PERT: A New Solution to Pinyin to Character Conversion Task

Anonymous ACL submission

Abstract

Pinyin to Character conversion (P2C) task is the key task of Input Method Engine (IME) in commercial input software for Asian languages, such as Chinese, Japanese, Thai language and so on. It’s usually taken as sequence labelling task and resolved by language model, i.e. n-gram or RNN. However, the low capacity of n-gram or RNN limits its performance. This paper introduces a new solution named PERT which stands for Bidirectional Pinyin Encoder Representations from Transformers. It achieves significant improvement of performance from baselines. Furthermore, we combine PERT with n-gram under Markov’s framework, and improve performance further. Lastly, the external lexicon is incorporated into PERT so as to resolve the OOD issue of IME.

1 Introduction

Some Asian languages, such as Chinese, Japanese and Thai language, can not be input directly through standard keyboard. User types in them via commercial input software, such as Microsoft Input Method (Gao et al., 2002), Sogou Input Method1, Google Input Method2, and so on. Pinyin is the official romanization representation for Chinese language. It’s natural for user to type in Chinese by pinyin sequence. For example, taking a snapshot from Sogou Input Method as Figure 1 shows, user inputs a pinyin sequence of wo men qu yong he gong from the standard keyboard and it is converted into the desired Chinese sentence of “我们去雍和宫”. Therefore, Pinyin to Character conversion (“P2C” for short) task is the key task of commercial input software.

Usually, the P2C task is taken as sequence labelling task and resolved by language model, i.e. n-gram. Specifically, there are three steps as Figure 2 shows. Firstly, the Chinese character candidates are generated according to the input pinyin tokens. All of candidates constitute of a lattice. Secondly, language model provides the probability between characters, i.e. \( P(我们 | 我) \), for later calculations. Thirdly, an optimal path with the highest probability in the lattice is found by the Viterbi algorithm. N-gram is the dominant model in the commercial input method software. However, its simplicity and low capacity limit the performance. In the trend of deep learning technique of recent year, more powerful models, such as RNN (Meng and Zhao, 2019; Yao et al., 2018; Wu et al., 2017), have been applied on the P2C task and achieve substantial improvement. The pre-training language models like BERT-CRF (Souza et al., 2019; Dai et al., 2019) are also applied to the sequence labelling task, such as named entity recognition (NER) and outperform n-gram and RNN significantly.

Inspired by the success of BERT-CRF, we design PERT (bidirectional Pinyin Encoder Representations from Transformers) especially for the P2C task. Instead of the pre-training technique as BERT-CRF, we train PERT directly on the P2C task on the massive pinyin-character parallel cor-
pus, and achieve substantial improvement of performance over n-gram and RNN. There are three contributions in this paper:

- We design PERT for the P2C task and achieve substantial improvement of performance.
- We combine PERT with n-gram under Markov framework and get further performance improvement.
- We incorporate external lexicon in PERT to solve the Out-Of-Domain (“OOD” for short) issue of IME.

2 Method

We describe our methodology in this section. Firstly, we present the implement of PERT in Section 2.1. Then we show how to combine PERT with n-gram in Section 2.2. Lastly, we incorporate external lexicon into PERT in Section 2.3.

2.1 PERT

Figure 3 shows the network architecture of PERT. There are two differences between BERT and PERT on architecture. Firstly, as shown in the bottom of Figure 3, PERT only takes the pinyin token as input since it is designed especially for the P2C task. Whereas, the BERT models for Chinese, such as ChineseBERT (Sun et al., 2021), take both pinyin and character as input because they are for the common tasks, i.e. Chinese Word Segmentation (CWS), Text Classification (TC) and so on. There is only 410 pinyin tokens (without tone) which is about 2x order of magnitude smaller than Chinese subword (usually about 50k). The scale of embedding layer of PERT is much smaller than BERT. Secondly, there are no segment embedding but only pinyin embedding and position embedding in PERT. PERT is trained directly on the P2C task on the massive pinyin-character parallel corpus, and there is no pre-training process. It dose not need Next Sentence Prediction (NSP) task, thus no segment embedding.

