CSC311 Final Project

Enhancing Pinyin Input for Chinese Characters: An LLM Approach

Shuoer Wang[1], Yushan Xie[1], and Xiaotong Huang[1]

[1]Department of Computer Science, University of Toronto

May 13, 2023

Abstract

We have presented a novel approach to Pinyin input using a large language model that enhances the accuracy and efficiency of word recommendation, thereby improving the overall user experience. While Pinyin input has become an essential tool for Chinese language users worldwide, its use has also posed significant challenges, particularly in the case of homophonic words, dialects, and variations in pronunciation. Traditional methods, such as the Hidden Markov Model(HMM), have limitations on accuracy and cannot handle cases when English and numbers are mixed with Pinyin. Therefore, we explored the use of deep learning techniques, specifically the T5 transformer-based language model, to address these challenges. We utilized three datasets for training our model, and after data cleaning and preparation, we trained the T5 model on an A100 GPU. Our model achieved promising results, as evidenced by our hard and soft accuracy evaluation criteria. Our approach has the potential to enhance the user experience of Pinyin input and increase typing efficiency for Chinese language users worldwide. In the future, we plan to explore further improvements to our model, such as incorporating user-specific preferences and feedback to further personalize the input system, and analyzing chatting context to make more accurate prediction.

1 Introduction

Pinyin input has become an essential tool for Chinese language users worldwide. It allows for faster and more efficient input of Chinese characters by using the Roman alphabet. To use the Mandarin Pinyin keyboard method, the user types the Romanized pronunciation in Mandarin of the Chinese character, which generates a list of possible Chinese characters with the same pronunciation. The user then selects the desired character from the list and it is inserted into the text.

However, despite its many advantages, pinyin input has also posed significant challenges for users, especially when they are unable to find the correct corresponding Chinese words. This issue arises due to the vast number of homophonic words in the Chinese language, which can lead to confusion and errors when using pinyin input. Moreover, the diversity of Chinese dialects and variations in pronunciation further complicate the process of finding the correct words. For example, pinyin input "xian" can be mapped to 西安 (city name), 先 (first), 线 (wire), 现 (now) and many other Chinese characters depended on different context.

Hidden Markov model (HMM) is commonly used in Pinyin to analyze the context and predict the correct mapping from pinyin to Chinese Character. However, this method has limitation on its accuracy. Another major problem of this traditional method is it cannot handle the case when English is mixed in the pinyin. number keys are also not available since they are used to choose the right character. We hope the input system are more aware of the context and the meaning of the semantics of the input.

In recent years, deep learning techniques have revolutionized the field of NLP and led to the development of many powerful language models. As an example, ChatGPT is recognized for its impressive capabilities. Large language models have proven their abilities in context learning, generating human-like text and translation. After exploring and studying several popular transformer-based language models such as GPT-2, BERT, and T5, we have decided to use

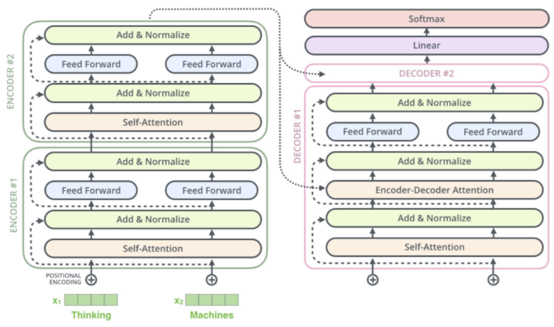


Figure 1: Illustration of t5 model [6]

T5 as our base model due to its dual encoder-decoder architecture (See Figure 1). This allows the model to effectively capture the dependencies between different parts of the input and output sequences. Basically, we want to do "language translation" between pinyin string and Chinese sentence.

