# Transn's Submissions to the WMT22 Translation Suggestion Task

**Hongbao Mao** **Wenbo Zhang** **Jie Cai** **Jianwei Cheng**
Transn IOL Research, Wuhan, China
{hubben.mao, albert01.zhang, jay.cai, nevil.cheng}@transn.com

## Abstract

This paper describes the Transn's submissions to the WMT2022 shared task on Translation Suggestion. Our team participated on two tasks: Naive Translation Suggestion and Translation Suggestion with Hints, focusing on two language directions Zh→En and En→Zh. Apart from the golden training data provided by the shared task, we utilized synthetic corpus to fine-tune on DeltaLM (ΔLM), which is a pre-trained encoder-decoder language model. We applied two-stage training strategy on ΔLM and several effective methods to generate synthetic corpus, which contribute a lot to the results. According to the official evaluation results in terms of BLEU scores, our submissions in Naive Translation Suggestion En→Zh and Translation Suggestion with Hints (both Zh→En and En→Zh) ranked 1st, and Naive Translation Suggestion Zh→En also achieved comparable result to the best score.

## 1 Introduction

Combining machine translation (MT) and human translation (HT) is becoming a popular way in translation practice, which uses a typical way of post edit (PE) – the human translators are asked to provide alternatives for the incorrect word spans in the results generated by MT (Green et al., 2013; Bahdanau et al., 2015; Vaswani et al., 2017; Zouhar et al., 2021; Yang et al., 2021). In order to improve the efficiency of PE, researchers proposed translation suggestion (TS) to provide the sub-segment suggestions for the annotated incorrect word spans, and experiments show that TS can substantially reduce translators' cognitive loads and the post-editing time (Wang et al., 2020; Lee et al., 2021; Yang et al., 2021).

This paper describes the contribution of Transn IOL Research to the WMT22 Translation Suggestion shared task, where systems were submitted to two tasks: 1) Naive Translation Suggestion; 2) Translation Suggestion with Hints. For both tasks we trained the models on pre-trained encoder-decoder language model ΔLM (Ma et al., 2021) with the corpus which were synthesized deliberately, then submitted the ensemble results of the trained models. Our main contributions are:

. We utilized the pre-trained language model ΔLM to generate TS, which gets good results on the shared tasks, and much lower computational budget than training from raw Transformers (Vaswani et al., 2017; Junczys-Dowmunt and Grundkiewicz, 2018; Yang et al., 2021) as well as better quality.

. Apart from the provided golden data annotated by expert translators, we proposed the constructing methods for silver and bronze data to train TS system based on parallel corpus and the NMT models provided by the shared tasks, which contributes a lot for the final results.

. Based on the Naive Translation Suggestion models, we proposed an effective algorithm for the task of Translation Suggestion with Hints, which improves BLEU scores significantly.

The rest of this paper is organized as below. Section 2 is a brief description for Translation Suggestion shared task of WMT2022. Section 3 presents our system, including data constructing and the training process with ΔLM. Section 4 reports experimental results in the participated language directions. Finally, we conclude our work in Section 5.

## 2 Translation Suggestion Tasks

Translation Suggestion is a new task on WMT2022, which includes two sub-tasks.

**Task 1 - Naive Translation Suggestion**: This sub-task focuses on the scenario where the user



*Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1205–1210
Abu Dhabi, December 7–8, 2022. 

selects the incorrect span of the MT sentence without entering any information, the model outputs the alternatives automatically. Consider the source sentence $x$, the MT sentence $m$, the incorrect span selected by the user as $w$, the alternative $y$ and the model parameter $\theta$, the naive TS can be formulated as: $P(y|x, m, w, \theta)$.

**Task2 - Translation Suggestion with Hints**: In actual applications, users usually have general ideas of what they want. If they are dissatisfied with all the suggestions provided by Naive TS, they are willing to enter some hints for the model to generate more accurate suggestions. Given the hints $h$ provided by users, the sub-task 2 can be formulated as: $P(y|x, m, w, h, \theta)$. In this task, we take the $top-k$ initial characters of the alternative words as the hint, and the $k$ is randomly selected for each example.

