IIGROUP Submissions for WMT22 Word-Level AutoCompletion Task

Cheng Yang   Siheng Li   Chufan Shi   Yujiu Yang
Tsinghua Shenzhen International Graduate School, Tsinghua University
{yangc21,lisiheng21,scf22}@mails.tsinghua.edu.cn
yang.yujiu@sz.tsinghua.edu.cn

Abstract

This paper presents IIGroup's submission to the WMT22 Word-Level AutoCompletion(WLAC) Shared Task in four language directions. We propose to use a *Generate-then-Rerank* framework to solve this task. More specifically, the generator is used to generate candidate words and recall as many positive candidates as possible. To facilitate the training process of the generator, we propose a span-level mask prediction task. Once we get the candidate words, we take the top-K candidates and feed them into the reranker. The reranker is used to select the most confident candidate. The experimental results in four language directions demonstrate the effectiveness of our systems. Our systems achieve competitive performance ranking 1^{st} in English to Chinese subtask and 2^{nd} in Chinese to English subtask.

1 Introduction

Recent advances in neural machine translation (Bahdanau et al., 2015; Vaswani et al., 2017) allow us to generate high-quality translation results. However, as it's pointed out by Li et al. (2021) that in some scenarios(e.g., legal instruments), the results of machine translation can't directly replace human translations due to their imperfections(e.g., terminology translation error). Therefore, more and more researchers pay attention to *Computer-aided translation*(CAT)(Barrachina et al., 2009; Santy et al., 2019; Huang et al., 2021; Xiao et al., 2022), which focuses on leveraging the advantages of NMT systems to increase the effectiveness and efficiency of the human translation process.

To further promote the development of CAT, WMT22 proposes a novel task —— Word-Level AutoCompletion(WLAC). In the Word-Level AutoCompletion task, given a source sentence $\boldsymbol{x}$, target context and human-typed characters $\boldsymbol{t}$, an ideal system is expected to be able to predict the target word w that should be placed in the target context.

We participate in the WMT22 shared Word-Level AutoCompletion task in four language directions: Chinese ⇒ English, English ⇒ Chinese, German ⇒ English and English ⇒ German and submit a system for each language direction.

We develop a *Generate-then-Rerank* framework-based system for each language direction. Based on the vanilla Transformer architecture, we adopt a bidirectional decoder, which can predict the current target word by paying attention to the source sentence and both the left-side and right-side target context.

The paper is organized as follows, section 2 gives the overview of the data used in the shared task and preprocessing operations for the data, while section 3 describes our training techniques, including model architecture, span-level mask prediction, etc. Section 4 presents our experimental results. Finally, our conclusions are summarized in Section 5.

2 Data

In this section, we first introduce the datasets used to train our systems, then we introduce how to prepare the simulated training data for the WLAC shard task and describe the vocabulary for each language direction.

2.1 Datasets

As the WLAC shared task is a data-constrained task, we can only use the parallel corpora provided by the WLAC organizers for all four language directions. Specifically, we use UN Parallel Corpus V1.0[1] (WMT 2017) for Chinese ⇒ English and English ⇒ Chinese. For German ⇒ English and English ⇒ German, we use the WMT 14 dataset pre-processed by Stanford NLP Group[2]. Details of the training resources provided are shown in Table 1.

[1] https://conferences.unite.un.org/uncorpus

[2] https://nlp.stanford.edu/projects/nmt

Proceedings of the Seventh Conference on Machine Translation (WMT), pages 1187 - 1191
December 7-8, 2022 ©2022 Association for Computational Linguistics

	Zh-En	De-En
Train Set	10M	4.5M
Validation Set	3k	3k

Table 1: The detailed statistics of training and validation data used in our system.

	Zh⇒En	En⇒Zh	De⇒En	En⇒De
source	60k	50k	50k	50k
target	50k	60k	50k	50k

Table 2: The vocabulary size of different language directions.

2.2 Simulated Training Data

Since the WLAC shared task only provides raw parallel corpora and does not provide supervised data, which complies with the WLAC shared task setting, we need to automatically construct supervised data from the raw parallel corpora.

Specifically, given a raw parallel sentence pair $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x} = (x_1, ..., x_m)$ is the source sentence, $\boldsymbol{y} = (y_1, ..., y_n)$ is the reference target sentence, we would like to construct a target word w and its corresponding target context $\boldsymbol{c} = (\boldsymbol{c}_l, \boldsymbol{c}_r)$ and human-typed characters $\boldsymbol{t}$, where the translation pieces $\boldsymbol{c}_l$ and $\boldsymbol{c}_r$ are on the left and right side of the target word w.

According to Li et al. (2021), we first randomly sample a target word $w = \boldsymbol{y}_t$, and then we sample four types of context types:

- Zero-context: both $\boldsymbol{c}_l$ and $\boldsymbol{c}_r$ are empty;
- Prefix: randomly sample a translation piece $\boldsymbol{c}_l = \boldsymbol{y}_{i:j}$ from $\boldsymbol{y}$, where $i < j < t$. The $\boldsymbol{c}_r$ is empty.
- Suffix: randomly sample a translation piece $\boldsymbol{c}_r = \boldsymbol{y}_{i:j}$ from $\boldsymbol{y}$, where $t < i < j$. The $\boldsymbol{c}_l$ is empty.
- Bi-context: sample $\boldsymbol{c}_l$ as in prefix, and sample $\boldsymbol{c}_r$ as in suffix.