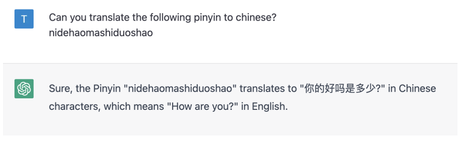
2 Literature Review

In general, there are two primary categories of methods for language understanding: rule-based methods and statistical methods. While rule-based methods rely on concrete linguistic information, statistical methods can effectively integrate abundant features, as well as automated learning and prediction.

One way to solve this problem is to apply a monotone phrasal Satisfiability Modulo Theories(SMT) based approach. Yang et. al begin by preparing a sentence-aligned corpus and performing the word alignment process. After extracting a translation table from the aligned corpus, they train a translation model using all of the features. Finally, they decode the source sentence to obtain the desired translation[1].

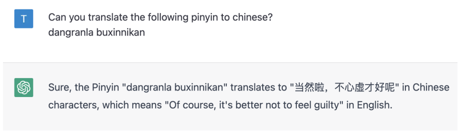
Another way to solve this problem is from a machine learning approach. Liu and Wang (2002) enhance the accuracy of distinguishing between Pinyin and Chinese characters by iteratively identifying new words and increasing the frequency of each word after segmentation. They also incorporate their implementation with unigram probability model and fewest words segmentation algorithms to reduce RAM usage. [2]

Moreover, there is an example-based and statistical approach for solving this problem. Chen and Lee(2000) utilize examples to generate candidate translations for a given Pinyin input. The approach involves several key steps, including word segmentation, example selection, and translation generation. After that, they integrated the model with a typing model to accommodate for typing errors.[3].


T
Can you translate the following pinyin to chinese?
nidehaomashiduoshao

Sure, the Pinyin "nidehaomashiduoshao" translates to "你的好吗是多少?" in Chinese characters, which means "How are you?" in English.

Figure 2: The correct output should be: 你的号码是多少 (What is your number)


T
Can you translate the following pinyin to chinese?
dangranla buxinnikan

Sure, the Pinyin "dangranla buxinnikan" translates to "当然啦，不心虚才好呢" in Chinese characters, which means "Of course, it's better not to feel guilty" in English.

Figure 3: The correct output should be: 当然啦, 不信你看 (Sure, see for yourself)

Li et al.(2014) proposed a linear re-ranking model that uses a minimum error learning method to combine different sub models, including word and character n-gram language models, part-of-speech tagging model, and dependency model, to address these issues. The impact of different sub models on the conversion is fully experimented and analyzed. Results from testing on the Lancaster Corpus of Mandarin Chinese show that the new model outperforms the traditional word n-gram language model. [4]

The most recent research for solving Pinyin to Chinese character problem is through proposing a neural P2C conversion model that incorporates an online updated vocabulary with a sampling mechanism to facilitate open vocabulary learning during IME operation. This approach has yielded a considerably better performance compared to previous ones. [5]

Our evaluation also included ChatGPT, which, unfortunately, demonstrated suboptimal performance on the given tasks. (See Figure 2 and Figure 3)

3 Data & Methods

To train our model, we utilized three distinct datasets: xiaohuangji - a casual chatting datalog [7], CLUECorpus2020 - a conversational dataset [11], and Baike2018QA - a collection of question and answer data that similar to Quora and wikipeida (but with Chinese) [8]. The Xiaohuangji dataset primarily consists of informal chatting logs, including personal names and casual language. CLUECorpus2020 is a large scale Chinese corpus collecting specifically for LLM pre-training, including corpus from news, comments, Wikipedia and social networks. In contrast, the Baike2018QA dataset is comprised of more formal language, with content that includes answers pertaining to topics such as math, law, and many other subject.

We selected around 30MB data from CLUECorpus2020, and together with around 33MB data from Baike2018QA. After obtaining datasets, we conducted data cleaning works such as removing empty lines. We separates the corpus into array of lines as labels, then we utilized the pypinyin package[9] to convert Chinese characters into Pinyin strings as input. From this, we have our inputs and output targets readied for training.

