Improve Japanese Input Method by Re-rank Candidate Words

Xinyang Jiang, Weitsung Lin, Gang Qiao, Xiaoliang Xu, Jianmin Wu, Cong Zhang

GBU, Baidu Inc, China, 518052
E-mail: {jiangxinyang, v_liweicong, qiaogang01, xuxiaoliang, wujianmin01, zhangcong05}@baidu.com

Abstract

We investigate a Two Stream Encoder model (TSE) for re-ranking candidate words in Japanese Input Method, which encodes Hiragana sequences and word sequences to represent the relationship between Hiragana sequences, context and candidate words. Compared with our current online model, the transformer-based model can capture more context information. Two auxiliary language model losses are introduced to help model learning, and the loss weights are dynamically adjusted by linear decay.

TSE is empirically effective and achieves a Top-1 accuracy of 82.3%, which is significantly higher than the 78.6% of the online model on the testing set. While there are only 2.43 million of parameters and 59.57 second inference time per 10000 reranks, the memory consumption when running on the mobile phone is only 12MB, let the model run smoothly on the mobile devices.

1 Introduction

Typing is one of the core methods that users interact with smartphones. Unlike other devices with keyboard and large screen, typing on mobile devices have more restrictions like limited input buttons and small space for showing candidate words. An intelligent input method to provide better text suggestions is important to boost typing efficiency on such devices.

In languages like Japanese and Chinese, there is an essential text suggestion task: conversion task. Conversion task is not a common scenario for Latin based languages which could directly type on a keyboard, but there are thousands of characters in Japanese and Chinese language. For example, in Japanese users type Hiragana sequence, a type of phonography, and conversion task converts it to a list of target word sequences for users to select from.

An n-gram language model is used to solve this problem. However, due to the exponential increase in model size, the n-gram model can only support up to bi-gram or tri-gram on mobile devices. With this constraint, the n-gram model often suffers from loss of accuracy because of its limitation on considering length of context. On the other hand, recently deep-learning based language models are shown that they can solve such issues more efficiently.

Although deep-learning based language model has achieved enormous success, it is still not used mainly for input method tasks. The major reason is that input methods have to run smoothly and interactively on mobile devices with limited resources. Unlike speech recognition or machine translation services served from online servers, input method applications have to sacrifice accuracy for execution speed and model size.

In this work, we focus on ranking the target word list to provide the best fit for users' original intention candidates in top 3 to top 6. We customized a natural language processing (NLP) model, Transformer [1], a self-attention mechanism model, to fit Japanese input method application. By introducing auxiliary training objectives we have benefit for learning better encoded representations. Also, we leverage byte pair encoding (BPE) [2] to segment and encode input context. BPE not only helps reduce the length of model input but also improves execution speed. Therefore, our model considers longer context distance and achieves a reasonable execution speed on mobile devices.

To sum up, our contributions are as follows:

- We customized a Two Stream Encoder model to improve re-rank candidate accuracy in input method.
- The model can run on mobile devices at a reasonable execution speed.

2 Input Method with Deep-learning Model

In the input activity, the user first types the Hiragana sequence. Then the possible candidates show up for the user to choose. Because we don't know when the user stops typing and chooses the candidates, we need to calculate every time a new

Hiragana character is typed. In this way, a simple sentence could easily involve multiple candidate prediction tasks, therefore the most crucial problem in using deep-learning technology on mobile devices is often the execution speed. With this restrick, we select our model task on re-ranking candidates after a translation from the Hiragana sequence. In the whole input activity, the user first types the Hiragana sequence. Then the possible candidates show up for the user to choose.

Although the language model would be a visible solution for scoring the candidates with given previous input words, it is difficult to train a stable deep-learning language model with shallow model size. Instead using a simple binary classification to model candidate scores and train a compact size model, but it is also hard to converge a useful model.

Recent NLP models like BERT [3] or GPT [4] are using pre-training from a larger corpus to learn well stated model parameters and then fine-tune on a task with smaller corpus. We leverage this training strategy on our training. But instead of two steps training, we join two training objectives to form a multi-task learning. The intuition behind this setting is that only the binary classification objective is hard to guide the model to learn, but with the language model objective as an auxiliary objective, helps the model to converge better.

We use GPT as our model encoder. GPT does not rely on recurrent mechanisms to follow language modeling objectives, instead it uses masks to achieve this manner. In this way, the calculation can run in parallel, which improves execution speed.

3 Method

3.1 BPE

To start the natural language processing, sentences need to be tokenized. For languages like Japanese or Chinese, there are no natural word boundaries, such as a space character. It is crucial to have a good tokenization methodology to start the whole process. Following GPT, it uses byte pair encoding (BPE) to process English corpus. We use BPE on Japanese texts as a tokenizer. BPE largely decreases out-of-vocabulary words and also reduces sequence length with only around 50k token size. This is important for running deep-learning models on mobile devices. Because in a normal tokenization setting, for a 95% vocabulary coverage, token size easily exceeds 100k. Embedding matrix size often contributes most parameters in the whole model. Hence, reducing embedding size can reduce model size significantly.