Task 1 includes 4 language directions (En⇔Zh and En⇔De) and Task 2 includes 2 language directions (En⇔Zh). We participated En⇔Zh language directions for both tasks.

## 3 Implemented Systems

We fine-tune the pre-trained language model ΔLM on synthetic data and the task golden data for Task1, then adjust N-best parameter along with an optimization algorithm for Task2. The details are described in this section.

### 3.1 Pre-trained Model

ΔLM is a pre-trained multilingual encoder-decoder model, which outperforms various strong baselines on both natural language generation and translation tasks (Ma et al., 2021). Its encoder and decoder are initialized with the pre-trained multilingual encoder InfoXLM (Chi et al., 2020), and trained in a self-supervised way. ΔLM's pre-training tasks include span corruption on monolingual data and translation span corruption on bilingual data. We choose ΔLM as the pre-trained model for TS task because the pre-training task of translation span corruption is similar to TS. The only difference is that ΔLM masks spans in target sentence as well as spans in source sentence on bilingual data, which follows the idea from mT6 (Chi et al., 2021), but TS only masks one span in target sentence.

We use ΔLM-base model in our experiments, which has 360M parameters, 12-6 encoder-decoder layers, 768 hidden size, 12 attention heads and 3072 FFN dimension.

### 3.2 Construct Synthetic Data

The golden data provided by the TS tasks are annotated by expert translators, which are expensive and labor-consuming. Since the 15k golden data are far from enough to fine-tune a ΔLM model, we propose several methods to construct synthetic data for TS on parallel corpus and the specified NMT models. These synthetic data are named as silver or bronze data according to its constructing complexity as well as effect contribution.

**Silver Data** Silver data are constructed on parallel corpus and additional models or tools. We implemented two kinds of silver data construction.

1) The data are obtained via difference comparison on MT and target sentences. Given a parallel corpus sentence pair of source and target sentence, we first translate the source sentence by the NMT model (which is used to generate the train/dev/test data of the TS task and released to all task participants), then compute edit distance (ED) between MT and target sentence to measure the cost of editing from MT to target. We choose ED metric of LCS (Longest Common Subsequence) (Bergroth et al., 2000), which means only insertion and deletion operations are allowed (not substitution operation). ED is usually calculated by dynamic programming, and it can indicate the words which are inserted or deleted from MT to target sentence by a trace-back approach. So we can get a TS span by concatenating all words between the first and last edited words in target sentence. Table 1 shows an example for it. This can be formulated as:

$$TS = diff(NMT(source), target) \quad (1)$$

Thus, $(source, target_{diff\_mask}, TS)$ is the constructed train data, and $target_{diff\_mask}$ is the masked translation where the TS span is replaced with a placeholder. If the edited parts in target sentence is too long, it will induce a long TS span and short masked translation, so we filter out such data by a threshold.

2) The data are constructed by masking special parts on target sentences. By browsing the golden TS train data, we found there were certain regularity. Apart from the haphazard TS spans, NEs (Named Entity) and non-translated elements (especially digits) inclined to be mistranslated. So we can focus on constructing synthetic data by masking and predicting NEs and non-translated elements in target sentences. We use spaCy[1] NER function

[1] https://spacy.io/

| | |
|---|---|
| $MT$ | 4.6.1 Suspension for Contractor reasons |
| $target$ | 4.6.1 Suspension because of Contractor reasons |
| $difference$ | 4.6.1 Suspension <add>because</add> <del>for</del><add>of</add> Contractor reasons |
| $TS$ | because of |
| $target_{diff\_mask}$ | 4.6.1 Suspension <mask> Contractor reasons |

Table 1: An example of synthetic data by difference comparison on MT and target sentences.

to extract such spans in target sentences, and select NE labels of PERSON, LOC, ORG, PRODUCT, MONEY and QUANTITY. This can be formulated as:

$$TS = NER(target) \quad (2)$$

Thus, $(source, target_{NER\_mask}, TS)$ is the constructed train data, and $target_{NER\_mask}$ is the masked translation where the TS span is replaced with a placeholder.

