Exploring and Adapting Chinese GPT to Pinyin Input Method

Minghuan Tan[1*] and **Yong Dai**[2*] and **Duyu Tang**[2†] and **Zhangyin Feng**[2]
Guoping Huang[2] and **Jing Jiang**[1] and **Jiwei Li**[3] and **Shuming Shi**[2]

[1] Singapore Management University. [2] Tencent AI Lab. [3] Zhejiang University.
mhtan.2017@phdcs.smu.edu.sg, jingjiang@smu.edu.sg
{yongdai,duyutang,aifeng,donkeyhuang,shumingshi}@tencent.com, jiwei_li@zju.edu.cn

Abstract

While GPT has become the de-facto method for text generation tasks, its application to pinyin input method remains unexplored. In this work, we make the first exploration to leverage Chinese GPT for pinyin input method. We find that a frozen GPT achieves state-of-the-art performance on perfect pinyin. However, the performance drops dramatically when the input includes abbreviated pinyin. A reason is that an abbreviated pinyin can be mapped to many perfect pinyin, which links to even larger number of Chinese characters. We mitigate this issue with two strategies, including enriching the context with pinyin and optimizing the training process to help distinguish homophones. To further facilitate the evaluation of pinyin input method, we create a dataset consisting of 270K instances from fifteen domains. Results show that our approach improves the performance on abbreviated pinyin across all domains. Model analysis demonstrates that both strategies contribute to the performance boost.

1 Introduction

GPT (Radford et al., 2018, 2019) is a Transformer-based (Vaswani et al., 2017) language model that predicts tokens in an autoregressive manner. With a generic model architecture and the availability of vast web text data, GPT has been successfully developed for English, Chinese (Du, 2019; Zhang et al., 2021b), and many other languages. It shows extraordinary ability to generate fluent sentences and has been successfully applied to a wide range of natural language generation tasks. However, it remains unexplored to what extent GPT handles Chinese pinyin input method[1], which is used by hundreds of millions people when they enter Chinese characters on computers and cellphones.

Character	Perfect Pinyin	Initial	Final
我	wo	w	o
们	men	m	en

Table 1: Examples of initials and finals for Chinese characters "我们 (we)".

Pinyin input method allows users to enter Chinese characters based on their pronunciations. Given a pinyin[2] as the input, pinyin input method returns a list of Chinese characters pronounced with that pinyin. Fundamental elements of pinyin include initials (声母) and finals (韵母). In most cases, a Chinese character is spelled with one initial followed by one final. For example, as shown in Table 1, the initial and final for the Chinese character "我 (me)" are `w` and `o`, respectively. People may enter **perfect pinyin** (e.g., "`wo men`" for "我们"), where initials and finals of all Chinese characters are entered. There are about 420 perfect pinyin in common use. Sometimes, especially when multiple Chinese characters are entered at once, people may use **abbreviated pinyin** by only entering the initials of characters (e.g., "`w m`" for "我们").

This work, to the best of our knowledge, is the first one to explore the use of Chinese GPT for pinyin input method. We start by testing the performance of a frozen GPT. In this setting, we fix the parameters of GPT and predict Chinese characters from left to right in an autoregressive manner. At each time step, only characters pronounced with the same pinyin are legitimate candidates to be predicted. We find that, when the input is perfect pinyin, a frozen GPT performs comparably to state-of-the-art systems on the benchmark dataset (Yang et al., 2012). However, when the input is abbreviated pinyin with only initials of characters, the

[*] Work done during internship at Tencent AI Lab. * indicates equal contribution.

[†] Corresponding author.

[1] https://en.wikipedia.org/wiki/Pinyin_input_method

[2] https://en.wikipedia.org/wiki/Pinyin

Id	Context of Characters	Input Pinyin	Target	Pinyin Type
s1	我下周有时间，除了	`li bai yi you dian shi`	礼拜一有点事	Perfect
s2	我下周有时间，除了	`l b y y d s`	礼拜一有点事	Abbreviated
s3	老板帮我解决了难题，	`l b y y d s`	老板永远滴神	Abbreviated

Table 2: Illustrative examples of the task of pinyin input method with perfect pinyin and abbreviated pinyin. In s3, the input pinyin "`l b y y d s`" is the abbreviation of "`lao ban yong yuan di shen`". The translations of s1 and s3 are "I am free next week except for the next Monday." and "Boss helps me overcome the obstacle. You are the greatest of all time.", respectively.

performance of GPT has a drastic drop. A major reason is that an abbreviated pinyin maps to many perfect pinyin. For example, the initial "`w`" can be the abbreviation for "`wo`", "`wei`", "`wang`", "`wai`", "`wu`", etc. This would lead to exponentially larger number of legitimate candidates of Chinese characters. We mitigate this problem by incorporating pinyin information from two directions. One is to enrich the input by adding pinyin as additional context. The other is learning over pinyin-constrained vocabulary, which enhances the model's ability to distinguish between Chinese characters pronounced with the same pinyin.

To further facilitate the research on pinyin input method, we construct a new dataset based on the WuDaoCorpora (Yuan et al., 2021). Our dataset includes 270K instances from 15 commonly used news domains.[3] To evaluate towards multiple facets, the dataset covers instances with different numbers of context characters and pinyin. From our experiment results, we have these key findings:

1. On perfect pinyin, frozen GPT achieves state-of-the-art results.

2. On abbreviated pinyin, the performance of frozen GPT drops drastically. Context enrichment with pinyin and pinyin-constrained training both improve the performance.

