Generative Input: Towards Next-Generation Input Methods Paradigm

Keyu Ding[†‡], **Yongcan Wang**[*‡§], **Zihang Xu**[*‡§], **Zhenzhen Jia**[‡§], **Enhong Chen**[¶]
[†]University of Science and Technology of China
[‡]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[§]iFLYTEK AI Research
[†]{kyding}@mail.ustc.edu.cn
[¶]{cheneh}@ustc.edu.cn
[‡]{kyding,ycwang12,zhxu13,zzjia3}@iflytek.com

Abstract

Since the release of ChatGPT, generative models have achieved tremendous success and become the de facto approach for various NLP tasks. However, its application in the field of input methods remains under-explored. Many neural network approaches have been applied to the construction of Chinese input method engines (IMEs). Previous research often assumed that the input pinyin was correct and focused on Pinyin-to-character (P2C) task, which significantly falls short of meeting users' demands. Moreover, previous research could not leverage user feedback to optimize the model and provide personalized results. In this study, we propose a novel **Gene**rative **Input** paradigm named **GeneInput**. It uses prompts to handle all input scenarios and other intelligent auxiliary input functions, optimizing the model with user feedback. The results demonstrate that we have achieved state-of-the-art performance for the first time in the Full-mode Key-sequence to Characters task. GeneInput also includes RLHF-IME, a novel RLHF application framework for input method, that eliminates the need for manual ranking annotations and the performance surpasses GPT-4. Relevant resources have been open-sourced[1].

1 Introduction

One of the primary objectives of IMEs is to assist users in efficient text input. In some Asian languages, such as Chinese, Japanese they do not use alphabetic characters and cannot be directly inputted through a standard keyboard. Users often need to employ commercial input software, such as Google Input Method[2], Sogou Input Method[3], iFlytek Input Method[4], and so on, to accomplish text input.

Figure 1: PinYin input scenarios in typical input modes.

Pinyin serves as the official romanization system for the Chinese language. In China, there are two common keyboard input methods: the 9-key keyboard and the 26-key keyboard. Each of these keyboard inputs further includes different input modes. In practice, user input scenarios are highly complex, with typical input modes illustrated in Figure 1, which illustrates various potential input modes for the Chinese sentence "我爱自然语言处理" (I love natural language processing). Some of the possible input modes include:

1) 26-keyboard perfect Pinyin sequence(e.g., "wo ai zi ran yu yan chu li"). 2) 26-keyboard abbreviated sequence(e.g., "w a z r y y c l"). Both of these input modes have been extensively studied in prior works (Chen et al., 2015; Tan et al., 2022; Xiao et al., 2022). 3) 9-keyboard perfect Pinyin sequence (e.g., '96'24'94'726'98'926'248'54'). 4) 26-keyboard random abbreviated pinyin sequence (e.g.,"wo ai zi ran y y c l").5) 26-keyboard pinyin with noise sequence (e.g., "wo ai zhi rna yu uan chu li").

With the rapid development of AI technology, the functionality of input methods has far exceeded P2C task. New features have emerged, such as

*Equal contributions by alphabetical order.

[1]https://github.com/spirit-wang/Generative-Input/tree/master

[2]https://www.google.com/inputtools/services/features/input-method.html

[3]https://pinyin.sogou.com/

[4]https://srf.xunfei.cn/

Findings of the Association for Computational Linguistics: ACL 2024, pages 3658–3669

intelligent association, conversational assistance, text correction, aimed at enhancing input efficiency, enjoyment and accuracy.

To the best of our knowledge, there is currently no related research work covering such a wide range of practical input modes and AI-assisted input scenarios. Traditional input methods typically treat P2C as a sequence labeling task. Pre-trained models such as BERT-CRF (Souza et al., 2019) and GPT (Tan et al., 2022) have been applied to P2C task. Cai et al. (2018) propose KNPTC to integrate letter-neighbor knowledge into NMT for Pinyin Error Correction.

This paper, for the first time, explores the feasibility of using a unified generative framework to model all typical tasks in IMEs. It unifies the tasks as text generation tasks, completely overturning the previous paradigm of constructing IMEs. Addressing the uncertainty in the task of mapping key sequences to character sequences in real input scenarios, a two-stage decoding strategy is proposed. To tackle the significant issue of noise in input, a soft constraint strategy is introduced, enabling users' key sequences to yield optimal text results. Unlike traditional input method models with fixed resources, this paper introduces a user feedback-based reinforcement learning approach in the context of IMEs, allowing the online updating of the model and greatly enhancing its self-evolving capabilities. The primary contributions of this paper are three-fold: 1) A novel generative input method framework is proposed, achieving state-of-the-art performance. 2) An optimization architecture of reinforcement learning based on user feedback is introduced in the context of IMEs, enabling online automatic optimization of LLM. 3) Addressing complex real-world user input scenarios, a two-stage decoding strategy and a soft constraint strategy are proposed.

2 Tasks

We selected the three most representative tasks in the input method scenarios for our research.