**Bronze Data** Bronze data are sampled directly on parallel corpus. Sampling on parallel corpus is straightforward and simple but effective for TS model. This method is also used by (Yang et al., 2021). Given the sentence pair (source,target) in the parallel corpus, we denote $target^{\setminus i:j}$ as a masked version of target sentence where its fragment from position $i$ to $j$ is replaced with a placeholder $(1 \leq i \leq j \leq |target|)$. The $target^{i:j}$ denotes the fragment of target from position $i$ to $j$. We treat $target^{i:j}$ and $target^{\setminus i:j}$ as the TS and masked translation respectively. This can be formulated as:

$$TS = target^{i:j} \quad (3)$$

So we get the constructed train data $(source, target^{\setminus i:j}, TS)$.

When the target language is Chinese, we tokenize the target sentence by Jieba[2] before sampling on it.

### 3.3 Training Process

We perform two-stage fine-tuning on ΔLM for training the TS models. In the first stage, we use the silver and bronze train data to fine-turn on the original ΔLM model. In the second stage, we continue to fine-tune on the results of the first stage with the golden train data. Because there are much more train data and time consumption in the first stage than that of the second stage, we just train one model for stage 1, but train several models for stage 2 with different parameters considering the plan of model ensemble. The details will be described in Section 4.

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

### 3.4 Optimization Algorithm for TS Candidates with Hints

For Task2, we use the same models as that of Task1 to generate TS candidates, and the minor adjustment is just generating more outputs with a larger N-best value during predicting. Given TS candidates by the initial predicting order, our optimization algorithm is simple and effective. Firstly, each TS candidates is converted to a string consisting of the first character of the words in TS, and secondly, we compute LCS (Bergroth et al., 2000) between each string and the hint by the candidates order, then choose the longest LCS from the results. If there are multiple longest LCSs, just choose the first one by the candidates order. Finally, the TS candidate corresponding to the longest LCS is selected as the best TS.

For Chinese language, first of all the TS candidates should be converted to phonetic symbols word by word, then perform the above process. We use pypinyin[3] to get phonetic symbols of Chinese words.

## 4 Experiments

We present the performance of the implemented models on the dev and test datasets, as well as some additional analysis.

### 4.1 Data Used

In addition to the golden train and dev data provided by the TS tasks, other data we used to train TS models are from WMT22 general translation task[4], and just part of the bilingual data are used.

**Data Used for Zh→En Direction** The original parallel corpus for generating synthetic data are 14 million ParaCrawl v9 Zh⇔En and 15 million UN Parallel Corpus V1.0 Zh⇔En bilingual data. Following the data constructing methods in section 3.2, all of the constructed silver and bronze data are 110 million. We sampled 4 times on different posi-

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

[4] https://www.statmt.org/wmt22/translation-task.html

| Dataset | Stage | Zh→En | En→Zh |
|---|---|---|---|
| dev set | Stage 1 | 15.62 | 25.10 |
| | Stage 2 | 28.15 | 38.08 |
| test set | Stage 2 | 28.42 | 39.71 |

Table 2: BLEU of two stages on dev or test sets for Zh⇔En language directions

tions for every sentence when constructing bronze data.

**Data Used for En→Zh Direction** Besides the original parallel corpus of Zh→En direction, we added 6 million CCMT corpus. The data constructing methods are the same as Zh→En direction, and we finally got 120 million silver and bronze data.

### 4.2 Results of Task 1

Following the two-stage fine-tuning process described in section 3.3, Table 2 summarizes the results of Task 1 for Zh⇔En language directions.

On stage 1, we fine-tune ΔLM-base with the constructed silver and bronze data. All models are implemented on top of the open source toolkit Fairseq[5]. We train on 6 GeForce RTX 3090 GPUs. The optimizer is Adam (Kingma and Ba, 2014) with $\beta 1 = 0.9$ and $\beta 2 = 0.98$. The learning rate is 6e-5 with a warming-up step of 8000. The models are trained with the label smoothing cross-entropy, and the smoothing ratio is 0.1. All the dropout probabilities are set to 0.3. The gradient accumulation is used due to the high GPU memory consumption, and we set max-tokens = 1600 and update-freq = 64. To speed up the training process, we conduct training with half precision floating point (FP16). We validate on dev set every 1000 updates, and the early stop patience is 5. Under these training parameters, the model converges at epoch 3.