Last but not least, we need to generate human-typed characters $\boldsymbol{t}$ for the target word w, we adopt a heuristic method - we randomly sample a position i in the target word w, where $0 < i < |w|$, and simulate human-typed characters $\boldsymbol{t} = w_{1:i}$. For languages like Chinese, the human input is the phonetic symbols of the word, we use pypinyin[3] to implement this conversion. So far, we get the tuple $(\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{t}, w)$, which can be viewed as a simulated training example for the WLAC shared task.

2.3 Vocabulary

Considering that WLAC is a word-level task, we don't use tools to do any subword segmentation. We directly use Moses scripts[4] to tokenize English and German sentences, and jieba[5] for Chinese sentences. The vocabulary size for each language direction is shown in Table 2.

3 Word-Level AutoCompletion Systems

In this section, we mainly introduce the *Generate-then-Rerank* framework. Both the generator and the reranker's architecture are based on Transformer(Vaswani et al., 2017) with the modification that the decoder is bi-directional to leverage more context information. It is important to note that we borrow the idea from Li et al. (2021) that we view WLAC as a word prediction task and *only use human-typed characters* $\boldsymbol{t}$ *as hard constraints*.

3.1 Model Architecture: Transformer

The vanilla Transformer (Vaswani et al., 2017) adopts a sequence-to-sequence architecture consisting of an encoder and a decoder. Specifically, the encoder is a stack of L encoder blocks and each block consists of a multi-head self-attention module and a feed-forward network (FFN). The decoder is also a stack of L decoder blocks, the main differences between the Transformer encoder and Transformer decoder are mainly reflected in two aspects: First, in each decoder block, there is an additional cross-attention module between the multi-head self-attention module and the feed-forward network. Second, the multi-head self-attention modules in the decoder are uni-directional while they are bi-directional in the encoder.

In the neural machine translation task setting, given a source sentence $\boldsymbol{x}$ and a target sentence $\boldsymbol{y}$, the decoder generates $\boldsymbol{y}$ as:

$$P(\boldsymbol{y}|\boldsymbol{x};\theta) = \prod_{t=1}^{|\boldsymbol{y}|} P(y_t|\boldsymbol{y}_{<t}, \boldsymbol{x};\theta) \qquad (1)$$

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

[4] https://github.com/moses-smt/mosesdecoder

[5] https://github.com/fxsjy/jieba

Thus, the Transformer model is typically trained by minimizing the cross entropy:

$$\mathcal{L}_{NMT} = -\sum_{t=1}^{|\boldsymbol{y}|} \log P(y_t | \boldsymbol{y}_{<t}, \boldsymbol{x}; \theta) \qquad (2)$$

Since Transformer is designed for auto-regressive generation tasks, we cannot directly adopt it to the WLAC task, which is essentially a natural language understanding task. Inspired by the successful practice of Conditional Masked Language Modeling (Ghazvininejad et al., 2019) in non-autoregressive machine translation, we take the same idea to train our model for the WLAC shared task.

Bi-directional Decoder Our decoder's architecture is roughly the same as the standard Transformer decoder except that the multi-head self-attention sub-layer. The standard Transformer decoder can only attend the left-side target context, while in our model, it can attend to all target words and make use of both left-side and right-side context information to better predict the <*mask*> token.

3.2 Generator

Span-Level Mask Prediction The primitive object function for a simulated training example in Generator is as follows:

$$\mathcal{L}_G = -\log P(w|\boldsymbol{x}, \boldsymbol{c}; \theta_G) \qquad (3)$$

In our preliminary experiments, we find that it is hard to train the generator because, in every mini-batch, a simulated training example provides *only one* training signal, which makes the model easy to overfit. The importance of the density of training signals has been discussed in the Pretrained Language Model(Clark et al., 2020). To this end, we adopt an efficient sampling approach —— Span-Level Mask Prediction. As described in section 2.2, once we get the tuple $(\boldsymbol{x}, \boldsymbol{c})$, we use it to predict all the missing words in the masked span between $\boldsymbol{c}_l$ and $\boldsymbol{c}_r$. In the Pretrained Language Model, Joshi et al. (2020) has adopted the same idea as in our work. But one major difference is that, unlike Joshi et al. (2020), we have to set the position id of the masked word to be the same; otherwise, there will be a large gap between the training stage and the inference stage.

3.3 Reranker

So far, we have modeled the WLAC task as a classification task, that is, an extreme classification task. Inspired by recent works to introduce label knowledge to enhance text representation (Yang et al., 2021; Ma et al., 2022), we propose to use a generator-reranker framework to solve the WLAC task. We use the generator to recall positive and negative labels and use a reranker to distinguish positive labels from these labels. Specifically, we use the same Transformer architecture as the generator. But the reranker's input and objective function are different from the generator.