3. The performance of GPT-based models increases as the context of Chinese characters becomes longer.

2 Task

The input of pinyin input method includes a sequence of Chinese characters $C = \{w_1, \dots, w_n\}$ as the context and a sequence of pinyin $P = \{p_{n+1}, \dots, p_{n+k}\}$, where $w_i \in \mathcal{V}_w$, $p_{n+j} \in \mathcal{V}_p$, and $\mathcal{V}_w$ and $\mathcal{V}_p$ are the vocabularies of words and pinyin, respectively. The output is a sequence of Chinese characters $O = \{w_{n+1}, \dots, w_{n+k}\}$, where $w_{n+i} \in \mathcal{V}_w$. The number of output characters is the same as the number of pinyin (i.e., k) and each character should be pronounced with the corresponding pinyin. The output sequence is desired to follow the context of Chinese characters to form a coherent sentence. As mentioned earlier in the introduction section, the input pinyin might be perfect (e.g., "`wo men`") or abbreviated (e.g., "`w m`"). Examples of the task are given in Table 2.[4] In our definition, one situation is that the context of characters is empty, which corresponds to the scenario that people are entering pinyin at the beginning of a sentence. The other situation is that the context includes real words, which stands for the scenario that people are entering pinyin in the middle of a written sentence.

In this paper, we assume that the oracle pinyin segmentation results are provided. Sometimes, a raw pinyin sequence can be mapped to different segmentation results. For example, the raw pinyin input "`jianshi`" can be segmented as "`ji an shi`" ("集安市", a city in the southwestern part of Jilin province, China) or "`jian shi`" ("见识", which is translated as "experience" in English). Pinyin segmentation is a subtask (Zhao et al., 2006; Zhou et al., 2007) of pinyin input method, which is well solved with the accuracy of 98% (Zhang et al., 2017). We leave the integration of pinyin segmentation as future work.

3 Models

In this section, we first introduce standard text-based GPT models adopted in this work (Section 3.1). Afterwards, we introduce how to extend GPT models for pinyin input method with enriched pinyin context (Section 3.2) and pinyin-constrained

[3]Our code and data will be released at `https://github.com/VisualJoyce/Transformers4IME`

[4]People may also input pinyin like "`l b y you dian shi`", we leave this as a future work.

training (Section 3.3), respectively.

3.1 GPT Baselines

In this work, we use character-level Chinese GPT as the backbone. We describe character-level GPT models in this subsection.

We start with a publicly available character-level GPT (Du, 2019)[5], which we call **GPT (public)**. The model has the same configuration as the standard 12-layer GPT[6]. It is trained on the CLUECorpusSmall dataset of 14GB (Xu et al., 2020), which consists of Chinese news, Wikipedia, online forum message, and consumer comments. We have tried another well known Chinese pretrained language model called CPM (Zhang et al., 2021b), which is trained on 100GB data. The vocabulary of CPM contains both Chinese characters and words.[7] We built a baseline with the CPM model of 12 layers[8] and forced the generated token to be a Chinese character. However, this baseline does not work well on pinyin input method, partly because our character-level decoding is inconsistent with the way how CPM is trained. It is promising to leverage the advantage of CPM on word-level decoding, and we leave this as a future work.

To build a stronger Chinese GPT baseline, we use GPT (public) as the starting point and further pretrain on a 800GB data crawled by us that is composed of news, Wikipedia, and novel texts. The model is trained with a batch size of 2,560 on 32x Tesla V100 GPUs. We adopt the Adam optimizer (Kingma and Ba, 2015) and set the learning rate to 1e-5 with a linear warmup scheduler. We run the warmup process for 10k steps and train 100k steps in total. We call this 12-layer GPT model as **GPT (ours)**.

To apply GPT (public) and GPT (ours) to pinyin input method, we use the traditional decoding pipeline of GPT to generate the sequence of Chinese characters in an autoregressive way. After encoding all the context of characters, the model predicts a Chinese character at each time step conditioned on the pinyin. Only Chinese characters pronounced with the same pinyin are legitimate candidates to be predicted. Without further clarification, this strategy is used in all the experiments.

3.2 Incorporating Pinyin Context

We explore two simple ways to incorporate pinyin information and build two models correspondingly. The first model uses pinyin information horizontally by concatenating pinyin input to the context of characters. The second model incorporates pinyin information vertically by adding a pinyin embedding layer at the bottom of GPT.

PinyinGPT-Concat In this model, we append a pinyin sequence to the context of Chinese characters. In the inference stage, the input has the form of $\mathbf{x} = [w_1, \ldots, w_n, \texttt{[SEP]}, p_{n+1}, \ldots, p_{n+k}, \texttt{[SEP]}]$, where `[SEP]` is a special token to separate text and pinyin. The model largely follows the architecture of the standard GPT. Since there is one-one relationship between pinyin tokens and generated Chinese characters (i.e., the pronunciation of w_{n+j} is p_{n+j}), we adjust the absolute positions of the characters to be generated. We assign the position of p_{n+j} to w_{n+j}, expecting the model to learn the alignments between pinyin and target characters.[9] We further expand the vocabulary of the word embedding layer by adding pinyin tokens.

