment 2, comparing the outputs of the Pure-SFT-7B model and the TIPA-7B model on the traditional CSC task.

A.7 Loss Analysis

Figure 6 compares training and validation loss across different methods. The TIPA-7B model exhibits the fastest loss reduction during training, indicating more efficient learning.

Dataset	Filtered Samples	Unique Tokens
CSCD-NS Test	5,000	15,946
NOV	6,000	10,503
CAR	3,245	10,881
COT	993	3,564
ENC	3,274	13,399
GAM	393	3,055
MEC	1,942	4,813
NEW	5,887	12,010

Table 17: Testing datasets used in our experiments after filtering. "Filtered Samples" indicates the number of samples remaining after filtering out those with unequal source and target sentence lengths. "Unique Tokens" refers to the number of unique tokens appearing in the tokenizer after deduplication.
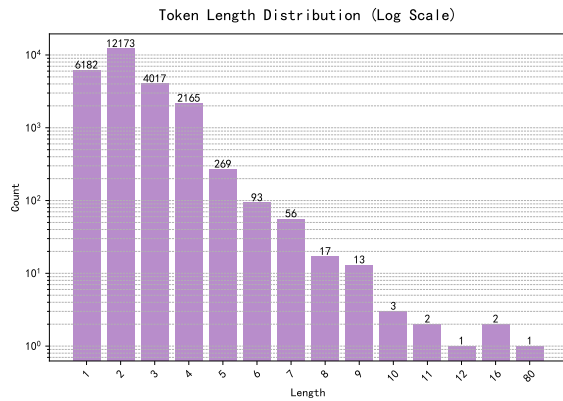

Figure 5: Token length distribution in the pruned token set used for constructing the TIPA dataset. The y-axis is on a logarithmic scale(Hunter, 2007). The token with a length of 80 is an exceptional case and can be excluded in practical applications to avoid excessively long tokens in TIPA.
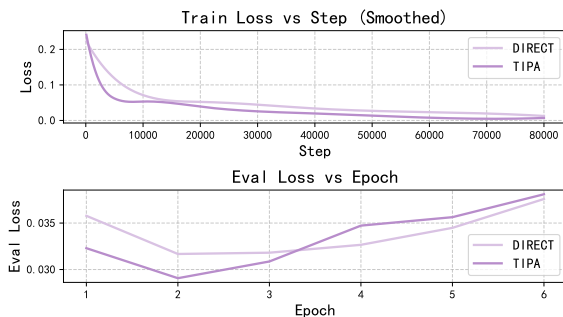

Figure 6: (Experiment 2)Training and validation loss comparison across different methods in the traditional CSC task. The TIPA-7B model exhibits the fastest reduction in loss during training, indicating more efficient learning and better generalization capabilities compared to the baseline.

Experiment	Prompt (Chinese / English) (temperature = 0.01)
Experiment 1 (All models)	**Chinese:** 纠正输入这段话中的错别字，以 [{position: 字符位置, incorrect: 错误字符, correction: 纠正后的字符}, ...] 形式给出，字符位置从 1 开始计数，必须是单个字符，如果全部正确，给出 []。 **English:** Correct the typos in this paragraph of input and provide it in the form [position: character position, incorrect: wrong character, correction: corrected character, ...], where character position starts counting from 1, must be a single character, if all is correct, give [].
Experiment 2 (Testing GPT-4o, Qwen2.5-7B original models) (temperature = 0.01)	**Chinese:** 纠正输入这段话中的错别字，直接给出纠正后的文本，无需任何解释，不要补充任何标点符号，尽可能输出等长的新句子！ **English:** Correct the typos in this paragraph of input and directly give the corrected text, no need for any explanation, do not add any punctuation marks, and try to output a new sentence of equal length!
Experiment 2 (Trained models)	**Chinese:** 纠正输入这段话中的错别字，直接给出纠正后的文本，无需任何解释。 **English:** Correct the typos in this paragraph of input, and directly give the corrected text, no need for any explanation.

Table 18: Prompt templates used in our experiments, presented in both Chinese and English. The prompts were designed to elicit the desired output formats from the models.