We applied a generalized model training structure for this project. Initially, we write our own InputDataset class which inherited the torch Dataset class, and specified a custom __getitem__ function. Following this, we used the T5Tokenizer and T5ForConditionalGeneration from the Transformers package to load the specific tokenizer and model for training. We then initialized the necessary components, including the data loader, optimizer, and linear schedule. With these components in place, we loaded the model, input data, attention mask, and labels into CUDA and started training. After each batch, we reset the optimizer gradients and performed a backward pass on the loss function.

Initially, we attempted to use the T5-small or T5-base model from Hugging Face. However, we discovered that the results were not ideal due to the fact that the model was not supported for Chinese language. Any Chinese words cannot be tokenized. So we then used another pre-trained Chinese T5 model called "Randeng-T5-784M-MultiTask-Chinese" [12], which is significantly larger (around 3GB) and took more time to train.

We rent an A100 from Lambda GPU Cloud, and run 3 epochs in total. The training process is about 35 hours (For this general model). We also trained a Xiaohuangji model which utilized only 45k lines of chatting logs from xiaohuangji dataset for comparison. The training process is around 5 hours.

We use other corpus in CLUECorpus2020 and Xiaohuangji as our test set to evaluate the training result.

For testing, we employed two criteria for evaluating the performance of our models:

1. hard accuracy, whereby every word in a sentence must be predicted correctly.

2. soft accuracy, calculated as the total number of correctly predicted words divided by the total number of words.

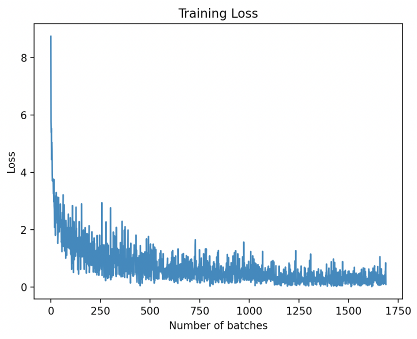
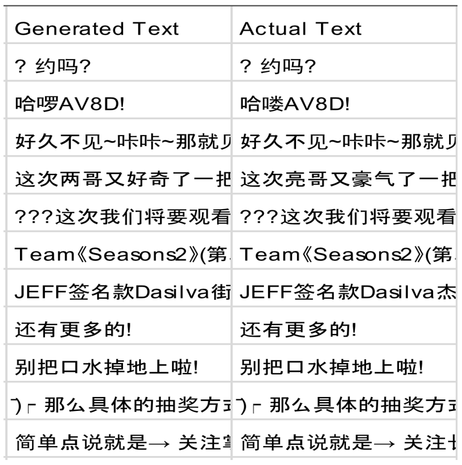


Figure 4: The training loss curve of our model



Generated Text	Actual Text
? 约吗?	? 约吗?
哈啰AV8D!	哈喽AV8D!
好久不见~咔咔~那就	好久不见~咔咔~那就
这次两哥又好奇了一把	这次亮哥又豪气了一把
???这次我们将要观看	???这次我们将要观看
Team《Seasons2》(第	Team《Seasons2》(第
JEFF签名款Dasilva街	JEFF签名款Dasilva杰
还有更多的!	还有更多的!
别把口水掉地上啦!	别把口水掉地上啦!
)┌ 那么具体的抽奖方	)┌ 那么具体的抽奖方
简单点说就是→ 关注	简单点说就是→ 关注

Figure 5: Example outputs

4 Results

Figure 4 is the training loss curve of xiaohuangji dataset. Figure 5 is an example output generated by the model.

Figure 6 illustrates the hard prediction accuracy versus the sentence length for testing with casual chatting data log (Xiaohuangji), using our general model.

The average hard accuracy for this test was determined to be 0.5417, with a soft accuracy of 0.8534. For long sentences (length $>=$ 20), the average hard accuracy is 0.2465, and the average soft accuracy is 0.8779.