Full-mode Key-sequence to Characters (FK2C) This task is the core function of the input method. Previous studies commonly assumed that the input sequence is noise-free and follows a specific mode, focusing solely on the conversion from pinyin to characters, denoted as P2C. However, the challenges in actual input scenarios are much more difficult, as illustrated in Fig1, and

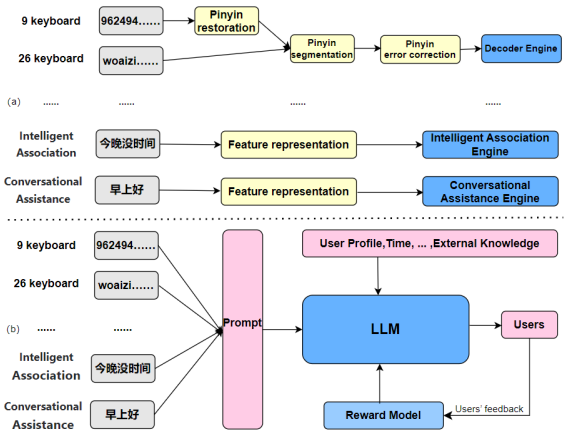

Figure 2: Comparison between the traditional(a) and GeneInput paradigm(b).

the same key sequence may correspond to different input mode results. Therefore, directly modeling the conversion of key sequences to characters is a more challenging task than traditional P2C tasks.

Intelligent Association (IntelAssoc) This is a typical text continuation task, which predicts the user's expected next sentence based on the previous content to improve input efficiency.

Conversational Assistance (ConvAssist) This is a text beautification task, rewriting the input to meet specific requirements, such as being more humorous, without changing the user's semantics.

3 Methodology

In this paper, we propose a generative modeling scheme GeneInput to uniformly model the typical tasks and different input modes contained in the input method scenario, as Figure 2(b) shows. It leverages an LLM to model Pinyin decoding tasks in various noisy scenarios, various AI-assisted input functionalities, and utilizes user feedback to automatically adjust and optimize the model. Additionally, it integrates historical user input information to provide personalized results.

3.1 GeneInput

Large language models (LLM) have achieved good results in many tasks, and studies show that LLM can distinguish tasks through different prompts, thus unifying the modeling of different tasks (Ouyang et al., 2022; OpenAI, 2023; Wei et al., 2021). However, to the best of our knowledge, there is no related work that uses LLM to uniformly model input method related tasks, and the existing LLM does not perform well on the

core tasks of input method. Therefore, this work attempts to use LLM for unified modeling of input method typical tasks.

We designed corresponding prompts for the three typical input method tasks introduced in section2, and fine-tuned them based on generative LLM. As shown in Figure 3(a), in the model, given the corresponding task description P and input X, we predict the corresponding output character $Y = [y_1, ..., y_n]$. The model training objective is to minimize the following loss function.

$$L = -\sum_{j=1}^{n} \log p\left(y_j | y_{<j}, P, X\right) \tag{1}$$

Specifically, for IntelAssoc, the input X is the current input sentence and the output Y is the corresponding possible next sentence. For ConvAssist, the input X is the user's original input sentence and the output Y is the beautified paraphrased sentence of this sentence. And for the FK2C task, the input X is the 26-key or 9-key keystroke sequence and the output Y is the corresponding Chinese character result. And for each task, we carefully design a corresponding task description P to ensure that the model knows the requirements of each task, thus making clear distinctions among tasks. Since FK2C is a more complex and challenging task, we have done a more extended design for this task, which will be introduced in section 3.2. We provide examples of each task in Appendix A.

The structure of GeneInput is simple and flexible, the task description, input and output can be modified or extended according to different tasks. With the user's informed consent and authorization, GenInput can conveniently incorporate existing historical input and user profile information to provide users with more accurate personalized results. We conducted a case study on a small group of internal users who participated in a performance improvement project, the details of which can be obtained in Appendix B.1.

3.2 Full-mode Key-sequence to Characters

Previous P2C studies assume that the input is segmented pinyin, such as "Wo'Ban", which already has a deterministic input mode. We follow the actual input process to model keystroke sequences to characters end-to-end. The same key-sequence may have multiple modes of results. As shown in Figure 3, when the key sequence "woban" is entered, it can be segmented into perfect pinyin

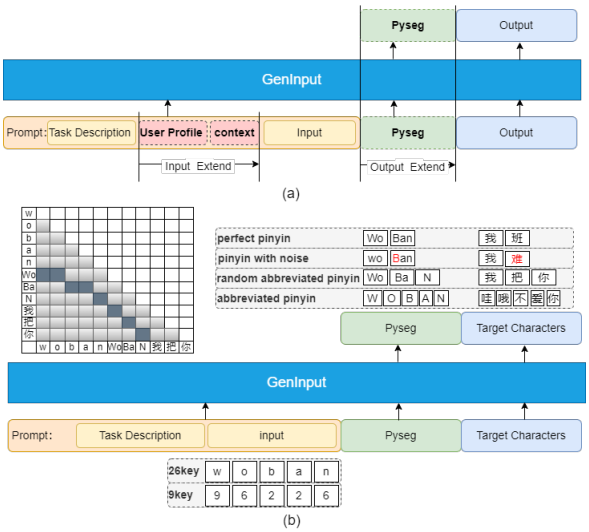

Figure 3: The architecture of IME unified modeling (a) and full-mode K2C (b).

"Wo'Ban", random abbreviate pinyin "WoBaN", or other modes. Hence, pinyin segmentation plays a significant role in FK2C. We refer to the pinyin segmentation results as "pyseg" for simplicity.