BERT adopts the pretraining-then-finetune paradigm so as to leverage the massive unlabelled text corpus to supplement limited labelled corpus of target task. However, for the P2C task, we can convert the massive character corpus into the according pinyin corpus accurately and efficiently, and build the massive parallel corpus. Then, PERT can be trained directly on the target P2C task on this corpus. It is called Text-to-Pinyin conversion (“T2P” for short) task (Zhang and Laprie, 2003) which is the contrary task of the P2C task. The similar tasks are the transliteration task (Kundu et al., 2018) and the phoneme-to-grapheme conversion task (Peters et al., 2017). Both of them are resolved in the sequence to sequence models. However, the T2P task is much simpler because most of pinyin information can be determined within its lexical context. We can firstly segment the Chinese character sequence into words and then get the correct pinyin token within its word context. The accuracy exceeds 99.9% (Zhang and Laprie, 2003) and there is the open toolkit named PYPinyin 3.

2.2 Combining PERT with N-gram

In practices, the commercial input software usually installs both n-gram model (for faster installation) and PERT (for higher performance) at the same time. In this section, we combine these two models together so as to get more capability. We start from the derivation of Bayes rule. Then we unify these two models under Markov’s framework.

In the P2C task, language model estimates the joint conditional probability of \( P(c_1...c_i|y_1...y_i) \), which is the probability of Chinese character sequence \( c_1...c_i \) given the input pinyin sequence \( y_1...y_i \). We can decompose it under the Bayes rule as Formula 1:

3https://github.com/mozillazg/python-pinyin
We further simplify these probabilities in two aspects. Firstly, we simplify \( P(c_1 \ldots c_{i-1} | y_1 \ldots y_i) \) to \( P(c_1 \ldots c_{i-1} | y_1 \ldots y_{i-1}) \) since \( y_i \) is in the future during user inputs. Secondly, we simplify \( P(c_i | c_1 \ldots c_{i-1}, y_1 \ldots y_i) \) into the product of \( P(c_i | c_1 \ldots c_{i-1}) \) and \( P(c_i | y_1 \ldots y_i) \) as the last line of Formula 1 shows. As presented in Figure 4, \( P(c_i | c_1 \ldots c_{i-1}) \) can be taken as the State Transition Probability of Markov framework, and can be estimated by n-gram. \( P(c_i | y_1 \ldots y_i) \) is the Emission Probability and can be estimated by PERT. \( P(c_1 \ldots c_{i-1} | y_1 \ldots y_{i-1}) \) is the history and can be further decomposed in the same way as above.

Note that in our implement, n-gram is estimated separately from PERT. Whereas, BERT-CRF estimates BERT and n-gram in an end-to-end way. It’s not applicable for BERT-CRF to the P2C task because of two reasons. Firstly, it usually requires different corpus to train n-gram and PERT separately in practices, i.e. the smaller dialog corpus for n-gram whose style is closer to user’s real input, but the whole huge corpus (including news, web-text and dialog) for PERT who has more parameters. It can not be trained in an end-to-end way. Secondly, the target estimation space of BERT-CRF is the whole Chinese characters whose number is more than 6k. It takes too much GPU memory and calculation resource during training, especially when calculating the normalization for CRF. So it’s infeasible to train BERT-CRF on a very large corpus.