The other benefit of BPE is the encoded sequence length. Other than using lots of tokens, using character based tokenization methods is another choice in Japanese and Chinese. In this setting, the token size is often between 5k to 10k to achieve similar word coverage, but this causes the encoded sequence length to be longer and too sparse for the embedding. The BPE balances this between normal tokenizer and character-based tokenizer.

3.2 Two-Stream-Encoded Model

We customize Transformer models to fit the Japanese input method and the re-rank candidates task. In Japanese input scenarios, users type a Hiragana sequence and the sequence is converted into words. This is a one to many mapping. Different words have the same pronunciation and all can be the potential candidates. There are two useful data that helps rank candidates: one is words users have already typed, the other is the Hiragana sequence users currently typing.

We use two GPT models to encode word sequence and Hiragana sequence. The word sequence is composed of history words and one candidate. These two encoded output logits then concatenate into one vector and feed into a binary classifier. The classifier outputs a scalar ranging from zero to one to indicate how likely the candidate fits for current usage.

Our model consists of two blocks, word encoder and reading encoder. We call it the Two-stream-encoded Model (TSE). The word encoder encodes word sequence and the reading encoder encodes Hiragana sequence respectively. Instead of using the identical structure on both encoders, the word encoder has more parameters than reading encoder due to the longer sequence and the larger unique token size. We append a self-attention layer to the final layer of word encoder and use reading encoder output as this self-attention layer's query input. We want the word encoder output to be conditioned on user typing input. The detail model architecture is in Figure 1.

3.3 Auxiliary Training Objectives

We use point-wise approaches to rank candidates, which means we give every candidate a score and use this score to rank. In this setting, a sigmoid function is a good fit for model output. We train models with cross-entropy loss between model output and real label, with user-select candidate label regarded as positive and non user-select candidate

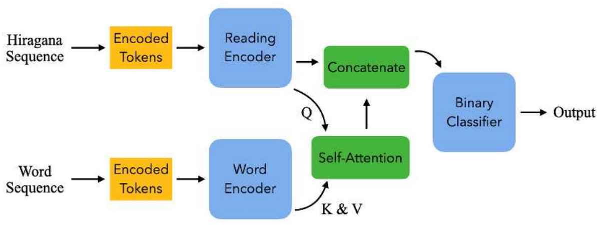

Figure 1: Two-Stream-Encoded model architecture. The word sequence is the concatenation of history words and one candidate. The binary classifier output is a scalar which indicates the score of the candidate.

label as negative. We found that using only this training objective cannot generalize well on the test set. To tackle this problem, auxiliary training objectives, language modeling loss is introduced to two GPT models in training. This helps models encoding better output logits for the classifier. Both GPT encoders have their own language modeling loss and total loss is the sum of three losses as the following Eq. 1. Alpha and beta here are the weights for controlling the language model loss that contribute to total loss.

$$L_t = \alpha * L_{w-encoder} + \beta * L_{r-encoder} + L_{cls} \quad (1)$$

4 Data and Experiment Setting

We train and evaluate the model using a dataset that was collected from volunteers . In this section, we first describe the dataset, and then show the experiment details.

4.1 Data

The dataset contains about 200 million training dataset and 0.3 million testing dataset. Data is collected from volunteers, each input data is associated with 2 to hundreds of candidate words. The words selected by the user are taken as positive samples, and negative samples are randomly selected from the candidate words not included in the words selected by the user.

4.2 Experiment Settings

4.2.1 Vocabulary

In order to get better representations from word encoder and reading encoder, We use BPE to learn subword vocabularies from Hiragana sequence and word sequence separately. There are 50k tokens for word encoder and 8k tokens for reading encoder. The Hiragana sequence has a smaller character space and uses less token size than word encoder.

4.2.2 Training Details

We set the batch size and the train steps to 256 and 2 million, warm-up steps set to 10000. The learning rate is initially set 8e-4 and will be reduced to 8e-6 by learning rate linear decay, Adam is used for optimization and L2 weight decay is set to 5e-3. We also use linear decay to dynamically adjust the alpha and beta values in the Eq. 1, the details are shown in Table 2.

5 Result and Discussion

To evaluate the inference speed, we conduct an inference procedure on the testing dataset with batch size of 9. Results are shown in Table 1 and the average running time of 10000 batches on a single NVIDIA K40 GPU.

We use Top-1 accuracy as the metric to evaluate model performance. Meanwhile, we also compare the model size and the speed of inference, which are the key factors that run smoothly on mobile devices. We investigate the effect of model size and loss weights on the TSE model, the detailed results and discussions are shown below.

5.1 Effect of model size

We take the online model (no rerank model), LSTM, and GPT as the baseline model, and compare them with our TSE model, we find that our TSE model has achieved better performance shown in the Table 1. LSTM and GPT models are trained with language model objectives on history words and user selected candidates. For comparison, we set the parameter size similar to the TSE model. Using language models to evaluate candidates is a straightforward method. But the restriction of model size and lack of user input information cause the accuracy far from our online model. The TSE model is designed to improve this problem. It introduces a reading encoder that can effectively encode the Hiragana sequence and provides the conversion information between Hiragana sequence and candidate words.