3.2.1 FK2C Modeling Based on Pyseg

Research shows that it is insufficient to directly model the mapping from input x to output y for complex problems, and the introduction of intermediate processes can greatly enhance the ability of LLM (Wei et al., 2022). As shown in Figure 3(b), we add pyseg as an intermediate result to enhance the modeling ability of different input modes, so as to realize the unified modeling of multiple input modes of the input method. First, from the task description P and the input keystroke sequence $X = [x_1, ..., x_m]$, we predict the possible pyseg $S = [s_1, ..., s_n]$, and then combine the first two to predict the final characters $Y = [y_1, ..., y_n]$. After the output is extended, the corresponding training loss function is:

$$L_{pysegs} = -\sum_{i=1}^{n} \log p\left(s_i | s_{<i}, P, X\right) \tag{2}$$

$$L_{characters} = -\sum_{j=1}^{n} \log p\left(y_j | y_{<j}, S, P, X\right) \tag{3}$$

$$L = \lambda \cdot L_{pysegs} + L_{characters} \tag{4}$$

Here, λ is an adjustable hyper-parameter.

3.2.2 Soft Alignment Constraint

Unlike hard constraint methods, such as constrained decoding (Tan et al., 2022), which can only be applied to extremely desirable input scenarios. First, when the input is an actual sequence of keystrokes, there is no clear alignment between the input and the output. Secondly, when the input is noisy, directly restricting the character decoding space to fully conform to pinyin may not yield results. To be able to handle noisy inputs while avoiding producing unacceptable results, we design a soft alignment constraint. See the appendix D for specific differences. The alignment constraint can be divided into two stages: pyseg-input alignment and output-pyseg alignment.

Pyseg-Input Alignment Since Pinyin segmentation only performs a limited spatial mapping and segmentation on the input, we can compare it with the original input by mapping the decoded pyseg back to the corresponding key-sequence, such as "Wo'Ban" to "woban" of 26-key or "96222" of 9-key, and remove the pysegs that are inconsistent with the input. s_i represents the generated i-th pinyin, and its probability is as follows.

$$p(s_i) = \frac{exp(g(s_i))}{\sum_{P2I(s_{<j}s_j)\in\nu_X} exp(g(s_j))} \tag{5}$$

In this context, g represents the logit before softmax, ν_X denotes the prefix subset corresponding to the input X, and $P2I()$ represents the mapping function from pinyin to input. That is, the current decoded pyseg should be restored as a prefix subset of the input X, and the final pyseg restored should be consistent with the input X.

Output-Pyseg Alignment The output string $Y = [y_1, ..., y_n]$ and pyseg $S = [s_1, ..., s_n]$ correspond one-to-one. By comparing the edit distance between the pinyin of Chinese characters and the corresponding position of pyseg, we impose a certain penalty on the noisy results, as shown in Figure 3(b), the pinyin "Nan" of the character "难(hard)" and the "Ban" in pyseg, so as to realize the processing of noisy input while avoiding excessive error correction. The correction penalty coefficient for the i-th step is as follows.

$$\varepsilon_i = \frac{\alpha}{n} EditDist(s_i, C2P(y_i, mode(s_i))) \tag{6}$$

In this context, n is the number of inputs corresponding to s_i, EditDistance(a, b) represents the edit distance between a and b. The function $C2P()$ is used to romanize the generated Chinese characters y_i and select the corresponding perfect pinyin or abbreviate pinyin based on the mode of s_i. α adjusts the correction penalty strength.

After increasing the alignment constraints, y_i represents the generated i-th Chinese character, and its corresponding probability is as follows.

$$p(y_i) = (1 - \varepsilon_i) \cdot \frac{exp(g(y_i))}{\sum exp(g(y_j)} \tag{7}$$

3.3 Online Optimization with Human Feedback

Previously, the optimization of language models behind input method editors mainly followed the classical paradigm of "pre-training + fine-tuning"(Radford and Narasimhan, 2018). However, such a complete model development process has very high requirements on the quality and quantity of training data, computational resources, time, and so on, so it is difficult to iterate models rapidly. In the input method scenario, the style and preference of people's daily communication language change fast with the passage of time, so the traditional paradigm can not meet the optimization needs of the input method models. Recently, the key technology behind ChatGPT, RLHF, can effectively help the model to follow the human preference(Ouyang et al., 2022). Therefore, we explored applying RLHF on input method editor models in order to make its outputs on the downstream tasks more in line with the requirements of the real application users. We refer to this part as RLHF-IME.

Studies show that the quality of the reward model (RM) determines the upper bound of RLHF to a large extent (Zheng et al., 2023). However, Ouyang et al. (2022) requires many high-quality trained annotators to give ranking annotations, which is too costly in terms of time and budget, we design fully automated annotation methods that are more feasible and friendly to a large number of real-world application scenarios in the industry. Considering the text characteristics of the input method scenario, multiple RM training methods based on two annotation systems are designed.

Figure 4 illustrates the whole process of RLHF-IME which consists of automatic data construction, reward modeling, and LLM optimized with the reinforcement learning iteratively.