Input We obtain top-K labels $\mathcal{W} = \{w_1, w_2, ..., w_K\}$ through ranking the scores generated by the generator. Then, for each candidate label w_i in $\mathcal{W}$, we replace the <*mask*> token with w_i. So the input tuple becomes $(\boldsymbol{x}, \boldsymbol{c}, w_i)$. And the multi-class classification head of the original decoder becomes a binary classification head, which is used to measure whether the candidate label w_i matches the source sentence and target context.

Objective Function The objective function is as follows.

$$\mathcal{L}_R = \begin{cases} -\log P(w_i, \boldsymbol{x}, \boldsymbol{c}; \theta_R), & \text{if } w_i = w \\ -(1 - \log P(w_i, \boldsymbol{x}, \boldsymbol{c}; \theta_R)), & \text{otherwise.} \end{cases} \qquad (4)$$

3.4 Model Configuration

We implement our models with Fairseq toolkit(Ott et al., 2019)[6]. Our models follow the Transformer-Base architecture(Vaswani et al., 2017), the key model architecture configurations and training configurations are listed in Table 4 and Table 5. Each model is trained on 8 NVIDIA Tesla V100 GPUs, each of which has 32GB memory.

4 Experimental Results

We report experimental results in four language directions: Chinese ⇒ English, English ⇒ Chinese, German ⇒ English and English ⇒ German. Table 3 shows the main experimental results on the official test sets with automatic accuracy evaluation and human accuracy evaluation.

[6] https://github.com/facebookresearch/fairseq

#	Systems	Zh⇒En		En⇒Zh		De⇒En		En⇒De	
		Auto	Human	Auto	Human	Auto	Human	Auto	Human
1	Generator	54.05	85.00	53.98	83.25	57.27	78.75	41.82	55.50
2	Reranker	51.11	83.75	48.90	77.50	54.32	76.25	40.69	53.50

Table 3: The main results of different systems in four language directions. The results are the averaged automatic accuracy and human accuracy on four types of translation context (i.e., zero context, prefix, suffix, and bi-context).

Configuration Name	Configuration Value
encoder layers	6
decoder layers	6
attention heads	8
word embedding dim	512
FFN embedding dim	2048
hidden dim	512
dropout	0.1
attention dropout	0.0
activation droupout	0.0
Pre-LN	False
share decoder input output embed	True

Table 4: The exact specifications of the Transformer we adopt.

The performance of the generator is as expected, and as demonstrated in Li et al. (2021), without using the bi-directional decoder, the generator performs relatively poorly. Additionally, we conduct an ablation study on Chinese ⇒ English subtask to demonstrate the effectiveness of the span-level mask prediction, the model without leveraging the span-level mask prediction strategy performs poorly, with a drop of -10.1 in accuracy on the validation set.

However, the performance of the reranker is not as expected. We conjecture that this is due to the insufficiency of the training procedure of the reranker. Initializing reranker's weights with generator's weights or with PLM's weight will boost the performance of reranker, we leave this as future work.

5 Conclusion

This paper describes the IIGROUP's systems submitted to the Word-Level AutoCompletion task at WMT22. We adopt a *Generate-then-Rerank* framework. The experimental results demonstrate the effectiveness of the generator.

However, due to the lack of computing power and time, the results of our experiments don't show the effectiveness of our reranker. We discuss this issue in section 4 and we will try to solve this in future work.

Configuration Name	Configuration Value
number of training steps	10000
update freq	1
learning rate scheduler	inverse sqrt
warmup updates	4000
warmup init learning rate	1e-7
learning rate (generator)	5e-3
learning rate (reranker)	1e-3
max tokens per batch	32k
optimizer	Adam

Table 5: Training configuration for our generator model and reranker model.

References

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, et al. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121.

Guoping Huang, Lemao Liu, Xing Wang, Longyue Wang, Huayang Li, Zhaopeng Tu, Chengyan Huang, and Shuming Shi. 2021. Transmart: A practical interactive machine translation system. *arXiv preprint arXiv:2105.13072*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Huayang Li, Lemao Liu, Guoping Huang, and Shuming Shi. 2021. Gwlan: General word-level autocompletion for computer-aided translation. 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 4792–4802.

Jie Ma, Miguel Ballesteros, Srikanth Doss, Rishita Anubhai, Sunil Mallya, Yaser Al-Onaizan, and Dan Roth. 2022. Label semantics for few shot named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1956–1971.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Sebastin Santy, Sandipan Dandapat, Monojit Choudhury, and Kalika Bali. 2019. Inmt: Interactive neural machine translation prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 103–108.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yanling Xiao, Lemao Liu, Guoping Huang, Qu Cui, Shujian Huang, Shuming Shi, and Jiajun Chen. 2022. Bitiimt: A bilingual text-infilling method for interactive machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1958–1969.

Pan Yang, Xin Cong, Zhenyu Sun, and Xingwu Liu. 2021. Enhanced language representation with label knowledge for span extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4623–4635.