In the training stage, given an training instance of $[w_1, \ldots, w_n, \texttt{[SEP]}, p_{n+1}, \ldots, p_{n+k}, \texttt{[SEP]}, w_{n+1}, \ldots, w_{n+k}]$, the model is trained to minimize the following loss function, where $\mathbf{w}_{<n+j}$ stands for the characters before w_{n+j} and $\mathbf{p} = [p_{n+1}, \ldots, p_{n+k}]$.

$$\mathcal{L}_{\text{concat}} = -\sum_{j=1}^{k} \log p(w_{n+j} | \mathbf{w}_{<n+j}, \mathbf{p}) \qquad (1)$$

PinyinGPT-Embed The original GPT model includes a word embedding layer and a position embedding layer. In this model, we add a pinyin embedding layer. The basic idea is to provide the model with the pinyin of the character to be generated next. Specifically, the embedding of each character is the sum of the token embedding of the current character, the position embedding of the current character and the pinyin embedding of the next character. When a word (e.g., numbers, punctuations and symbols) has no corresponding pinyin, we use a special token `[unk]` to represent it instead. The training process is similar with

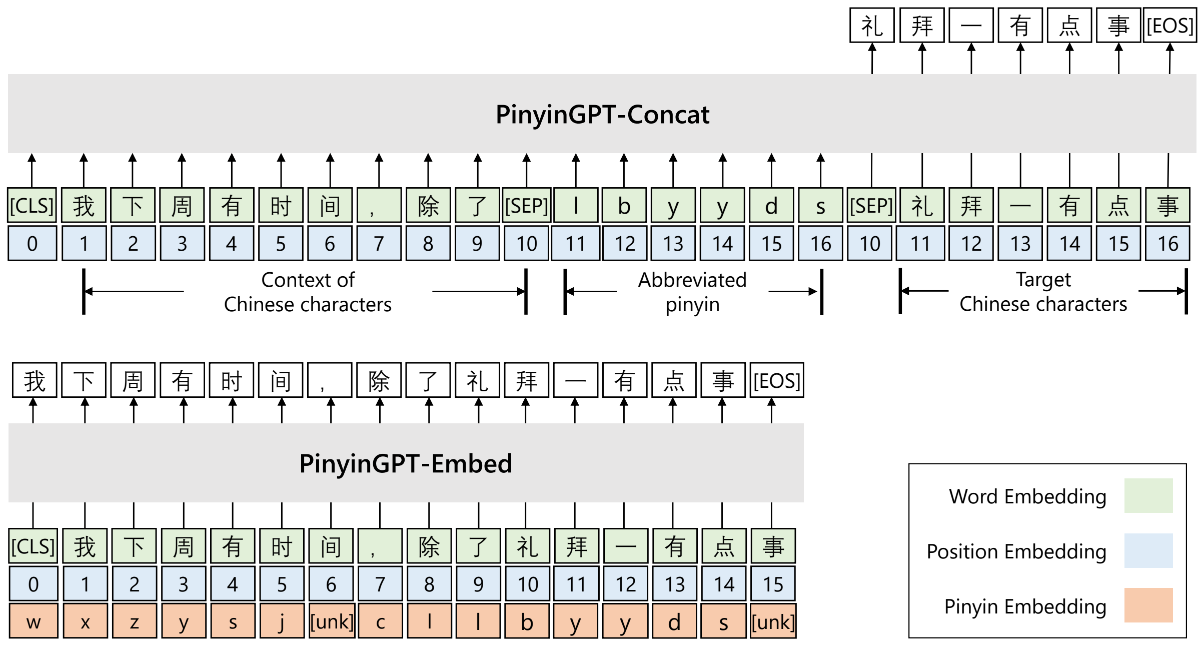
[5] https://github.com/Morizeyao/GPT2-Chinese

[6] https://huggingface.co/gpt2

[7] A Chinese word may consist of multiple Chinese characters. For example, the word “我们” (we) includes two characters “我” and “们”.

[8] https://github.com/TsinghuaAI/CPM-1-Distill

[9] On abbreviated pinyin, this strategy could bring 0.3 points in terms of P@5.



PinyinGPT-Concat
[CLS] 我 下 周 有 时 间 , 除 了 [SEP] l b y y d s [SEP] 礼 拜 一 有 点 事
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 10 11 12 13 14 15 16
礼 拜 一 有 点 事 [EOS]
Context of Chinese characters
Abbreviated pinyin
Target Chinese characters
PinyinGPT-Embed
[CLS] 我 下 周 有 时 间 , 除 了 礼 拜 一 有 点 事
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
w x z y s j [unk] c l l b y y d s [unk]
我 下 周 有 时 间 , 除 了 礼 拜 一 有 点 事 [EOS]
Word Embedding
Position Embedding
Pinyin Embedding

Figure 1: An illustration of the training process of Pinyin-Concat (top) and Pinyin-Embed (bottom), respectively. The example is same as the instance of s2 from Table 2.

the standard GPT, as shown in Figure 1. The loss function is given as follows.

$$\mathcal{L}_{\text{embed}} = -\sum_{j=1}^{n+k} \log p(w_j | \mathbf{w}_{<j}, \mathbf{p}_{<j+1}) \qquad (2)$$

In the inference stage, we transform the input sequence to the same format.