### 2.3 Incorporating External Lexicon

The commercial IME engine adopts external lexicon to solve the OOD issue on edge device. For example, in Figure 5, we incorporate a external lexicon to enrich the vocabulary for PERT. We can observe that, for the case that the character “宫” (gong) is not contained in the training corpus, we can still output its probability by utilizing the external lexicon. The probability is calculated as:

\[
P_{\text{emit}}(宫) = P_{\text{py-bert}}(宫) P_{\text{py-bert}}(宫) P_{\text{py-bert}}(宫)
\]

We tried to train a BERT-CRF of the BERT-tiny scale with about 300M training text corpus on one V100 GPU. It takes more than five weeks to process about 6 epoch. However, it only takes about 15 minutes to train PERT-tiny under the same settings.
example, it adopts the computer lexicon to adapt to the computer domain, adopts the location lexicon to support some location names, i.e., “雍和宫 (Yonghe Lama Temple)”, adopts the user lexicon to fit for user’s input custom, and so on. Figure 5 illustrates our way to incorporate external lexicon in PERT. Specifically there are three steps. Firstly, we recognize the word item according to the external lexicon, and add it into the lattice of candidates, as shown in the left part of Figure 5. Secondly, we estimate the word probability by its component characters according to Formula 2. Here we adopt the geometric mean of probability according to the article (Nelson, 2017). Other metrics can also be tried. Finally, we search an optimal path together with the added word by Viterbi algorithm, as shown by the red arrow of Figure 5’s right part.

\[
P_{\text{emit}}(\text{雍和宫}) = \sqrt[3]{P_{\text{Py-bert}}(雍) \times P_{\text{Py-bert}}(和) \times P_{\text{Py-bert}}(宫)}
\]  

(2)

3 Experiment

3.1 Description of Data Set and Lexicon

As far as we know, there is no benchmark available to the P2C task. So we build our own data set and make it public to the community later. Table 1 describes the detailed information. These articles are collected from some major Chinese news websites, such as Netease, Tencent News and so on. Total 2.6M articles are taken as the training corpus, and another 1k disjoint articles as the test corpus. Besides, two additional corpus from the Baike website and the Society forum are chosen to evaluate the OOD performance. All these corpus are firstly segmented into sentences according to a punctuation list including comma, period, and so on. Secondly, the non-pinyin characters are filtered out, i.e. number, punctuation, English. Thirdly, they are further segmented by a max length (16 in our experiment) because user only types a few tokens once a time. Lastly, we convert them into the pinyin sequence by PYPinyin.

The Table of General Standard Chinese Characters containing more than 6k characters is taken as the basic lexicon in the experiments. Besides, the pinyin sequence by PYPinyin.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>#Articles</th>
<th>#Chars</th>
<th>#Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>News-Train</td>
<td>2,603,869</td>
<td>2,432,585,138</td>
<td>9.7G</td>
</tr>
<tr>
<td>News-Test</td>
<td>1000</td>
<td>926,792</td>
<td>3.7M</td>
</tr>
<tr>
<td>Baike-Test</td>
<td>49,357</td>
<td>6,238,834</td>
<td>30M</td>
</tr>
<tr>
<td>Society-Test</td>
<td>68,345</td>
<td>5,709,450</td>
<td>30M</td>
</tr>
</tbody>
</table>

Table 1: The Detailed Information of Corpus

Xiandai Hanyu Changyongcibiao (Common words in Contemporary Chinese, “Common-Words” for short) containing 0.1m items (Li and Su, 2008) and Tencent Network Lexicon (’Network-Words’ for short) published by Tencent Al Lab containing 8.8 million items are taken as the external lexicons.

3.2 Evaluation Metrics

In order to evaluate the performance, we use both character-level precision and sentence-level precision. The character-level precision is defined as the ratio that the IME engine converts to the Chinese characters correctly, as described in Formula 3.

\[
\text{Precision}_{\text{char}} = \frac{\#\text{correct}_{\text{converted char}}}{\#\text{total}_{\text{converted char}}}
\]  

(3)

Similarly we can define the sentence-level precision which is much stricter since it requires the correctness of whole sequence. Yet it is more meaningful in practice because the input software usually prompts the conversion of whole sequence in its first place and there is shortcut for user to choose that result.

Besides the performance, we also choose millisecond per token to evaluate the latency.