The average hard accuracy for testing with chatting data log for xiaohuangji model is 0.691, and with a soft accuracy of 0.885. For long sentences (length $>=$ 20), the average hard accuracy is 0.452, and the average soft accuracy is 0.864.

Figure 7 illustrates the hard prediction accu-

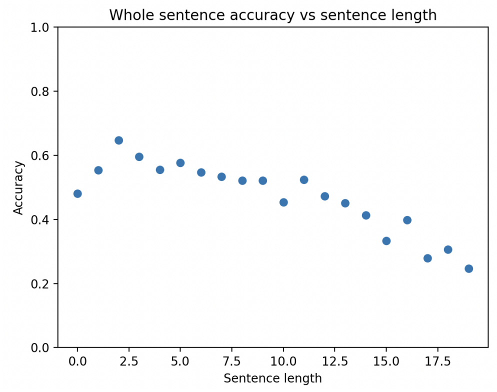


Figure 6: Predicting casual sentences with general model

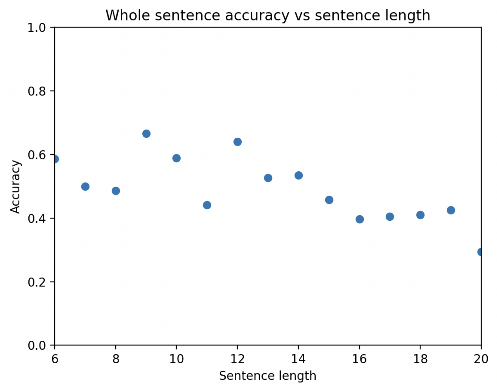


Figure 7: Predicting formal sentences with general model

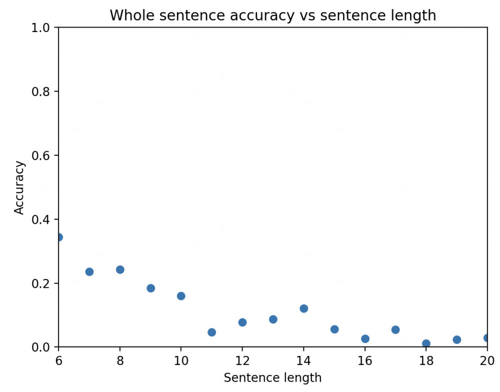


Figure 8: Predicting formal sentences with xiaohuangji model

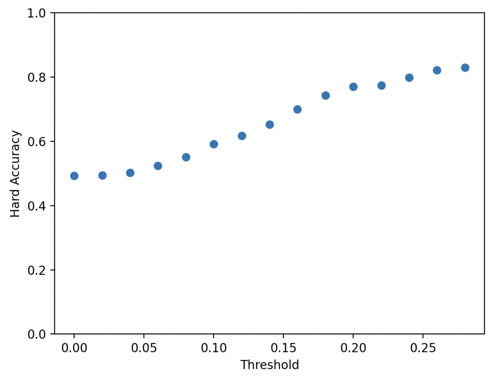


Figure 9: Threshold vs Hard Acc

racy versus the sentence length for testing with more formal sentences (CLUECorpus2020), using our general model.

Figure 8 illustrates the hard prediction accuracy versus the sentence length for testing with formal sentences, using our xiaohuangji model.

In terms of soft accuracy, the general model achieved a performance of 0.853 on casual chatting data and 0.886 on more formal language.

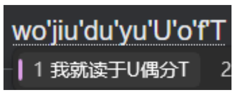
In addition, we tested an allowance threshold for the hard accuracy metric, whereby a sentence is considered correct if the ratio of correct words to the total number of words in the sentence is above a certain threshold. (general model)

5 Discussion

As large language models typically require extensive amounts of data and computational power to train, it was surprising to observe that this model performed better than initially anticipated. Based on the training loss curve, we have determined that a training period of 3 epochs is sufficient.