3.3 Pinyin-Constrained Training

We describe training details in this subsection. In standard GPT, the loss function is computed over the whole vocabulary. However, this is suboptimal for pinyin input method because the major challenge in the inference stage is how to select the best one from characters pronounced with the same pinyin (as described in the end of Section 3.1). This leads to inconsistency between training and inference stages. Therefore, in the training stage, the probability of a character is calculated over characters pronounced with the same pinyin, which is formulated as follows.

$$p(w_i) = \frac{\exp\left(g(w_i)\right)}{\sum_{w_j \in \mathcal{V}_{p_i}} \exp\left(g(w_j)\right)}, \qquad (3)$$

where $\mathcal{V}_{p_i}$ is the set of Chinese characters whose pinyin is p_i and g is the logit before the softmax layer.

4 Experiment

In this section, we show the results on pinyin input method over the two settings (i.e., perfect pinyin and abbreviated pinyin).

4.1 Settings

We describe the two datasets used in the following experiments and the evaluation metric.

PD Dataset PD dataset (Yang et al., 2012) is a commonly used benchmark dataset for the evaluation of pinyin input method (Jia and Zhao, 2014; Zhang et al., 2017; Huang et al., 2018; Zhang et al., 2019). The texts in PD are extracted from the People's Daily[10] from 1992 to 1998. It contains 5.04 million segments of consecutive Chinese characters (or Maximum Input Unit in some literature) for training and 2,000 segments for testing. For each test case, the input pinyin are all perfect pinyin and the context is null.

WD Dataset Since the PD data includes out-of-date news from 20 years ago and does not support us to study the scenario where the context includes real words, we construct a new dataset called WD. We use the WuDaoCorpora (Yuan et al., 2021) that contains 3TB Chinese corpus collected from 822 million Web pages. Currently, 200GB of the corpus

[10] http://www.people.com.cn/

has been made publicly available [11]. We randomly select 15 domains from WuDaoCorpora. For each domain, we first use an off-the-shelf Chinese segmentation toolkit (Zhang et al., 2020) to segment the documents into sentences. Then we automatically obtain the perfect pinyin and abbreviated pinyin of characters with pinyin converting tools. For each sentence, we randomly choose a context with a range from 0-3, 4-9 and 10+ words. Consecutively, we choose the target to be 1-3, 4-9 or 10+ words, respectively. It's further required that the target should be continuous characters that each has its own pinyin. We call each context-target length tuple like (4-9, 10+) as an evaluation configuration. For each configuration, we sample 2,000 test cases. In total, there are 9 configurations of 18,000 cases for each domain. The whole dataset consists of 270,000 examples. We investigate extremely long target lengths for the purpose of research that these configurations may not appear in real cases. All the instances in the WD dataset are only used for evaluation.

Evaluation Metric We use precision at top-K (P@K) as the evaluation metric, which is widely adopted in the literature (Jia and Zhao, 2014; Zhang et al., 2017, 2019). It measures if the ground truth exists in the top-K generated results. Some existing works also use keystroke-based metrics (Jia and Zhao, 2013; Huang et al., 2015) and human evaluation, which we don't use in this work because the evaluation process is more complex and time-consuming.

Other Settings We train both PinyinGPT models with the training data of GPT (ours). To preprocess the corpus, we use a public library *pypinyin*[12] to get the pinyin of Chinese characters.[13] We initialize both PinyinGPT models with GPT (ours). Both models are trained for 100k steps on 32 GPUs of NVIDIA V100 Tensor Core with a bach size of 25,000. The learning rate is 5e-5. We maintain a maximum of 128 tokens for every training example. We use a probability of 50% to sample a target sequence with less than 5 words, otherwise we randomly sample a target sequence with 6 to 25 words. This rate is empirically selected as it's less practical for users to type very long sequences.

[11]https://resource.wudaoai.cn/home

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

[13]If there are heteronym issues, we further verify them with an online dictionary ZDic (https://www.zdic.net/).

Model	P@1	P@5	P@10
Google IME	70.90	78.30	82.30
On-OMWA	64.40	72.90	77.90
On-P2C	71.30	80.50	81.30
GPT (public)	67.35	79.95	81.60
GPT (ours)	**73.15**	**84.10**	**85.45**

Table 3: Comparison with different methods over PD using perfect pinyin.

During inference stage, we use beam search with a beam size of 16 for text generation.

4.2 Results on Perfect Pinyin

We report results on the PD dataset (Yang et al., 2012). We use pinyin-constraint training in all configurations and train PinyinGPT models with different pinyin vocabularies for perfect pinyin and abbreviated pinyin, respectively. We compare with the following baselines.

- Google IME is a commercial Chinese IME which provides a debuggable API.
- On-OMWA (Zhang et al., 2017) is an online model for word acquisition which adaptively learns new words for Chinese IME.
- On-P2C (Zhang et al., 2019) is a neural pinyin-to-Chinese character conversion model, which is augmented by an online updated vocabulary to support open vocabulary learning.