3.3 Baselines and Experiment Settings

Two kinds of language models, Bigram and LSTM, are taken as baselines.

- **Bigram.** Bigram is the De facto model adopted widely in the commercial IME engine. We build it on The Table of General Standard Chinese Characters on the training corpus presented in Table 1. No pruning strategy is adopted since the scale of corpus is large enough.

- **LSTM.** LSTM is reported that gets better performance than Bigram (Meng and Zhao, 2019; Yao et al., 2018; Malhotra et al., 2015) in the...
We also set its max length to 16 and gets more capability. Lastly, which gets 4 improvement on character-level precision.

Thus, it is reasonable to draw the conclusion that the experimental results are presented in Table 2.

<table>
<thead>
<tr>
<th>Model</th>
<th>#Parameter</th>
<th>Char_Precision</th>
<th>Char_Improvement</th>
<th>Sen_Precision</th>
<th>Sen_Improvement</th>
<th>ms/token</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bigram</td>
<td>7.4M</td>
<td>85.10%</td>
<td>NA</td>
<td>34.26%</td>
<td>NA</td>
<td>1.80</td>
</tr>
<tr>
<td>LSTM</td>
<td>4.2M</td>
<td>89.71%</td>
<td>4.61%↑</td>
<td>47.87%</td>
<td>13.61%↑</td>
<td>54.63</td>
</tr>
<tr>
<td>PERT-tiny</td>
<td>1.3M</td>
<td>85.18%</td>
<td>0.08%↑</td>
<td>32.01%</td>
<td>-2.25%↓</td>
<td>0.34</td>
</tr>
<tr>
<td>PERT-mini</td>
<td>5.0M</td>
<td>90.57%</td>
<td>5.47%↑</td>
<td>48.77%</td>
<td>14.51%↑</td>
<td>0.62</td>
</tr>
<tr>
<td>PERT-small</td>
<td>16.5M</td>
<td>95.41%</td>
<td>11.49%↑</td>
<td>64.38%</td>
<td>30.12%↑</td>
<td>0.63</td>
</tr>
<tr>
<td>PERT-medium</td>
<td>29.1M</td>
<td>95.59%</td>
<td>10.31%↑</td>
<td>65.74%</td>
<td>31.48%↑</td>
<td>1.11</td>
</tr>
<tr>
<td>PERT-base</td>
<td>91.1M</td>
<td>96.59%</td>
<td>11.49%↑</td>
<td>73.31%</td>
<td>39.05%↑</td>
<td>1.71</td>
</tr>
</tbody>
</table>

Table 2: Performances on the P2C Task. Char_Precision is the character-level precision; Char_Improvement is the improvement of character-level precision; Sen_Precision is the sentence-level precision; Sen_Improvement is the improvement of sentence-level precision. ms/token is the millisecond per token.

For PERT, we follow the specifications of Google from the tiny model to the base model. We also set its max length to 16 instead of 512, so as to be consistent to the training corpus.

### 3.4 Experiments on the P2C Task

The experimental results are presented in Table 2. Firstly, LSTM outperforms Bigram significantly, which gets 4.61% improvement on character-level precision and 13.61% on sentence-level precision. It confirms the previous conclusion in Yao et al. (2018). Secondly, PERT-tiny gets the comparable performance to Bigram with only one-sixth parameter number. And PERT-mini achieves much better performance than Bigram with similar yet smaller number of parameter. It outperforms LSTM too. It proves that PERT has a better network architecture and gets more capability. Lastly, as the scale increases, PERT gets better and better performances. PERT-base achieves 11.49% improvement on character-level precision and 39.05% on sentence-level precision from Bigram. It can be deployed in the resource-rich environment, i.e. on the cloud, so as to provide better services.

Table 2 also presents the inference speed. Bigram is evaluated on CPU of Xeon, and others are on GPU of V100. Although LSTM takes much more time than Bigram, it has already been deployed in the real products successfully (Meng and Zhao, 2019; Yao et al., 2018). PERT at all the scales takes much less time than LSTM because it can take fully advantage of parallelism of GPU. Thus, it is reasonable to draw the conclusion that PERT is deployable to real product.