3.3.1 Ranking System

Ouyang et al. (2022) shows us that it is effective to train RMs based on manually labeled rankings

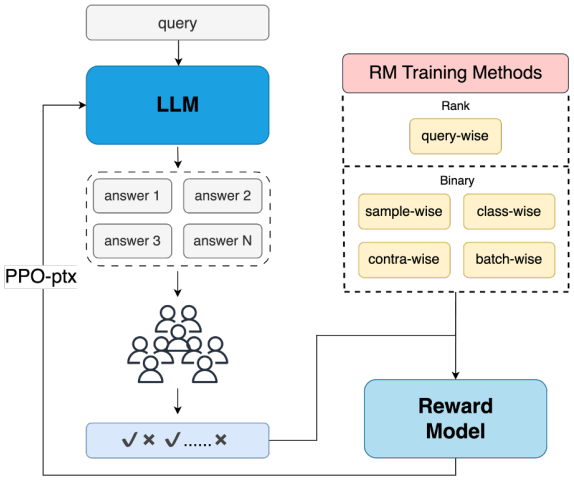

Figure 4: Workflow of RLHF-IME.

of LLM outputs. For labeling automatically, we take three months as the statistical cycle, the label score is calculated based on the percentage of the number of times the answer is selected to the number of times the answer is provided as a candidate answer as shown in Equation 8, where $N^i_{selected}$ is the number of i_{th} sample selected by users.

$$label_i = \begin{cases} 0 & if N^i_{selected} = 0, \\ max(1, \frac{N^i_{selected}}{N^i_{provided}} * 100) & otherwise. \end{cases} \tag{8}$$

Under the ranking annotation system, we design the following RM training method.

Query-Wise The Query-Wise method is designed based on comparing samples with the same query but different answers. Under the same query, it is reasonable to rank different answers according to the user labeling scores and train models to judge the quality of answers, otherwise, comparing the scores is meaningless. The loss function is constructed by the formula 9, 10 and 11, where n denotes the number of answers of the current query, $query_i$ means the query of the i_{th} sample and $label_i$ means the human preference score of the i_{th} answer. $s(x, y)$ stands for the RM's scoring of answer y when the query is x.

$$ld(i, j) = label_i - label_j \tag{9}$$

$$bt(i) = \sum_{j=1}^{n} \mathbb{I}_{query_i = query_j} \mathbb{I}_{ld(i,j)>0} \cdot 1 \tag{10}$$

$$\mathcal{L}_{\text{QW}} = -\frac{1}{n} \sum_{i=1}^{n} \frac{1}{bt(i)} \sum_{j=1}^{bt(i)} \ln \sigma[s(x, y_i) - s(x, y_j)]^{ld(i,j)^{-1}} \tag{11}$$

3.3.2 Binary Classification System

Texts in input method scenarios are characterized by high contextual diversity and short length (less information in a single sentence), so it is challenging to rank different answers in one order because most of them can be correct in some specific contexts. For example, in IntelAssoc, when the query is "I haven't slept yet", the candidate answers "because I haven't finished my homework yet" and "because I drank too much coffee and am suffering from insomnia" may be the preferred answer for different users. However, for answers that are definitely impossible (e.g., "steak and black pepper go well together"), people can make a clear distinction without any doubt. Therefore, we can classify the answers into two classes, i.e., whether they are likely to be reasonable answers or not. In this system, we also use a fully automated labeling scheme where answers that have been selected by real users during the current three-month statistic cycle are labeled as positive answers. Based on this, we designed various RM training methods as follows.

Sample-Wise This is the sample-level training method based on each sample for a binary classification task, allowing the model to judge whether the current answer is correct or not given the specific task and query. The loss function worked as shown in equation 12, where y_i and $\hat{y}_i$ denote the ground truth label and model predict value.

$$\mathcal{L}_{\text{SW}} = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] \tag{12}$$

Class-Wise In the training method for category granularity, we expect models to acquire the ability to judge the correctness of an answer by taking the samples from each category as a whole and training the model so that its score for the correct answers is greater than that for the incorrect. The loss function is shown in Equation 13, where n and m denote the number of samples of correct and incorrect answers.

$$\mathcal{L}_{\text{CW}} = -\log \sigma(\frac{1}{n} \sum_{i=1}^{n} s(x, y_i) - \frac{1}{m} \sum_{j=1}^{m} s(x, y_j)) \tag{13}$$

Batch-Wise This is a kind of pair-level training method. For the data within a batch, we pair every correct sample with every incorrect one, then construct a pair-grained training loss. In this

method, it is not required that the queries of the two samples in the pair are the same because, under the binary annotation system, we believe that the model scores of any correct answer should always be higher than any incorrect answer, no matter what the query is. For example, in the IntelAssoc, the model should judge [query="早上好(Good morning)", answer="吃早饭了吗？(Have you had breakfast yet?)"] to be superior to [query="今天天气不错(it's a nice day)", answer="衬衫的价格是九榜十五便士(the price of the shirt is nine pounds fifteen pence)"]. For the loss function, please refer to Equation 14, where n and m refer to the number of correct and incorrect answers in a batch.

$$\mathcal{L}_{\text{BW}} = -\frac{1}{nm}\sum_{j=1}^{n}\sum_{j=1}^{m}\log\sigma[s(x,y_i) - s(x,y_j)] \quad (14)$$

Contra-Wise Contrastive learning has been verified to be effective in many areas of NLP, so we explored introducing supervised contrastive learning into RM training by using the categorization information of the samples as supervised signals and the loss function is constructed in line with Khosla et al. (2020).

In the reinforcement learning stage, we use the training method of the model "ppo-ptx" in Ouyang et al. (2022), i.e., combining ppo loss with pre-train loss. During the training process, we use the reward models trained based on the above methods to score the answers generated by fine-tuned Spark. The reward models serve as an approximation of human preferences to guide the optimization direction of the generative large language models.

4 Experiments

4.1 Experiments Settings

4.1.1 Public Datasets

PD dataset (Yang et al., 2012) and TP dataset (Zhang et al., 2017) are publicly available datasets commonly used to evaluate the P2C effect of input methods. PD is constructed based on the People's Daily corpus from 1992 to 1998. Meanwhile, TP is constructed from user chat logs collected by TouchPal IME. Each dataset contains 2K samples with perfect pinyin input only.