On stage 2, we use the golden train and dev data provided by the TS tasks, and continue to fine-tune on the checkpoint with the best validation performance of stage 1. Only a few training parameters were adjusted on stage 1. The learning rate is reduced to 3e-5. In order to apply model ensemble strategies, the dropout varies in [0.1, 0.2, 0.3], and the update-freq varies in [3, 4, 5] with the fixed max-tokens 1600. The submissions are ensemble results of all models trained on stage 2.

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

| Dataset & parameter | Zh→En | En→Zh |
|---|---|---|
| test set, N-best=100 | 39.95 | 48.60 |

Table 3: BLEU of test set with hints when N-best=100

### 4.3 Results of Task 2

Task 2 intends to predict more accurate translation suggestions under additional hints. So we enlarge the N-best value gradually from 5 to 100 to generate more TS candidates, and search the optimal TS by the algorithm described in section 3.4. Figure 1 shows the results on dev set where the N-best value is set in [5, 10, 20, 30, 50, 80, 100]. It seems that the BLEU rises with the increase of N-best value, but the gains diminish when N-best exceeds 50.

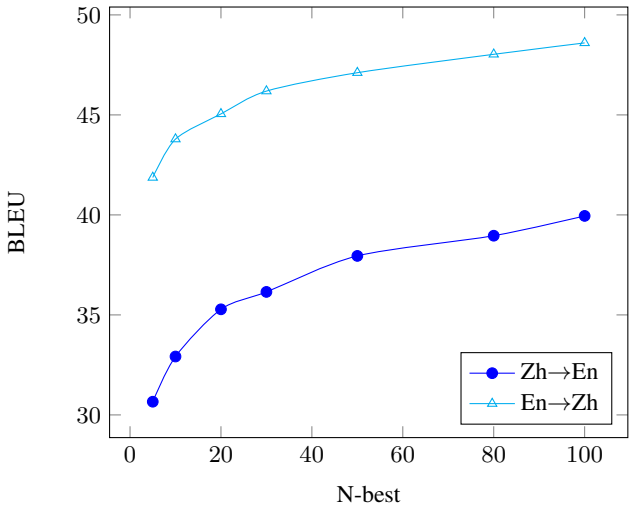


Figure 1: BLEU of dev set with hints for different N-best values

We get the final submissions on test set under N-best = 100, and the official BLEU scores are shown in Table 3 for Zh→En and En→Zh language directions.

### 4.4 Results Analysis Considering TS Accuracy

In practice, TS is designed to replace the incorrect span in target sentence during PE, so an absolutely accurate TS is important for post-editing translators. Therefore we analyze the accuracy indicator for TS in this section. Predicting an absolutely accurate TS relies heavily on TS length, then we analyze it based on different TS lengths, as well as top-k predictions, considering that instead of the top predicted TS, an accurate but top-k-located TS is also valuable for PE through the interactive options. Here top-k predictions are generated in the same

| TS Len | | =1 | ≤3 | ≤5 | ≤10 | All |
|---|---|---|---|---|---|---|
| dev TS, Num=2767 (Proportion) | | 1279 (46.2%) | 2181 (78.8%) | 2488 (89.9%) | 2709 (97.9%) | 2767 (100%) |
| Top-1 predictions | Positive Num | 657 | 900 | 931 | 949 | 950 |
| | **Accuracy** | **51.4%** | **41.3%** | **37.4%** | **35.0%** | **34.3%** |
| Top-3 predictions | Positive Num | 806 | 1159 | 1198 | 1217 | 1217 |
| | **Accuracy** | **63.0%** | **53.1%** | **48.2%** | **44.9%** | **44.0%** |
| Top-5 predictions | Positive Num | 943 | 1379 | 1434 | 1458 | 1458 |
| | **Accuracy** | **73.7%** | **63.2%** | **57.6%** | **53.8%** | **52.7%** |

Table 4: TS accuracy analysis on dev set for Zh→En language direction

way as the TS candidates in Section 3.4.