For PERT, we follow the specifications of Google from the tiny model to the base model. We also set its max length to 16 instead of 512, so as to be consistent to the training corpus.

<table>
<thead>
<tr>
<th>Model</th>
<th>Char_Precision</th>
<th>+Bigram</th>
<th>Improved</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERT-tiny</td>
<td>85.18%</td>
<td>91.28%</td>
<td>6.10%↑</td>
</tr>
<tr>
<td>PERT-mini</td>
<td>90.57%</td>
<td>93.68%</td>
<td>3.11%↑</td>
</tr>
<tr>
<td>PERT-small</td>
<td>95.41%</td>
<td>96.18%</td>
<td>0.77%↑</td>
</tr>
<tr>
<td>PERT-medium</td>
<td>95.59%</td>
<td>96.63%</td>
<td>1.03%↑</td>
</tr>
<tr>
<td>PERT-base</td>
<td>96.59%</td>
<td>96.99%</td>
<td>0.40%↑</td>
</tr>
</tbody>
</table>

Table 3: Performances on Combining PERT with Bi-gram.

#### 3.5 Combining PERT with N-gram

Table 3 presents the experimental results that combines PERT with Bigram as described in Section 2.2. As we can see, the combined model outperforms the single PERT at every scale of model, which proves that our method can take effective use of the capability of each sub-model and get better performance. Not surprisingly, as the scale increases, PERT becomes more powerful and the gain from the combined model becomes smaller.

It’s Ok because the combined model is expected to deploy only on edge device as described in Section 2.2. In the resource-rich environments, we can just deploy PERT as large as possible.

#### 3.6 Incorporating External Lexicon

In this section, we evaluate PERT on two OOD corpus: the Baike corpus and the Society Forum corpus. Then two external lexicons are incorporated into PERT separately so as to improve the performance, as described in Section 2.3. The experimental results are presented in Table 4 and 5 respectively.

Firstly, PERT-tiny gets 83.40% precision on the Baike corpus in Table 4 and 80.26% on the Society Forum corpus in Table 5, whereas it gets 85.18% on the in-domain corpus as shown in Table 2. The performance drops dramatically, which indicates that the OOD issue is very severe to the commercial input software. Secondly, the precision is improved by 3.77% with the Common-words lexicon, and by 5.35% with the Network-words lexicon on
Table 4: Character-level Precision of PERT with External Lexicon on the Baike Domain. *Common-Words* is Xiandai Hanyu Changyongcibiao and *Network-Words* is Tencent Network Lexicon as described in Section 3.1.

<table>
<thead>
<tr>
<th>Model</th>
<th>Char_Precision</th>
<th>+Common-Words</th>
<th>Improved</th>
<th>+Network-Words</th>
<th>Improved</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERT-tiny</td>
<td>83.40%</td>
<td>87.17%</td>
<td>3.77%↑</td>
<td>88.75%</td>
<td>5.35%↑</td>
</tr>
<tr>
<td>PERT-mini</td>
<td>88.80%</td>
<td>90.04%</td>
<td>1.24%↑</td>
<td>91.24%</td>
<td>2.44%↑</td>
</tr>
<tr>
<td>PERT-small</td>
<td>91.28%</td>
<td>91.86%</td>
<td>0.58%↑</td>
<td>92.75%</td>
<td>1.47%↑</td>
</tr>
<tr>
<td>PERT-medium</td>
<td>92.41%</td>
<td>92.88%</td>
<td>0.47%↑</td>
<td>93.57%</td>
<td>1.16%↑</td>
</tr>
<tr>
<td>PERT-base</td>
<td>93.68%</td>
<td>94.12%</td>
<td>0.44%↑</td>
<td>94.53%</td>
<td>0.85%↑</td>
</tr>
</tbody>
</table>

Table 5: Character-level Precision of PERT with External Lexicon on the Society Domain. *Common-Words* is Xiandai Hanyu Changyongcibiao and *Network-Words* is Tencent Network Lexicon as described in Section 3.1.