4.1.2 XF Datasets

Due to the lack of publicly available datasets, we constructed XF-datasets[5] consists of datasets that cover all tasks and training settings (as shown in Table 1).

SFT Datasets We built millions of training data for each of the three tasks. Additionally, we constructed test sets for IntelAssoc and ConvAssist with 2K samples each for manual evaluation of the final results. For the evaluation of FK2C effects under different keyboards of 26-key and 9-key, we constructed a test set of 57K, covering different input modes such as perfect pinyin, abbreviated pinyin, random abbreviated pinyin, and noisy input with different error types.

RM/RL Datasets For RLHF-IME, we constructed multiple datasets for reward modeling and the reinforcement learning phase. All the samples are derived from the user behavior of real users recruited via the Internet to participate in the user improvement program. These users only need to choose the most satisfactory one among multiple model-generated candidate results as ordinary users and do not need to do anything special to the rest bad results, not to mention the need to rank as in Ouyang et al. (2022). This consistency between annotated data and the real input method ensures the reliability of the data. Simultaneously, the simplicity of this annotation method significantly increases efficiency compared to using specifically trained annotators for ranking annotations, thereby reducing time and monetary costs. As described in Section 3.3, for both annotation systems, we generate labels in a fully automated way. Considering that the number of users participating in the user improvement program for ConvAssist is relatively small, so there are only datasets in the binary classification system for it.

4.1.3 Evaluation Metrics

We used a manual subjective metric - MOS for IntelAssoc and ConvAssist, where the model-generated results were independently scored by ten human evaluators with a range from 1 (worst) to 5 (best).

We use the precision of top-K (P@K) as the evaluation metric for the K2C task, which is often used in the past P2C tasks (Tan et al., 2022; Zhang et al., 2019), indicating whether the desired result is included in the generated top-K results.

[5]Publicly available at https://github.com/spirit-wang/Generative-Input/tree/master

	IntelAssoc		ConvAssist	FK2C	
	Binary	Rank	Binary	Binary	Rank
SFT					
Train Set	8M		6M	12M	
Validation Set	100K		100K	100K	
Test Set	2K		2K	57K	
RM					
Train Set	6.5M	2.5M	4.9M	8.4M	9.4M
Test Set	1.6M	0.6M	1.2M	2.1M	2.3M
RL					
Prompt Set	4.1M		1.9M	4.0M	

Table 1: Statistics of XF datasets.

For evaluating the RMs, we designed the following metrics for different annotation systems.

Accuracy-Rank Accuracy-Rank(Acc_R) is designed to evaluate RMs trained by the training method under the ranking annotation system. Its core idea is to compare how well the model's predicted scores match the ranked labeling information as displayed as Formula 15, where $score_i$ represents the model score for the i_{th} sample.

$$Acc_R = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{I}_{label_i>label_j}\mathbb{I}_{score_i>score_j}\cdot 1}{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbb{I}_{label_i>label_j}\cdot 1} \quad (15)$$

Accuracy-Binary Accuracy-Binary(Acc_B) is designed for the binary classification annotation system. Positive samples are paired with the negative ones and then we compare the scores given by the RMs and then calculate the accuracy as Equation 16, where N_{pos} denotes the number of positive samples, and $score(pair_i^{pos})$ denotes the model score for the positive sample in the i_{th} sample pair.

$$Acc_B = \frac{\sum_{i=1}^{N_{pos}\cdot N_{neg}}\mathbb{I}_{score(pair_i^{pos})>score(pair_i^{neg})}\cdot 1}{N_{pos}\cdot N_{neg}} \quad (16)$$

4.1.4 Base Models and Configs

In order to balance the model capability on various downstream tasks and the cost of hundreds of millions of calls per day in the input method scenario, we conduct experiments on the 2.6B version of Spark, an open-sourced LLM for Chinese [6](except for the RM) which has a GPT-like structure containing 32 layers of transformers and is equipped with strong text generation capability after pre-training with a large amount of Chinese corpus. For the RM, we use the Chinese version of DeBERTa-v2-large (Wang et al., 2022) as the foundation model, and then add 3 linear layers connected by the GELU (Hendrycks and Gimpel, 2016) activation function and the final output size is 1 to let the RM give a scaler value representing the reward score.

The SFT model is trained on 8 Atlas 800T A2 NPUs for about 1 week, the batch size is 128, we use a cosine annealing learning schedule with an initial learning rate of 1.6e-5, and we use the Adam optimizer with parameters of 0.9 and 0.95. The hyperparameters λ and α are set as 1.1 and 0.5, respectively.

In RLHF-IME, the Spark is trained on the same devices as the SFT model for 5 epochs and we set the batch size as 4096. We employ the AdamW optimizer (Loshchilov and Hutter, 2017) with a peak learning rate of 9e-5 and a 10% warm-up cosine scheduler. For reward modeling, we run experiments on 4 Atlas 800T A2 NPUs with smaller batch sizes (64 or 128) and learning rates (from 5e-6 to 1e-5) for different tasks.

[6] https://gitee.com/iflytekopensource

	PD		TP	
System	P@1	P@5	P@1	P@5
Google IME	70.9	78.3	57.5	63.8
On-OMWA	64.6	72.9	57.1	71.1
On-P2C	71.3	80.5	71.9	89.7
Pinyin-GPT	73.2	84.1	-	-
GeneInput	**88.4**	**96.2**	**77.0**	**92.9**
- align	88.1	95.9	76.4	92.5
- align - pyseg	82.1	92.4	70.1	88.6

Table 2: The results of the comparison between different methods on the PD and TP datasets.