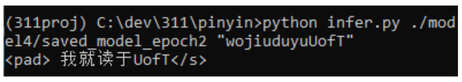
One advantage of this model is that it does not require word segmentation, meaning that each word in a sentence does not need to be split. Instead, a whole sentence with pinyin (without space) can be inputted.

Another notable advantage of this model is its natural support for multilingual input. It is capable of identifying sentences containing a mix of languages, including English, Chinese, symbols, and others, as long as the input can be tokenized. We have found inputs like: "wenzhongsuoyoutupianaiziTero Repo/Red Bull Content Pool" can be correctly mapped to 文中所有图片均来自Tero Repo/Red Bull Content Pool (All images in this article comes from Tero Repo/Red Bull Content Pool). We also conducted a comparison between our traditional input method and the output generated by our model. See



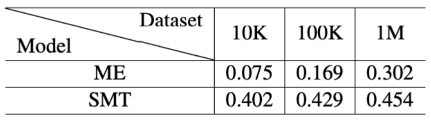
```
wo'jiu'du'yu'U'o'f'T
1 我就读于U偶分T   2
```

Figure 10: Traditional input method



```
(311proj) C:\dev\311\pinyin>python infer.py ./mod
el4/saved_model_epoch2 "wojiuduyuUofT"
<pad> 我就读于UofT</s>
```

Figure 11: Model generated output



Model \ Dataset	10K	100K	1M
ME	0.075	0.169	0.302
SMT	0.402	0.429	0.454

Figure 12: The whole sentence accuracy table for ME and SMT model[1]

Figure 10 and Figure 11.

Furthermore, the model's accuracy for long sentence inputs is considerably high, even when using noisy chatting data that contains personal names and incorrect word choices. With training set size of 45k, our fine-tune t5 model can achieve accuracy of 0.452 for long sentences (word length greater or equal to 20). Figure 12 illustrates the accuracy of whole sentence prediction for ME and SMT models, with respect to the length of training data set.[1] Thus, it is evident that our Pinyin-to-Chinese language model conversion requires a significantly smaller training set to attain high accuracy levels for long sentences.

As large language models typically require extensive amounts of data and computational power to train, it was surprising to observe that this model performed better than we initially anticipated. One of the primary limitations is the size of the model and the speed of its training. Training the model with a batch size of 8 requires 38 GB of GPU RAM and takes approximately 4 hours to complete when using a data set containing 50,000 corpus with 3 epochs (Using a A100 GPU on Google Colab). However, while T5-base does not officially support Chinese words, we opted for an unofficially trained T5 model that includes Chinese language support. It should be noted that this model is significantly larger than T5-base. (The T5-base model has a size of 400MB, while our unofficially trained T5 model is considerably larger at 3GB.) We believe a Chinese T5 model that has been fine-tuned and optimized can have a much smaller size, similar to t5-small model with size less than 250MB, which would make it possible to train this model on a non-professional GPU.

In terms of time complexity, the transformer model is commonly recognized to have a complexity of $O(n^2)$, which can be considered slow in certain contexts. However, with the help of GPU optimization, the self-attention layer is able to handle input sequences of 400-800 words in length (which is a typical length for articles or chats) with impressive speed and without quadratic trend [10]. Based on our testing, the time required for generating a prediction from a single Pinyin input string is 0.3 seconds. This is a reasonable time for word recommendation.

Another limitation of this model is that the predictions are highly dependent on the data used for training. In our case, we trained a model using chatting data, which includes many informal words and phrases. As a result, the model's predictions are less accurate for formal text such as news articles or formal reports. In our evaluation, we utilized a completely different dataset from our training set, and observed a decrease in long sentence accuracy to 0.233. (Our evaluation conditions are as follows: every word in the sentence must match the actual sentence, and the length of the predicted sentence must be greater than or equal to 20.) Therefore, it should be noted that a single model may not be suitable for use by multiple individuals with significantly different writing styles or working environments. Nevertheless, this characteristic can be beneficial in certain aspects, as it allows for accurate predictions of an individual's personal tone and writing style if the model is trained on their message input.