<table>
<thead>
<tr>
<th>Model</th>
<th>Char_Precision</th>
<th>+Common-Words</th>
<th>Improved</th>
<th>+Network-Words</th>
<th>Improved</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERT-tiny</td>
<td>80.26%</td>
<td>84.04%</td>
<td>3.75%↑</td>
<td>86.21%</td>
<td>5.95%↑</td>
</tr>
<tr>
<td>PERT-mini</td>
<td>87.13%</td>
<td>87.89%</td>
<td>0.76%↑</td>
<td>89.52%</td>
<td>2.39%↑</td>
</tr>
<tr>
<td>PERT-small</td>
<td>90.16%</td>
<td>90.17%</td>
<td>0.01%↑</td>
<td>91.37%</td>
<td>1.21%↑</td>
</tr>
<tr>
<td>PERT-medium</td>
<td>91.52%</td>
<td>91.35%</td>
<td>0.17%↓</td>
<td>92.38%</td>
<td>0.86%↑</td>
</tr>
<tr>
<td>PERT-base</td>
<td>93.16%</td>
<td>92.97%</td>
<td>0.19%↓</td>
<td>93.61%</td>
<td>0.45%↑</td>
</tr>
</tbody>
</table>

The PERT trained on 10% corpus gets only 93.41% precision, which is dramatically lower than 96.59% of the model trained on the full corpus. Moreover, as the scale increases, the performance increases accordingly. It indicates that it always benefits from the increasing scale of corpus. However, the gain becomes more and more marginal. The PERT on 90% corpus performs almost as good as the full model.

3.7 Ablation on Scale of Corpus

In the experiments of above sections, we uses a large text corpus containing about 2432m Chinese characters, whose scale is comparable to Google BERT (about 3300m English words). It takes a lot of calculations as well as carbon footprint. In this section, we do the ablation study on the scale of corpus. We divide the whole corpus into ten pieces, and train PERT-base on them in an accumulated way in the same settings. The results are presented in Figure 6.

![Figure 6: Character-level Precision of PERT on Different Scale of Corpus](image)

4 Related Work

There are several technical approaches to solve the P2C tasks. It can be taken as sequence labeling task like POS tagging, or as seq2seq task like machine translation, or resolved by pre-trained language model.

4.1 Sequence Labeling Task

In industry, the P2C task is taken as sequence labeling task. N-gram (Bahl et al., 1983) is the De facto model adopted widely in commercial input method. Some smoothing methods, such as additive smoothing (Laplace, 1825), interpolation smoothing (Je-linek and Mercer, 1980), back-off smoothing (Katz, 1987), Kneser-Ney smoothing (Kneser and Ney, 1995), are adopted to solve the zero-probability problem. Laterly, the exponential models, such as Maximum Entropy Markov Model (“MEMM”
for short) (McCallum et al., 2000) and Conditional Random Field (“CRF” for short) (Lafferty et al., 2001), are proposed to improve the performance from n-gram. In recent year, under the trend of deep learning, RNN (Wu et al., 2017) gets more model capacity by capturing longer context. LSTM (Yao et al., 2018) is applied on the IME engine and achieves the improvements both in the P2C task and in the candidate prompt task. An incremental selective softmax method is further proposed to speed up the inference. Different from the above works, we adopt the bidirectional encoder representation of Transformer network. Trained on the massive parallel corpus, PERT gets significant performance improvement from n-gram as well as LSTM on the P2C task.