We also compare performance with different depths and widths on Top-1 accuracy and inference speed. The TSE-1 model with a wider and deeper model achieves the best performance, with a Top-1 accuracy of 82.5%. Reducing the width of the word encoder by half (TSE-2), the Top-1 accuracy drops to 82.1%, with equivalent model size and inference speed. Reducing the depth of the word encoder and reading encoder by half (TSE-3), the Top-1 accuracy drops to 82.3%, but the model size and inference speed has also reduced by 0.45M and 33s respectively, the memory consump-

	Reading Encoder	Word Encoder	#Param	Inference Time(s)	Top-1 Accuracy
Online	-	-	-	-	78.6%
LSTM	-	n_layer=2; d_lstm=256; d_embed=32	2.42	85.55(1.44x)	69.0%
GPT	-	n_layer=6; d_attn=128; d_ffn=256; d_embed=32	2.78	117.48(1.97x)	68.8%
TSE-1	n_layer=2; d_attn=32; d_ffn=64; d_embed=32	n_layer=4; d_attn=128; d_ffn=256; d_embed=32	2.88	93.15(1.56x)	82.5%
TSE-2	n_layer=2; d_attn=32; d_ffn=64; d_embed=32	n_layer=4; d_attn=64; d_ffn=128; d_embed=32	2.46	93.12(1.55x)	82.1%
TSE-3	n_layer=1; d_attn=64; d_ffn=128; d_embed=32	n_layer=2; d_attn=128; d_ffn=256; d_embed=32	2.43	59.57(1x)	82.3%

Table 1: The model sizes and inference time for baselines and TSE models. (n_layer means the number of the model layer; d_lstm means LSTM hidden size; d_embed means the embedding size; d_attn means the hidden size of the self-multi-attention block in Transformer; d_ffn means the hidden size of FFN sub-layer in Transformer; #Param means the number of parameters.)

System	Initial value of alpha and beta in first 1.5 million	Initial value of alpha and beta in last 0.5 million	Decay Method	Top-1 Accuracy
No LML	0.0	0.0	-	55.6%
No LWD	1.0	1.0	-	80.9%
	1.0	0.0	-	82.0%
LWD	1.0	0.1	Linear	82.1%
	1.0	0.0	Linear	82.3%

Table 2: Ablation studies of loss weights in the TSE-3 Model learning. (No LML means without the auxiliary training objectives; No LWD means alpha and beta in the Eq.1 remain constant; LWD means alpha and beta will decay from the initial value of the first 1.5 million to the last 0.5 million during the first 1.5 million.)

tion when running on the mobile phone is only 12MB. We think that re-ranking candidate words does not require much semantic understanding, but more to capture the usage of words. Therefore, the shallow network can still achieve a good performance, and the speed is faster.

5.2 Effects of loss weights

We investigate the effects of the loss weights (alpha and beta in the Eq. 1) on the TSE-3 model learning. The training period is divided into the first 1.5 million steps and the last 0.5 million steps, the two stages using different language model loss weights respectively. During the training process, the alpha and beta in the last 0.5 million will remain constant, but in the first 1.5 million may be dynamically decayed. Note that we set the alpha and beta to same value during the training period.

Several baselines are proposed including the TSE-3 model without the auxiliary training objectives (Abbreviated as No LML), the loss weights decay (Means alpha and beta in the first 1.5 million remain constant, abbreviated as No LWD) respectively. The results are illustrated in Table 2 and show the best performance using linear decay in the first 1.5 million steps and removing the language model losses in the last 0.5 million steps. The Top-1 accuracy is only 55.6% under the setting (No LML), which is much lower than the baseline. This result proves that auxiliary training objectives are the key for TSE model learning, they can effectively help the model converge.

Furthermore, we investigate the contributions of the auxiliary training objectives in the TSE-3 model learning (No LWD), we set the alpha and beta to 1.0, and the Top-1 accuracy is 80.9%. However, if we remove it in the last 0.5 million training steps, the Top-1 accuracy will increase to 82.0%. This results shows that the effects of the auxiliary training objectives to the model learning should be reduced in the last training period, enabling the classification task dominating the model learning.

Finally, we introduce linear decay in the first 1.5 million training steps to dynamically adjust the loss weights, and the Top-1 accuracy has been further improved slightly. And it is better to remove the language model loss in the last 0.5 million training steps than setting the loss weights to 0.1, which the accuracy improves by 0.2. This may be due to the fact that the model has converged well in the first 1.5 million steps, removing the language model loss in the last 0.5 million steps can ensure the classification tasks are better trained.

6 Conclusion

In this paper, We empirically investigate a Transformer-based model for the re-ranking of the candidate words in input method, finding that it outperforms the online model, Lstm-based model and GPT-based model. Transformer-based models can encode more context compared with the traditional model which is the N-gram model. While getting a better performance, we also reduce the size of the model to ensure that it can run smoothly on mobile devices. Finally, we introduce two auxiliary training objectives to help train the small Transformer-based model, and dynamically adjust the loss weights to obtain better performance.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Åukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[2] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units, 2015.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding, 2018.

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