In Table 3, the first group (top) shows the results of the aforementioned baselines, which are directly extracted from On-P2C (Zhang et al., 2019). The bottom group shows the performance of GPT (public) and GPT (ours) with frozen parameters. We can find that GPT (public) achieves comparative performance with existing systems in terms of P@5 and P@10. After being trained with a larger corpus, GPT (ours) surpasses all the baseline models in terms of all metrics. It is worth noting that existing baselines are supervised models that are fine-tuned on training instances. The results demonstrate the effectiveness of GPT models pretrained on vast amount of texts.

4.3 Results on Abbreviated Pinyin

In this section, we report results for both perfect pinyin and abbreviated pinyin on WD.

Model	Fix GPT Parameters	Perfect Pinyin			Abbreviated Pinyin		
		P@1	P@5	P@10	P@1	P@5	P@10
GPT (public)		76.55	87.07	88.58	22.22	29.99	31.48
GPT (ours)		80.22	90.20	91.09	26.90	35.56	37.03
PinyinGPT-Embed	Y	72.41	83.44	84.78	26.95	35.56	37.06
PinyinGPT-Embed	N	69.34	81.54	82.99	23.73	31.80	33.33
PinyinGPT-Concat	Y	**80.24**	90.21	91.10	26.91	35.56	37.03
PinyinGPT-Concat	N	78.12	**90.38**	**92.06**	**27.75**	**40.66**	**44.20**

Table 4: Overall results on WD dataset for perfect pinyin and abbreviated pinyin, respectively.

	Model	1-3			4-9			10+		
		P@1	P@5	P@10	P@1	P@5	P@10	P@1	P@5	P@10
0-3	GPT (ours)	30.11	42.27	45.25	13.33	18.24	18.99	4.16	5.86	6.00
	PinyinGPT-Concat	**31.72**	**48.09**	**53.94**	**15.21**	**24.39**	**26.94**	**5.58**	**9.22**	**10.09**
4-9	GPT (ours)	49.83	65.03	67.96	25.53	34.48	35.89	9.38	12.70	13.03
	PinyinGPT-Concat	**50.78**	**70.11**	**75.58**	**26.44**	**41.51**	**45.52**	**10.20**	**17.02**	**18.80**
10+	GPT (ours)	59.39	75.00	77.60	**35.42**	46.32	47.94	**14.96**	20.11	20.63
	PinyinGPT-Concat	**59.89**	**78.81**	**83.33**	34.99	**51.99**	**56.62**	14.93	**24.78**	**27.03**

Table 5: Results of different context-target configurations over WD for abbreviated pinyin. The first column and top row stand for context length range and target length range, respectively.

In Table 4, we list the overall experiment results of two GPT baselines as well as our PinyinGPT models. We have several findings based on the results. First, from each row, we can see that there is a drastic performance drop for all models. The reason is that each abbreviated pinyin can be mapped to a large amount of candidate characters, so that the problem is more challenging compared to perfect pinyin. We also believe that the evaluation metric of P@1 might be too strict for abbreviated pinyin because sometimes the top predictions might be correct (as reflected in Figure 3) even though they may be different from the ground truth. Second, adding pinyin information to GPT obtains limited improvement on perfect pinyin, but boosts the abbreviated setting by 5 points on P@5 and 7 points on P@10, respectively. Third, concatenating pinyin context horizontally is better than adding pinyin embedding vertically. Last, fine-tuning all the parameters performs better than keeping the parameters of GPT fixed.

4.4 Model Analysis: Ablation Study

In this section, we conduct experiments to understand the importance of pinyin context and pinyin-constrained training. Results are given in Figure 2. The baseline model is GPT (ours). The model + *Pinyin Context* means that we concatenate pinyin context (i.e., PinyinGPT-Concat) and learn over the whole vocabulary. The model + *Pinyin Context* + *PC-LOSS* means that we use both pinyin context and pinyin-constrained training. The figure shows that taking pinyin as extra context works well to improve results in terms of P@5 and P@10. When the two components are adopted, the performance is further improved.

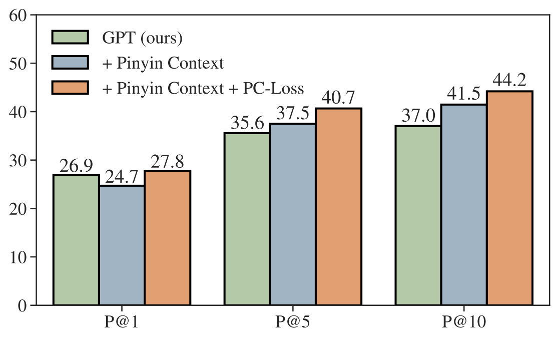


Figure 2: Ablation study for concatenating pinyin context and pinyin-constrained training.