4.2 Pre-trained Language Model on P2C Task

Recently, the emergence of pre-trained models (PTMs) has brought natural language processing to a new era. BERT (Devlin et al., 2019) takes the bidirectional encoder representation of transformer networks. It is firstly pre-trained on some self-supervised tasks on the massive unlabelled corpus, such as Masked Language Modeling (“MLM” for short) and Next Sentence Predicting (“NSP” for short). Then it is fine-tuned on the target tasks on the small labelled corpus. It achieves the SOTA results on many tasks, including the sequence labeling task. Zhang and Joe (2020) applies pre-trained language model to the P2C task and proposes BERT-P2C. In the experiments, BERT-P2C outperforms other pre-trained models such as ELMO. Different from BERT-P2C, we train PERT directly on the target P2C task by creating the massive labelled corpus instead of the pretrain-then-finetune paradigm, as described in Section 2.1. Besides, comparing with Zhang and Joe (2020) which costs the parameters of two encoders and one decoder of transformers, PERT only takes one encoder whose parameter is much smaller (about one third) than BERT-P2C. Moreover, PERT also outperforms BERT-P2C in the P2C experiments. We describes the experimental results in details in Appendix A.

BERT-CRF (Souza et al., 2019; Dai et al., 2019) further employs the CRF decoder on the top of BERT encoder, so as to exploit the structure information of target sequence. It is trained in an end-to-end way and achieves the SOTA result on the NER task of Portuguese. Different from BERT-CRF, we train PERT and n-gram separately for two reasons as described in Section 2.2. Firstly, it is required some flexibility when deploying on real product. Secondly, it costs too much to calculate normalization of BERT-CRF on the P2C task.

ChineseBERT (Sun et al., 2021) exploits some information specific to Chinese, such as glyph and pinyin, so as to enhance its capability on Chinese language. It gets improvement from Google-BERT on a variety of Chinese NLP tasks, such as machine reading comprehension, text classification, named entity recognition and so on. PLOME (Liu et al., 2021) designs the pre-trained tasks specific to glyph and pinyin in the input text. It is applied on Chinese spelling correction task and gets the SOTA result. For these models, pinyin is only the auxiliary information which has to be input together with glyph and text. It can not be input alone, as the P2C task requires. However, PERT only takes pinyin as input, and it’s especially designed for the P2C task.

4.3 Machine Translation Approach

In academic community, the P2C task is also considered as machine translation task (Zhu et al., 2020; Zhang and Joe, 2020; Zhang et al., 2019; Meng and Zhao, 2019) in which pinyin is taken as source language and Chinese character is target language. It’s usually resolved in an encoder-decoder network like Transformer. In practice, the uni-directional decoder is usually the efficiency bottleneck during inference. Then the Non-AutoRegressive (“NAR” for short) decoder with bi-directional attentions is proposed to solve this problem (Gu et al., 2018; Kasner et al., 2020; Gu and Kong, 2021).

In the P2C task, the number of input pinyin tokens is exactly the same as the number of output Chinese characters, which is an explicit constraint that we can utilize. Thus it’s more natural to take it as sequence labeling task as this paper presents. In the future, we can adopt the encoder-decoder solution when we take the continuous letter-level input (without pinyin segmentation before P2C) or the sub-pinyin token input (segmenting input letter sequence into sub-pinyin token sequence) whose lengths are much longer than the length of target character sequence. These models are expected to be more tolerated to input error. We leave them in our future works. Besides, comparing PERT
with NAR machine translation system, there is no heavy decoder used in PERT. Thus, the number of PERT parameter is about a half of NAR machine translation system.

5 Conclusions

In this paper, we propose PERT for the P2C task which is crucial to the IME engine of commercial input software. In the experiments, PERT outperforms n-gram as well as LSTM significantly. Moreover, we combine PERT with n-gram under Markov framework and get further improvement. Lastly, we incorporate external lexicon into PERT so as to resolve the OOD issue of IME.

6 Future Work

There are several ideas to try in the future. One interesting idea is to replace the current pinyin-level input with the letter-level input, and convert it into Chinese character sequence by the seq2seq model as discussed in Section 4.3. The built model is expected to be more tolerant to input error, and makes the IME engine more robust. Online learning is another interesting and important topic. 