Model	CAR	COT	ENC	GAM	MEC	NEW	NOV	CSCD-NS	AVG
Qwen2.5-7B	59.69	65.16	63.44	55.98	65.71	58.72	58.33	66.02	61.63
GPT-4o	77.69	83.18	83.87	82.44	86.56	79.80	79.63	87.74	82.61
Pure-SFT-1.5B	**95.13**	**96.88**	**96.06**	96.18	**95.83**	97.27	96.12	96.28	96.22
TIPA(→)-1.5B	95.04	95.47	95.97	**96.95**	95.67	97.06	**96.33**	**98.68**	**96.40**
TIPA(←)-1.5B	94.67	95.57	95.97	95.42	95.62	96.96	96.0	98.52	96.09
Pure-SFT-3B	94.82	95.37	95.66	95.67	96.55	97.50	94.97	**98.92**	96.18
TIPA(→)-3B	95.01	95.37	**96.46**	95.93	96.14	**97.55**	94.43	98.66	96.19
TIPA(←)-3B	**95.50**	**96.07**	95.94	**97.20**	**96.86**	97.40	**95.47**	98.66	**96.64**
Pure-SFT-7B	93.93	95.27	95.51	96.44	93.72	97.03	94.05	98.60	95.57
TIPA(→)-7B	**95.75**	**96.27**	**96.30**	**97.71**	**96.04**	**98.05**	**96.07**	**99.16**	**96.92**
TIPA(←)-7B	94.58	94.76	95.51	97.20	95.57	97.55	95.42	98.84	96.18

Table 19: Proportion (%) of outputs where the corrected text has the same character length as the original text across different domains and models. The average (AVG) is also provided.

Source Text:

...产品的移动芯片及相关平台开发，期 芯片类产品支持...

Pure-SFT-7B Output:

[{ "position": 53 , "incorrect": "期", "correction": "其" }]

MTIPA-7B Output:

[{ "position": 52 , "incorrect": "期", "correction": "其" }]

Translation: "...development of mobile chips and related platforms for products, its chip products support..."

Comment: The MTIPA-7B model correctly identifies the position of the incorrect character "期" (should be "其") at position 52, whereas the Pure-SFT-7B model predicts position 53.

Source Text:

...加盟中超大连实德后 再 球队效力三年，上赛季结束后离队，最终选择退役

Pure-SFT-7B Output:

[{ "position": 38 , "incorrect": "再", "correction": "在" }]

MTIPA-7B Output:

[{ "position": 37 , "incorrect": "再", "correction": "在" }]

Translation: "...after joining Dalian Shide in the Chinese Super League, he served the team for three years, left after last season, and finally chose to retire."

Comment: The MTIPA-7B model accurately predicts the position of the incorrect character "再" (should be "在") at position 37, while the Pure-SFT-7B model predicts position 38.

Table 20: Examples from Experiment 1 comparing Pure-SFT-7B and MTIPA-7B outputs on the CSC task with position prediction. The MTIPA-7B model demonstrates enhanced position prediction accuracy.

Source Tokens:

.../ 以 / 吃的 / 名 意 / 来 / 一场 / 说 / 走 / 就 / 走 / 的 / 美食 / 之旅 /

Pure-SFT-7B Output:

.../ 以 / 吃的 / 心情 / 来 / 一场 / 说 / 走 / 就 / 走 / 的 / 美食 / 之旅 /

TIPA-7B Output:

.../ 以 / 吃的 / 名义 / 来 / 一场 / 说 / 走 / 就 / 走 / 的 / 美食 / 之旅 /

Translation: "Take a spontaneous food journey in the name of eating."

Comment: The TIPA-7B model correctly reconstructs "名义" ("name"), improving the fluency and accuracy of the sentence compared to the Pure-SFT-7B model's output.

Source Tokens:

/ 伦敦 / 又 / 粗 / 线 / 了一 / 家 / 哈利 / · / 波特 / 主题 / 的 / 酒店 / 套房 /

Pure-SFT-7B Output:

/ 伦敦 / 又 / 粗 / 线 / 了一 / 家 / 哈利 / · / 波特 / 主题 / 的 / 酒店 / 套房 /

TIPA-7B Output:

/ 伦敦 / 又 / 出现 / 了一 / 家 / 哈利 / · / 波特 / 主题 / 的 / 酒店 / 套房 /

Translation: "London has another Harry Potter themed hotel suite appeared ."

Comment: The TIPA-7B model accurately corrects "粗线" (nonsensical in this context) to "出现" ("appear"), enhancing the sentence's coherence.

Source Tokens:

/ 本事 / 一家人 / ...

Pure-SFT-7B Output:

/ 本 / 是 / 亲 / 一家 / ...

TIPA-7B Output:

/ 本 / 是一 / 家人 / ...

Translation: "We were originally a family."

Comment: The TIPA-7B model correctly reconstructs the phrase "是一家人" ("are a family"), improving the grammaticality and meaning of the sentence.

Table 21: Examples from Experiment 2 comparing Pure-SFT-7B and TIPA-7B outputs on the traditional CSC task. The TIPA-7B model demonstrates an enhanced understanding of token internal structures, leading to better corrections.