4.5 Model Analysis: Context-Target Length

To analyze how context length and target length affect performance, we aggregate experiment results

Id	Case	Predictions	
1	**Context**: 奥斯卡组委会 **Pinyin**: qing xiang yu kan hao **Abbreviated**: No **Target**: 倾向于看好 **Translation**: *The Oscar Organizing Committee inclined to prefer*	**GPT (ours):** 1. 倾向于看好 *inclined to prefer* 2. 倾向于看豪 *inclined to look at*	**PinyinGPT-Concat:** 1. 倾向于看好 *inclined to prefer* 2. 倾向与看好 *tendency and optimism*
2	**Context**: 奥斯卡组委会 **Pinyin**: q x y k h **Abbreviated**: Yes **Target**: 倾向于看好 **Translation**: *The Oscar Organizing Committee inclined to prefer*	**GPT (ours):** 1. 旗下一款很 *one of its very* 2. 旗下一款豪 *one of its luxury*	**PinyinGPT-Concat:** 1. 倾向于看好 *inclined to prefer* 2. 倾向于抗衡 *inclined to fight against*
3	**Context**: 而中国队作为本次 **Pinyin**: j s d c b g **Abbreviated**: Yes **Target**: 竞赛的承办国 **Translation**: *And the Chinese team as the host country of this contest*	**GPT (ours):** 1. 决赛的承办国 *the host country of the finals* 2. 决赛的场边观 *at the ringside of the finals*	**PinyinGPT-Concat:** 1. 决赛的承办国 *the host country of the finals* 2. 竞赛的承办国 *the host country of the contest*

Figure 3: Case study for GPT (ours) and PinyinGPT-Concat in both perfect pinyin and abbreviated pinyin.

to form a matrix of accuracy for each configuration in Table 5. Each score is averaged over all the domains. From each column, we can see that longer context benefits both GPT and our model in pinyin input method, which verifies the power of context understanding ability of GPT models. An interesting finding is that, when the context is long enough (e.g., 10+), adding pinyin does not help improve the P@1.

4.6 Model Analysis: Case Study

We list three cases in Figure 3 to compare model outputs produced by GPT (ours) and PinyinGPT-Concat. The first case shows that, given perfect pinyin as the input, both GPT (ours) and PinyinGPT-Concat make the correct predictions. In the second case, abbreviated pinyin is given as the input. PinyinGPT-Concat makes the correct prediction while the prediction of GPT (ours) does not fit to the context well. In Case 3, even if PinyinGPT-Concat ranks the ground truth as the second best, the top 1 prediction still makes much sense and fit well with the context. In all cases, GPT (ours) usually generate predictions which are grammatically sound but semantically inappropriate.

4.7 Model Analysis: Domains

In this subsection, we analyze how performance differs with respect to domains. We sample six domains for illustration in Table 6.[14] The table shows that PinyinGPT-Concat achieves consistent improvement over GPT on all domains. We also find that the absolute scores vary a lot across domains. This reflects different predictability for texts on different domains. For example, the P@10 score of the *Culture* domain is 16 points lower than the *Medical* domain. In the Medical domain, the texts contain plenty of descriptions of symptoms and instructions of medicines, which are somehow canonically used. While in the Culture domain, the texts are less constrained and have more variations.

4.8 Model Analysis: Accuracy versus Latency

Considering pinyin input method requires both accuracy and efficiency, we further train a 6-layer GPT to investigate the trade-off. Our 6-layer GPT is directly truncated and initialized from the 12-layer GPT and is continually trained for 50k steps with the same configuration of 12-layer GPT.

The evaluation is conducted over the 9 configurations of context-target length and averaged across all domains. Specifically, each configuration is inferred using a data center GPU NVIDIA V100 Tensor Core, and the GPU is fully occupied by one model. The beam size is set to be 16. We report the average inference time in millisecond as well as accuracy in terms of P@K of PinyinGPT-Concat. Table 7 is the result for the configuration (4-9, 4-9). The table shows that the inference time of the model with 6-layer transformer is almost 30% faster than that with 12-layer. However, the performance of the 6-layer model drops consistently in the abbreviated setting. [15]

[14]The table of all 15 domains is attached in the Appendix.

[15]We also list the experiment results for all configurations

Model	Games			Culture			Sports		
	P@1	P@5	P@10	P@1	P@5	P@10	P@1	P@5	P@10
GPT (ours)	24.04	32.78	34.23	21.86	29.33	30.94	28.54	37.13	38.69
PinyinGPT-Concat	**25.78**	**38.26**	**41.89**	**22.10**	**33.33**	**36.72**	**29.81**	**43.56**	**46.95**
	Real Estate			Medical			Finance		
	P@1	P@5	P@10	P@1	P@5	P@10	P@1	P@5	P@10
GPT (ours)	26.53	35.27	36.74	33.59	43.54	44.93	29.00	37.24	38.47
PinyinGPT-Concat	**27.28**	**40.16**	**43.86**	**34.76**	**49.28**	**52.56**	**29.17**	**42.17**	**45.52**

Table 6: Performance of six sample domains over WD using abbreviated pinyin.

Model	Time (ms)	P@5
GPT (ours, 6L)	94	27.45
GPT (ours, 12L)	142	34.48
PinyinGPT-Concat (6L)	94	32.70
PinyinGPT-Concat (12L)	145	41.51

Table 7: Average inference time for one instance and the overall P@5 for the configuration of (4-9, 4-9).