Zhang et al. (2019) designs an efficient method to update the vocabulary during the user input process, and augment the IME engine with the learned vocabulary effectively. However, Zhang et al. (2019) adopts RNN as encoder, which is usually regarded as less capability than Transformer architecture. We are going to incorporate PERT with the online learning method to get further improvement. Lastly, we are going to deploy our model to real product, and compare performance with some commercial input software on real input of user (Meng and Zhao, 2019; Huang et al., 2018). It needs budget and many engineering works. We are working on it.

References


A Comparison with BERT-P2C

We compare the performance of PERT with BERT-P2C (Zhang and Joe, 2020) in this section. Like BERT, BERT-P2C is firstly pre-trained on the
### Case 1

**Pinyin Sequence:** zhi shi dai le yi fu mo jing  
**Bigram Result:** 只是戴了一副墨镜 (Only wear a sunglasses)  
**LSTM Result:** 只是戴了一副墨镜 (Only wear a sunglasses)  

### Case 2

**Pinyin Sequence:** shi wan jia neng gou qing song ying dui  
**Bigram Result:** 走玩家能够轻松应对 (Is player handle it easily)  
**LSTM Result:** 使玩家能够轻松应对 (Enable player handle it easily)  

### Case 3

**Pinyin Sequence:** ji qing liang you shi shang  
**Bigram Result:** 及清凉有时尚 (And cool has fashion)  
**LSTM Result:** 既清凉又时尚 (Both cool and fashion)  

<table>
<thead>
<tr>
<th>Case 1</th>
<th>Pinyin Sequence</th>
<th>er shi gong gong wei sheng fang kong de wen ti</th>
<th>LSTM Result</th>
<th>PERT Result</th>
<th>(Twenty problems of public health prevention and control)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 2</td>
<td>Pinyin Sequence</td>
<td>zhe ge shi pin hen chong qing</td>
<td>LSTM Result</td>
<td>PERT Result</td>
<td>(The second problem is about public health prevention and control)</td>
</tr>
<tr>
<td>Case 3</td>
<td>Pinyin Sequence</td>
<td>man wei zhong gu yi fa shi</td>
<td>LSTM Result</td>
<td>PERT Result</td>
<td>(The role of intentional pledge in Marvel is awesome)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 7: Case Studies on Comparing LSTM with PERT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Case 1</strong></td>
</tr>
<tr>
<td><strong>Case 2</strong></td>
</tr>
<tr>
<td><strong>Case 3</strong></td>
</tr>
</tbody>
</table>

### B Error Analysis

In this section, we choose some cases from the experiments and get in-depth analysis.

Table 6 compares Bigram with LSTM. In Case 1, the conversion of "dai" is determined by the last word of "墨镜 (sunglasses)". Bigram can not capture such a long context and it gets an incorrect conversion of "戴 (wear)". Whereas, LSTM can model the whole sentence and get the correct conversion of "戴 (wear)". It’s similar for the pinyin "shi" in Case 2 which can be predicted by the word of "应 (handle)". Moreover, LSTM can also recognize some fix collocation like "既...又... (both...and...)", and get the result correctly, as shown in Case 3.

In a word, Bigram can only captures local conversion, which limits its performance on the P2C task. However, LSTM can take the whole sequence into consideration, and get improvement from Bigram.
Table 7 compares LSTM with PERT. In Case 1, both conversions are grammatically correct. However, from the semantic point of view, the result of PERT makes more sense. In Case 2, the city name of “重庆(chongqing)” acts as an adjective which means *the style or fashion like Chongqing* in the context. PERT can capture these semantic information and get the correct conversion, whereas LSTM can not. Moreover, we also find that PERT can handle named entity in sentence much better than LSTM, as shown in Case 3. All these cases prove that PERT can better understand the semantic information in sentence and thus get better performance.