For future improvements, we believe that incorporating user-specific preferences and feedback to further personalize the input system, as well as analyzing the context of conversations to enable more accurate predictions are good approaches for improving our model. In addition, optimizing and reducing the size of the model is an important step in making our model more practical and efficient.

Conclusions

In conclusion, we explored a novel approach for Pinyin input that incorporates a large language model to enhance the accuracy and efficiency of word recommendation. This model works well in transforming long pinyin string input into Chinese sentences. One of the key advantages of our model is its ability to process input without requiring word segmentation, and it can naturally support multi language inputs. The transformer model's time complexity is optimized with GPU optimization, and it has a reasonable time for generating predictions. Furthermore, the model can accurately predict an individual's personal tone and writing style if trained on their message input. However, the model's size and training speed are limiting factors, and its predictions are highly dependent on the training data. Also, the size of the model we use exceeds the practical limitations, making it unfeasible to run in a real-world scenario, like mobile device. As a future work, we could explore the possibility of optimizing the model architecture to use a reduced number of parameters for the given task, and train it with larger dataset if computing resource are available.

References

[1] Yang, Shaohua, Hai Zhao and Bao-Liang Lu. "A Machine Translation Approach for Chinese Whole-Sentence Pinyin-to-Character Conversion." (2012).

[2] Bing-Quan Liu and Xiao-Long Wang, "An approach to machine learning of Chinese Pinyin-to-character conversion for small-memory application," Proceedings. International Conference on Machine Learning and Cybernetics, Beijing, China, 2002, pp. 1287-1291 vol.3, doi: 10.1109/ICMLC.2002.1167411.

[3] Here's another. Write these out as you would references anywhere else. The key I've assigned to this reference is 'other_ref' - therefore, a cite command in the body of your paper with 'other_ref' entered as the sole argument will automatically insert the appropriate reference number.

[4] Li, Xinxin & Wang, Xuan & Yao, Lin & Anwar, Muhammad. (2014). Linear Reranking Model for Chinese Pinyin-to-Character Conversion. Research Journal of Applied Sciences, Engineering and Technology. 7. 975-980. 10.19026/rjaset.7.344.

[5] Zhang, Z., Huang, Y., & Zhao, H. (2018). Open vocabulary learning for neural Chinese pinyin IME. arXiv preprint arXiv:1811.04352.

[6] Chen, Q. (2020, June 11). T5: A detailed explanation. Medium. Retrieved March 25, 2023, from https://medium.com/analytics-vidhya/t5-a-detailed-explanation-a0ac9bc53e51

[7] https://github.com/aceimnorstuvwxz/dgk_lost_conv/tree/master/results

[8] https://drive.google.com/file/d/1_vgGQZpfSxN_Ng9iTAvE7hM3Z7NVwXP2/view

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

[10] Sabran, A., &amp; Matton, A. (n.d.). CS224N project report faster transformers for text summarization. Retrieved March 26, 2023, from https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15839671.pdf

[11] Liang Xu and Xuanwei Zhang and Qianqian Dong. CLUECorpus2020: A Large-scale Chinese Corpus for Pre-training Language Model, from https://github.com/CLUEbenchmark/CLUECorpus2020/

[12] Junjie Wang, et al (2022). Fengshenbang 1.0: Being the Foundation of Chinese Cognitive Intelligence. CoRR, abs/2209.02970.

Files

GitHub repository:

1. https://github.com/ShuoerWang/pinyin

Our models:

1. General Model: https://drive.google.com/drive/folders/1_9A5FrClr-PRIWB7K1MObNsC8BWTz43I?usp=share_link

2. Xiaohuangji Model: https://drive.google.com/drive/folders/1y5DSDRcHzfVoKlv-OWSa76TI91ugVKOm?usp=share_link