5 Related work

Pinyin Input Method We describe existing works based on whether the input pinyin is perfect or abbreviated. A majority of existing works focus on perfect pinyin. Traditional models are typically based on statistical language models (Chen and Lee, 2000) and statistical machine translation (Yang et al., 2012). Recent works are usually built with neural network. For example, Moon IME (Huang et al., 2018) integrates attention-based neural network and an information retrieval module. Zhang et al. (2019) improves an LSTM-based encoder-decoder model with online vocabulary adaptation. For abbreviated pinyin, CoCAT (Huang et al., 2015) uses machine translation technology to reduce the number of the typing letters. Huang and Zhao (2018) propose an LSTM-based encoder-decoder approach with the concatenation of context words and abbreviated pinyin as input. Our work differs from existing works in that we are the first one to exploit GPT and verify the pros and cons of GPT in different situations. In addition, there are some works handling pinyin with typing errors. Chen and Lee (2000) investigate a typing model which handles spelling correction in sentence-based pinyin input method. CHIME (Zheng et al., 2011) is a error-tolerant Chinese pinyin input method. It finds similar pinyin which will be further ranked with Chinese specific features. Jia and Zhao (2014) propose a joint graph model to globally optimize the tasks of pinyin input method and typo correction. We leave error-tolerant pinyin input method as a future work.

in the Appendix. We recommend readers to select models in a more cost-effective way based on their requirements.

Pinyin-enhanced Pretrained Models Our methodology also relates to pretrained models that use pinyin information. Sun et al. (2021) propose a general-purpose Chinese BERT with new embedding layers to inject pinyin and glyph information of characters. There are also task-specific BERT models, especially for the task of grammatical error correction since an important type of error is caused by characters pronounced with the same pinyin. Zhang et al. (2021a) add a pinyin embedding layer and learns to predict characters from similarly pronounced candidates. PLOME (Liu et al., 2021) add two embedding layers implemented with two GRU networks to inject both pinyin and shape of characters, respectively. Xu et al. (2021) add a hierarchical encoder to inject the pinyin letters at character and sentence levels, and add a ResNet encoder to use graphic features of character image.

6 Conclusion

In this paper, we explore how to adapt pretrained Chinese GPT to pinyin input method. To begin with, we find that a frozen GPT with decoding conditioned on pinyin can reach state-of-the-art performance on perfect pinyin. However, in abbreviated setting, the performance drops by a large gap. Through our experiments, we find that both context enrichment with pinyin and pinyin-constrained training improve the performance. In the future, we would like to investigate more challenging settings including error-tolerant pinyin input method and mixtures of perfect pinyin and abbreviated pinyin.

References

Zheng Chen and Kai-Fu Lee. 2000. A new statistical approach to Chinese Pinyin input. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 241–247, Hong Kong. Association for Computational Linguistics.

Zeyao Du. 2019. Gpt2-chinese: Tools for training gpt2 model in chinese language. https://github.com/Morizeyao/GPT2-Chinese.

Guoping Huang, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2015. A new input method for human translators: Integrating machine translation effectively and imperceptibly. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 1163–1169. AAAI Press.

Yafang Huang, Zuchao Li, Zhuosheng Zhang, and Hai Zhao. 2018. Moon IME: Neural-based Chinese Pinyin aided input method with customizable association. In *Proceedings of ACL 2018, System Demonstrations*, pages 140–145, Melbourne, Australia. Association for Computational Linguistics.

Yafang Huang and Hai Zhao. 2018. Chinese Pinyin aided IME, input what you have not keystroked yet. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2923–2929, Brussels, Belgium. Association for Computational Linguistics.

Zhongye Jia and Hai Zhao. 2013. KySS 1.0: a framework for automatic evaluation of Chinese input method engines. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1195–1201, Nagoya, Japan. Asian Federation of Natural Language Processing.

Zhongye Jia and Hai Zhao. 2014. A joint graph model for Pinyin-to-Chinese conversion with typo correction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1512–1523, Baltimore, Maryland. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Shulin Liu, Tao Yang, Tianchi Yue, Feng Zhang, and Di Wang. 2021. PLOME: Pre-training with misspelled knowledge for Chinese spelling correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2991–3000, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Zijun Sun, Xiaoya Li, Xiaofei Sun, Yuxian Meng, Xiang Ao, Qing He, Fei Wu, and Jiwei Li. 2021. ChineseBERT: Chinese pretraining enhanced by glyph and Pinyin information. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2065–2075, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Heng-Da Xu, Zhongli Li, Qingyu Zhou, Chao Li, Zizhen Wang, Yunbo Cao, Heyan Huang, and Xian-Ling Mao. 2021. Read, listen, and see: Leveraging multimodal information helps Chinese spell checking. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 716–728, Online. Association for Computational Linguistics.

Liang Xu, Xuanwei Zhang, and Qianqian Dong. 2020. Cluecorpus2020: A large-scale chinese corpus for pre-training language model. *ArXiv*, abs/2003.01355.

Shaohua Yang, Hai Zhao, and Bao-liang Lu. 2012. Towards a semantic annotation of English television news - building and evaluating a constraint grammar FrameNet. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 333–342, Bali, Indonesia. Faculty of Computer Science, Universitas Indonesia.

Sha Yuan, Hanyu Zhao, Zhengxiao Du, Ming Ding, Xiao Liu, Yukuo Cen, Xu Zou, Zhilin Yang, and Jie Tang. 2021. Wudaocorpora: A super large-scale chinese corpora for pre-training language models. *AI Open*, 2:65–68.

Haisong Zhang, Lemao Liu, Haiyun Jiang, Yangming Li, Enbo Zhao, Kun Xu, Linfeng Song, Suncong Zheng, Botong Zhou, Jianchen Zhu, Xiao Feng, Tao Chen, Tao Yang, Dong Yu, Feng Zhang, Zhanhui Kang, and Shuming Shi. 2020. Texsmart: A text understanding system for fine-grained ner and enhanced semantic analysis. *arXiv preprint arXiv:2012.15639*.