For Zh→En language direction, Table 4 shows that if there is just one word in TS, the accuracy is 51.4% for the top predictions; and the accuracy reaches 63.0% or 73.7% if we consider top-3 or top-5 predictions. Similarly, the accuracy is 41.3%, 53.1%, or 63.2% if considering top-1, top-3, or top-5 predictions respectively for the TSs no more than 3 words. The accuracy decreases gradually as the TS length increases. A significant finding is that even on the whole dev set, the accuracy still reaches 52.7% if we consider the top-5 predictions. Therefore, the accuracy indicator may help us determine when and how the TS options are activated.

### 4.5 Effects of Training Procedure and Synthetic Data

The two-stage fine-tuning procedure is essential for our results. If stage 1 is not applied, which means just fine-tuning ΔLM on the golden data, we get very low BLEU scores, i.e., 2.19 in Zh→En language direction on dev set. If stage 2 is not applied and just fine-tuning ΔLM on the synthetic silver and bronze data, the BLEU scores are 15.62 in Zh→En and 25.10 in En→Zh (see Table 2), with a decrease of about 13 BLEU score than the full two-stage fine-tuning procedure.

The effects of the synthetic silver and bronze data are also analyzed. Table 5 lists the results in Zh→En language direction for the single silver or bronze data on stage 1 and stage2. It shows that the silver synthetic data plays a more important role for the final performance than the bronze data.

| Systems | Zh→En | |
|---|---|---|
| | **Stage 1** | **Stage 2** |
| silver & bronze data | 15.62 | 28.15 |
| only silver data | 13.53 | 26.11 |
| only bronze data | 10.24 | 24.06 |

Table 5: Effects of the synthetic silver and bronze data for Zh→En language direction on dev set

## 5 Conclusions

We present the Transn IOL Research submissions of the WMT2022 shared task on Translation Suggestion. Our system is implemented with two-stage fine-tuning on ΔLM, which is a pre-trained encoder-decoder language model. To improve the performance, we construct synthetic data by difference comparison, named-entity masking and random sampling on parallel corpus. We propose an effective algorithm to choose optimal translation suggestion with hints. The accuracy indicator of TS is also analyzed for more efficient PE in practice. On the participated 4 tracks of En⇔Zh language directions, we achieved best scores on 3 tracks and comparable result on another track.

Effective translation suggestions benefit a lot for post editing. In the future, we plan to research field related and fine-grained TS models to improve system performance, and will integrate these advanced techniques in our translation practice.

## References

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

L. Bergroth, H. Hakonen, and T. Raita. 2000. A survey of longest common subsequence algorithms. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, pages 39–48.

Zewen Chi, Li Dong, Shuming Ma, Shaohan Huang,

Xian-Ling Mao, Heyan Huang, and Furu Wei. 2021. mt6: Multilingual pretrained text-to-text transformer with translation pairs. *CoRR*, abs/2104.08692.

Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2020. Infoxlm: An information-theoretic framework for cross-lingual language model pre-training. *CoRR*, abs/2007.07834.

Spence Green, Jeffrey Heer, and Christopher D. Manning. 2013. The efficacy of human post-editing for language translation. In *ACM Human Factors in Computing Systems (CHI)*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2018. Ms-uedin submission to the WMT2018 APE shared task: Dual-source transformer for automatic post-editing. *CoRR*, abs/1809.00188.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Dongjun Lee, Junhyeong Ahn, Heesoo Park, and Jaemin Jo. 2021. Intellicat: Intelligent machine translation post-editing with quality estimation and translation suggestion. *CoRR*, abs/2105.12172.

Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, Alexandre Muzio, Saksham Singhal, Hany Hassan Awadalla, Xia Song, and Furu Wei. 2021. Deltalm: Encoder-decoder pre-training for language generation and translation by augmenting pretrained multilingual encoders. *CoRR*, abs/2106.13736.

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.

Qian Wang, Jiajun Zhang, Lemao Liu, Guoping Huang, and Chengqing Zong. 2020. Touch editing: A flexible one-time interaction approach for translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 1–11, Suzhou, China. Association for Computational Linguistics.

Zhen Yang, Yingxue Zhang, Ernan Li, Fandong Meng, and Jie Zhou. 2021. Wets: A benchmark for translation suggestion. *arXiv preprint arXiv:2110.05151*.

Vilém Zouhar, Martin Popel, Ondřej Bojar, and Aleš Tamchyna. 2021. Neural machine translation quality and post-editing performance. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10204–10214, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.