4.1.5 Baselines

K2C Since existing research on input methods mainly focuses on solving the 26-key perfect pinyin input, we compare with the following baselines based on public datasets PD and TP. GoogleIME is a commercial Chinese IME. On-OMWA (Zhang et al., 2017), On-P2C (Zhang et al., 2019) and Pinyin-GPT (Tan et al., 2022) are some existing works on this task.

LLM ChatGPT and GPT4 are chosen as representatives of SOTA general LLMs to explore their performance on input method tasks.

4.2 Results and Analyses

4.2.1 K2C Results

Compared with Existing Methods In Table 2, the upper part is the baseline effect extracted directly from previous papers(Tan et al., 2022; Zhang

System	26-key P@1	26-key P@5	9-key P@1	9-key P@5
Perfect Pinyin				
Google IME	88.0	90.1	75.3	77.1
GeneInput	**94.2**	**99.5**	**92.0**	**98.4**
Abbreviated Pinyin				
Google IME	30.2	32.2	**2.4**	3.3
GeneInput	**67.0**	**86.7**	1.6	**4.6**
Random Abbreviated Pinyin				
Google IME	65.1	66.9	41.3	43.4
GeneInput	**81.5**	**95.3**	**73.4**	**88.4**
Pinyin with Noise				
Google IME	55.2	67.2	7.2	11.3
GeneInput	**75.2**	**90.7**	**46.2**	**67.5**

Table 3: Results of different input modes on XF dataset.

Method	IntelAssoc	ConvAssist	FK2C
Rank			
Query-Wise	68.5	-	73.5
Binary			
Sample-Wise	99.3	77.7	98.9
Class-Wise	93.0	73.1	81.2
Contra-Wise	97.7	67.9	97.9
Batch-Wise	**99.5**	**78.1**	**99.6**

Table 4: Results of reward modeling.

et al., 2019), and the lower part is the effect of our GeneInput. The P@1 and P@5 of our method significantly exceed the existing system effects on the two datasets. The results show that we have achieved full-mode k2C, and still have significant advantages over the existing single-mode modeling methods on traditional P2C tasks.

Results on Full-mode K2C Due to the lack of research work on full-mode modeling, we compare with googleIME, an open commercial Chinese input method, to verify the full-mode K2C effect of the model. In Table 3, our method performs significantly better than Google IME in each input mode except in the 9-key abbreviated pinyin mode. Especially on noisy input data, we also have very good performance. Since users rarely type in the abbreviated pinyin mode on a 9-key keyboard, we consider the effect on the 9-key abbreviated pinyin test set to be insignificant. The above results show that our full-mode unified modeling is successful.

4.2.2 RLHF-IME Results

RM Results Table 4 shows the results of RMs trained with training methods in Section 3.3. In the ranking system, we tested the models in different scenarios similar to Table 8 and the results indicated that the models perform highly consistently with human preference. In the binary classification system, Batch-Wise achieves the best results on all tasks, and surprisingly, the simplest method Sample-Wise performs nearly as well as it. Contra-Wise also achieves good results on IntelAssoc and FK2C. However, there is a fact that the texts in the input method scenario are flexible in context and short in length, which leads to a blurring of the boundaries of contrastive learning, so there is still a gap of about 3 or 4 points with Batch-Wise. Class-Wise performs the worst as expected because it is the method with the coarsest learning granularity, and it is difficult for the model to capture fine-grained sample-level preference characteristics when learning with inter-category differences. Based on the analysis, we finally decided to apply RM from Batch-Wise in the RL stage.

System	IntelAssoc	ConvAssist	FK2C
public			
ChatGPT	3.88	4.26	12.3
GPT-4	4.41	4.35	18.1
GeneInput			
Spark			
+ SFT	4.38	4.25	81.0
+ RLHF-IME rank	4.40	-	81.1
+ RLHF-IME binary	**4.43**	**4.52**	**84.6**

Table 5: Results of LLMs on IntelAssoc, ConvAssist and FK2C.

SFT/RL Results The performance of LLMs on IntelAssoc, ConvAssist and FK2C is provided in Table 5. Since ChatGPT and GPT-4 are not capable of understanding the relations between 9-key input sequences and corresponding Chinese sequences, we only show the average results on the 26-key test set of XF dataset for comparing the performance of FK2C among LLMs. Apparently, the existing LLMs for general purpose perform extremely poorly on FK2C and cannot meet the commercial product requirements, so it is necessary to propose our input-method-specific LLM. On IntelAssoc and ConvAssist, after SFT, Spark has been equipped with competitive capabilities, but there is still a gap with the state-of-the-art LLMs. Fortunately, after optimized with RLHF-IME, Spark outperforms GPT-4 on both tasks, especially on ConvAssist where the MOS score is even higher by 0.17. Based on the above results, it is clear that GeneInput is undoubtedly effective in improving the performance of LLMs in the domain of IMEs.

4.2.3 Abalation Analysis

To understand the effect of adding pyseg and soft alignment constraint, we performed some ablation experiments. As shown in Table 2, removing the soft alignment constraint has an average loss of about 0.5 points on P@1, while P@1 drops by about 6.5 points on both datasets if FK2C is modeled without pyseg. This result fully reflects the important role of adding pyseg in the training process. The soft alignment constraint is mainly to prevent some unacceptable errors in practical use, as well as to handle noisy inputs flexibly. Since the model effect has been greatly improved after adding pyseg, the improvement space of alignment constraints is significantly reduced. Thus, the contribution of the alignment constraint to P@1 is not obvious here, but it is crucial for the actual user experience.