Ruiqing Zhang, Chao Pang, Chuanqiang Zhang, Shuohuan Wang, Zhongjun He, Yu Sun, Hua Wu, and Haifeng Wang. 2021a. Correcting Chinese spelling errors with phonetic pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2250–2261, Online. Association for Computational Linguistics.

Xihu Zhang, Chu Wei, and Hai Zhao. 2017. Tracing a loose wordhood for chinese input method engine. *CoRR*, abs/1712.04158.

Zhengyan Zhang, Xu Han, Hao Zhou, Pei Ke, Yuxian Gu, Deming Ye, Yujia Qin, Yusheng Su, Haozhe Ji, Jian Guan, et al. 2021b. Cpm: A large-scale generative chinese pre-trained language model. *AI Open*, 2:93–99.

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.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An improved Chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 162–165, Sydney, Australia. Association for Computational Linguistics.

Yabin Zheng, Chen Li, and Maosong Sun. 2011. Chime: An efficient error-tolerant chinese pinyin input method. In *IJCAI*, pages 2551–2556. IJCAI/AAAI.

Xiaohua Zhou, Xiaohua Hu, Xiaodan Zhang, and Xiajiong Shen. 2007. A segment-based hidden markov model for real-setting pinyin-to-chinese conversion. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, page 1027–1030, New York, NY, USA. Association for Computing Machinery.

Model	Top1	Top5	Top10	Top1	Top5	Top10	Top1	Top5	Top10
	Entertainment			Automobile			Technology		
GPT (ours)	26.84	35.97	37.73	27.84	36.56	38.03	26.01	34.48	35.86
PinyinGPT-Concat	**28.74**	**41.68**	**45.48**	**28.74**	**41.55**	**45.28**	**26.82**	**40.17**	**43.65**
	Education			Agriculture			Economy		
GPT (ours)	27.31	36.71	38.28	26.57	35.08	36.59	27.93	36.04	37.20
PinyinGPT-Concat	**27.65**	**41.17**	**44.87**	**27.27**	**39.73**	**43.17**	**28.47**	**40.99**	**44.53**
	Games			Culture			Sports		
GPT (ours)	24.04	32.78	34.23	21.86	29.33	30.94	28.54	37.13	38.69
PinyinGPT-Concat	**25.78**	**38.26**	**41.89**	**22.10**	**33.33**	**36.72**	**29.81**	**43.56**	**46.95**
	International			Society			Military		
GPT (ours)	26.42	34.82	36.24	26.57	36.15	37.78	24.46	32.26	33.75
PinyinGPT-Concat	**27.49**	**40.16**	**43.66**	**27.34**	**40.94**	**44.89**	**24.82**	**36.73**	**40.03**
	Real Estate			Medical			Finance		
GPT (ours)	26.53	35.27	36.74	33.59	43.54	44.93	29.00	37.24	38.47
PinyinGPT-Concat	**27.28**	**40.16**	**43.86**	**34.76**	**49.28**	**52.56**	**29.17**	**42.17**	**45.52**

Table 8: Results of different domains over WD using abbreviated pinyin. Each score is averaged over all the context-target length configurations.

	Models	1-3				4-9				10+			
		T	P@1	P@5	P@10	T	P@1	P@5	P@10	T	P@1	P@5	P@10
0-3	GPT (ours, 6L)	38	26.74	38.45	41.50	98	10.46	14.41	15.19	201	2.72	3.70	3.85
	GPT (ours, 12L)	58	30.11	42.27	45.25	148	13.33	18.24	18.99	303	4.16	5.86	6.00
	PinyinGPT-Concat (6L)	40	29.17	45.17	50.73	98	11.92	19.55	21.84	197	3.20	5.67	6.22
	PinyinGPT-Concat (12L)	61	31.72	48.09	53.94	148	15.21	24.39	26.94	305	5.58	9.22	10.09
4-9	GPT (ours, 6L)	38	44.02	59.02	62.32	94	20.02	27.45	28.76	198	5.72	8.05	8.31
	GPT (ours, 12L)	57	49.83	65.03	67.96	142	25.53	34.48	35.89	301	9.38	12.70	13.03
	PinyinGPT-Concat (6L)	38	45.66	65.08	70.56	94	20.25	32.70	36.14	192	5.98	10.23	11.29
	PinyinGPT-Concat (12L)	58	50.78	70.11	75.58	145	26.44	41.51	45.52	298	10.20	17.02	18.80
10+	GPT (ours, 6L)	42	54.38	69.94	72.92	99	28.81	38.98	40.41	198	10.32	14.18	14.64
	GPT (ours, 12L)	64	59.39	75.00	77.60	149	35.42	46.32	47.94	301	14.96	20.11	20.63
	PinyinGPT-Concat (6L)	43	53.91	73.21	78.14	98	27.21	42.36	46.45	198	9.15	15.49	17.05
	PinyinGPT-Concat (12L)	66	59.89	78.81	83.33	154	34.99	51.99	56.62	306	14.93	24.78	27.03

Table 9: Experiment results for different configurations over WD using abbreviated pinyin, each score is averaged over all the domains. The first column is the context length while the first row is the target length. The field T is the average inference time in millisecond.