For better exploring in RLHF-IME, we separately applied reinforcement learning with reward models trained based on the best method of each annotation system (Query-Wise for the ranking system and Batch-Wise for the binary classification system). According to the results displayed in the bottom half of Table 5), we can conclude that the binary classification system is more suitable for helping to optimize the input method LLM.

5 Conclusion

In this work, we explore how the next-generation generative paradigm **GeneInput** can be employed to uniformly model typical tasks within IMEs using generative models through prompts. We harness the text generation capability of the model to empower the input method system, extending their functionality beyond mere P2C and full-mode K2C is realized for the first time. Furthermore, we introduced RLHF-IME, a novel human preference alignment framework, allowing online model updates without the need for external annotated data, and achieved state-of-the-art performance across all downstream tasks. In the future, we plan to address more auxiliary input functions and further reduce the model size to make it capable of running efficiently on most smartphones while maintaining high performance.

6 Limitations

In RLHF-IME, we have verified that the binary classification system is better based on the experiments on IntelAssoc and FK2C. If a sufficient number of volunteers participate in ConvAssist, we will conduct experiments on all these tasks, thereby providing stronger support for that conclusion (if exists). Besides, we have not conducted large-scale experiments on input method personalization due to the limitations of the scale of personalized information extraction, and we will follow up with detailed experiments on personalization to draw definitive conclusions like those drawn on IntelAssoc, ConvAssist and FK2C. In addition, the input method editor is one of the most basic applications used throughout people's digital lives with or without access to the Internet on PCs or mobile devices. GeneInput will also explore incorporating model lightweight techniques to support real-time reasoning of terminal mobile devices to fit the actual use scenarios of input methods.

References

Hengyi Cai, Xingguang Ji, Yonghao Song, Yan Jin, Yang Zhang, Mairgup Mansur, and Xiaofang Zhao. 2018. Knptc: Knowledge and neural machine translation powered chinese pinyin typo correction. *arXiv preprint arXiv:1805.00741*.

Shenyuan Chen, Hai Zhao, and Rui Wang. 2015. Neural network language model for Chinese Pinyin input method engine. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 455–461, Shanghai, China.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv: Learning*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *ArXiv*, abs/2004.11362.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, pages 27730–27744.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2019. Portuguese named entity recognition using bert-crf. *arXiv preprint arXiv:1909.10649*.

Minghuan Tan, Yong Dai, Duyu Tang, Zhangyin Feng, Guoping Huang, Jing Jiang, Jiwei Li, and Shuming Shi. 2022. Exploring and adapting Chinese GPT to Pinyin input method. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1899–1909.

Junjie Wang, Yuxiang Zhang, Lin Zhang, Ping Yang, Xinyu Gao, Ziwei Wu, Xiaoqun Dong, Junqing He, Jianheng Zhuo, Qi Yang, Yongfeng Huang, Xiayu Li, Yanghan Wu, Junyu Lu, Xinyu Zhu, Weifeng Chen, Ting Han, Kunhao Pan, Rui Wang, Hao Wang, Xiaojun Wu, Zhongshen Zeng, Chongpei Chen, Ruyi Gan, and Jiaxing Zhang. 2022. Fengshenbang 1.0: Being the foundation of chinese cognitive intelligence. *CoRR*, abs/2209.02970.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Jinghui Xiao, Qun Liu, Xin Jiang, Yuanfeng Xiong, Haiteng Wu, and Zhe Zhang. 2022. Pert: A new solution to pinyin to character conversion task. *arXiv preprint arXiv:2205.11737*.

Shaohua Yang, Hai Zhao, and Bao-liang Lu. 2012. A machine translation approach for chinese whole-sentence pinyin-to-character conversion. pages 333–342.

Xihu Zhang, Chu Wei, and Hai Zhao. 2017. Tracing a loose wordhood for chinese input method engine. *arXiv preprint arXiv:1712.04158*.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2019. Open vocabulary learning for neural Chinese Pinyin IME. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1584–1594, Florence, Italy. Association for Computational Linguistics.

Rui Zheng, Shihan Dou, Songyang Gao, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Limao Xiong, Lu Chen, Zhiheng Xi, Yuhao Zhou, Nuo Xu, Wenbin Lai, Minghao Zhu, Rongxiang Weng, Wensen Cheng, Cheng Chang, Zhangyue Yin, Yuan Hua, Haoran Huang, Tianxiang Sun, Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. 2023. Secrets of rlhf in large language models part i: Ppo.

A Data and prompt

In Table 6, we provide the corresponding prompt and data examples for each subtask.

B Case Study

B.1 Personalization

For the personalization of the input method, we have done some simple case analysis. As shown in Table 7, for the same user inputs, we add different input expansions and can get different output results. For example, in IntelAsso, for the same input "I don't have time tonight", we add different user information to the input expansion, and then we can output different results that are more in line with the user's occupational characteristics, thus achieving personalized output. Similarly, in the FK2C task, according to different context information or user feature descriptions, corresponding more accurate results can be output. We leave a more detailed analysis as future work.

To prevent the leakage of user personal information, several measures are implemented. Firstly, users must provide explicit consent for the utilization of their personal data to enhance system performance. Secondly, all user personal information is solely used in its distributed representation form, and the original text is not retained. Finally, all data undergoes encryption procedures and is regularly cleared within fixed time intervals.

B.2 Reward Modeling

As mentioned in Section 3.3, the reward model determines the upper bound of RLHF performance to some extent. Therefore, we expect the model scores on different answers highly compatible with human preferences. Table 8 shows examples of our reward model scoring for Conversational Assistance in different test scenarios. It is clear that it has human-consistent perceptions of ordinary answers and those with high quality, shows significant resistance to profanity and irrelevant replies, and at the same time has good discriminative ability for answers to different generative tasks (e.g., the intelligent association answer in the table). Consequently, we firmly believe that it can give good feedback to the large language models for IME in line with human preferences during reinforcement learning.

C The importance of the FK2C task

We briefly mentioned the application value of FK2C in the Tasks part of the paper. In the following, we will introduce the necessity of FK2C in practical applications from several aspects:

Task	Prompt
IntelAssoc	根据用户已输入的句子，续写可能的下一句。用户输入：我今天没时间 Based on the sentence the user has already entered, write the next possible sentence. Input: I don't have time today 明天可以吗 How about tomorrow
ConvAssist	将用户输入的句子，改写成更加精彩的句子。用户输入：秋天来了 Rewrite the user's input into something better. Input: Autumn is coming 秋天到了，缤纷的树叶掉落一地，让我们一起享受这美好的季节吧！ Autumn is coming, colorful leaves fall down, let's enjoy this beautiful season!
FK2C	在输入法中，根据用户输入的按键序列，解码出合理的拼音切分与对应的中文结果。用户输入：woaiini Please decode the pinyin segmentation and corresponding Chinese characters according to the key sequence. Input: woaiini Wo'Aii'Ni，我爱你 Wo'Aii'Ni, I love you

Table 6: Examples of data and prompt for each task (the corresponding translations in English are provided below the sentences in Chinese).

Task	Input	Input Extend	Output
IntelAssoc	今晚没时间 There's no time tonight.	- 用户：学生 User: Student 用户：程序员 User: Programmer	明天可以吗 How about tomorrow? 作业还没写完呢 I haven't finished my homework yet 晚上还得加班呢 I have to work late at night
FK2C	wany	- 上文：来找我 Context: Come to me 用户：习惯全简拼输入 User: Used to abbreviate pinyin input 用户词：婉莹 User word: Wan Ying	万一 in case 玩呀 playing 我爱你呀 I love you 婉莹 Wan Ying

Table 7: Examples of IME personalization (the corresponding translations in English are provided below the sentences in Chinese).

Test Type	Query	Answer	RM Score
Good Conversational Assistance	早上好 Good morning.	宝贝早安喔，希望你今天一天都开开心心 Good morning, my love. I hope you are happy all the day.	0.974
Bad Conversational Assistance		嘿嘿，早上好哦 Hey, hey, good morning. (fondly)	0.941
Bad Language		去你妈的 Fuck you, man.	0.022
Irrelevant Words		你还爱他吗 Do you still love him?	0.083
Intelligent Association		昨晚睡得怎么样 How did you sleep last night?	0.139

Table 8: Examples of RM for Conversational Assistance scoring in different test settings with the same query (the corresponding translations in English are provided below the sentences in Chinese).

- First, commercial IME products may provide service for a billion users and their input habits may vary from person to person. To support various input patterns, the model behind commercial IME products must have FK2C capabilities.

- Second, even for the same user, although he is used to a particular input pattern, he will still use other input patterns in practice. For example, we found that with a 9-key keyboard, users tend to use both perfect pinyin and random abbreviated pinyin. With a 26-key keyboard, users tend to use perfect pinyin, perfect abbreviated pinyin, and random abbreviated pinyin. And noisy input can happen at any time.

- Of course, we would like to personalize it so that we can only show the results for a specific input pattern that the user is used to, but providing results for only one input pattern is risky as discussed above. Therefore, the model still needs the full-mode decoding capability.

In summary, the FK2C task has great application value and is the basis for commercial IME products to be able to be used in practice.

D Difference between Constrained Decoding and Alignment Constraint

Since we need to cover the full mode input scenario, alignment constraint differs from constraint decoding in two main points:

- The alignment constraint we propose is a kind of soft constraint, while the constrained decoding is a kind of hard constraint that can only be applied in extremely ideal input scenarios. When there is noise in pinyin, directly constraining decoding to restrict the character decoding space to the character subspace that conforms to the pinyin pronunciation does not yield any results. For example, "aii" does not have characters that conform to the pronunciation. Therefore, by calculating the edit distance between the input pinyin and the character pinyin, we impose error correction penalty for soft constraint, so as to ensure that the generated result does not completely deviate from the user's input intention, and has error correction ability for noisy input, and the error correction preference strength can be adjusted by hyper-parameters.

- We are a two-stage constraint and they have only one-stage constraint. Since assumes that user input is segmented pinyin, there is a clear positional alignment between input and output, and only the generated characters need to be constrained to conform to the corresponding pinyin pronunciation requirements. Our input is the actual keystroke sequence entered by the user, so in addition to imposing constraints between characters and pinyin, we also need to constrain the pinyin segmentation process, and this positional relationship is uncertain, so we constrain the generated pinyin to the prefix subset of the keystroke sequence.

Our alignment constraints are mainly designed to prevent some unacceptable errors in practical use, as well as flexible control of error correction capabilities for noisy inputs.