NLP Final Project: Neural Machine Translation with Seq2Seq Modeling

Jingyi Huang (5122FG14): Introduction, Related work, Overall Modeling
Xin Fan(5122F073): Experiment, Results and Discussions, Future work
Aiwen Su(5122F059): Proposed methods, Results and Discussions

July 31, 2022

1 Introduction

Machine Translation (MT) is a NLP task that translates from one language to another, where a semantic understanding of human language is involved. Instead of conventional methods like Statistical Machine Translation (SMT), we focus on Neural Machine Translation (NMT) emerged under the development of deep learning, which relies little on hand-crafted features and has deeper semantic meanings.

2 Related work

Recent techniques have brought about the tendency of performing MT using deep Neural Networks, since it have achieved promising results. Recurrent Neural Network (RNN)-based NMT [1] was first proposed since human languages have auto-regressive properties, while it has a drawback of deteriorating performance for long sentences due to gradient vanishing. The problem can be alleviated by Long short-term memory (LSTM) or Gated Recurrent Unit (GRU) where gates are served to lead a longer memory. Some also tried Convolutional Neural Network (CNN)-based methods to increase the training speed, since Convolutions are allowed to be computed in parallel. While the intrinsic characteristic of Convolutions indicates a local pattern, and global dependencies can only be seen in high-level layers, which prevents a very good result. The above mentioned approaches are later combined with the attention mechanism [2] to give a flexible length of representation (rather than compress the whole sentence into a fix-length vector). Also, attention mechanism can extract global dependencies and be benefit from parallel computation. Besides, the original mode of RNN looks from left to right (i.e. unidirectional). This can be problematic especially for tasks like reading comprehension which require information from both before and after contents. As such, bidirectional RNN [3] was proposed, where a representation on the other direction providing extra information is concatenated. This technique is proved to largely promote the result. However, with the seminal work of Transformer [4], which dispenses of the aforementioned RNN/CNN structure and relies only on the attention mechanism, has brought NLP into a new world. It is also shown to be successful in MT tasks. By adopting the encoder of Transformer, BERT opened up the mode for pre-training and fine-tuning. While for translation tasks, only monolingual pre-training may not be able to fully exploit its potential, thus multilingual models can be considered. For example, mBart [10] is the first model pre-trained on multilingual NLG tasks, which achieves state-of-the-art result on translation tasks. Since NMT is a generation task, the decoding process is of great importance. It is proved that the reversed direction in bidirectional decoding can be served as a complement to improve the results. Non-autoregressive NMT is also proposed to lower the latency in the inference stage. Apart from that, some also do prior knowledge integration including linguistic knowledge, lexical knowledge and syntactic structure, which requires implementing dependency parsing.

3 Overall Modeling

Formulation

MT is a sequence-to-sequence (seq2seq) task [1], with both input and output are sequence. Its objective is to find the most probable sequence given the input. Different from the Classification tasks, its modeling serves an

Encoder-Decoder structure, which gives the representation for the source tokens and the generation for target tokens respectively. In general, seq2seq tasks can be modeled as Conditional Language Models, which is given as

$$\mathbf{y}^* = \underset{\mathbf{y}}{argmax}\ p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) \tag{1}$$

$$= \underset{y_t}{argmax} \prod_{t=1}^{n} p(y_t|y_{<t}, \mathbf{x}) \tag{2}$$

where $\mathbf{x}$ and $\mathbf{y}$ are source tokens and target tokens respectively; y_t is the target token at the step, $y_{<t}$ are target tokens at previous steps; n is the length of the sentence.

Encoder-Decoder structure

Since MT is a generation task, it needs both an encoder and a decoder into its modeling structure. The source tokens are input into the encoder, which encode the semantic meaning of the words; And the target tokens generated by the previous steps are input into the decoder, which did the reverse work of converting the hidden representations into real words; The encoder and the decoder are connected. The principle of the Encoder-Decoder structure lies in that source language and target language can be mapped into a same semantic space [6].

Training

Since the determination of target tokens is essentially a classification process, the Cross-Entropy Loss is applied here. It actually shares the same objective function as maximum likelihood estimation (MLE), which is

$$L(\boldsymbol{\theta}) = \sum_{t=1}^{n} log\ p(y_t|y_{<t}, \mathbf{x}; \boldsymbol{\theta}) \tag{3}$$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{argmax} L(\boldsymbol{\theta}) \tag{4}$$

The parameters can be updated following the rules of stochastic gradient descent by using mini batches

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \tag{5}$$

Inference

Due to the intractably large search space, it is only practical to find the local optimum in the generation process. The following are two commonly used search strategies.

1) Greedy search Greedy search picks the most probable word at each step. While the best solution at one step does not guarantee a best translation for the whole sentence. Thus the performance would degrade into a suboptimal one.

2) Beam search In contrast, beam search keep track of the top n best solutions. It would generally give a better result than greedy search, since it offers chances of looking back. We can tune the beam size to achieve a balance between computational efficiency and accuracy.

A problem of these methods is that it prefers shorter sentences, since the negative log likelihood (i.e. our objective function) tends to be larger when involving more words. This can be solved by length normalization and coverage penalty [6].

4 Proposed methods

Transformer

Transformer specifies a encoder-decoder structure, and relies only on the attention mechanism. In the encoder, the stacked self-attention makes source tokens share information and have a better understanding of each other in the context. The cross attention connects encoder and decoder for target token at the step to look at source

representations. And masked self-attention in the decoder links target token at the step to previous target tokens. The attention can be computed as

$$\textbf{Attention}(Q, K, V) = \textbf{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{6}$$

where Q is the query matrix, K is target matrix, and v is the value of the target words, from which model extracts the correlation between the query words and the target words.

Multi-head attention is applied, with each head independently focus on different features, which appear to have some connections with syntactic and semantic structure of the sentences.

As for the input of the model, it combines token embedding with positional embedding. Token embedding usually uses a pre-trained model to predict each token with its context; Positional embedding is used to learn the order of the sequence. A sinusoid positional encoding is given by

$$PE_{(pos,2i)} = sin(pos/10000^{2i/dmodel}) \tag{7}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/dmodel}) \tag{8}$$

There are also other tricks used in Transformer, like residual connection and normalization.

Transfer learning

Transfer learning focuses on transferring the knowledge learned from one task to a different but related problem [19]. The concept of transfer learning has been widely used in the NLP field, among which the pre-training and fine-tuning paradigm have achieved prominent results. For machine translation tasks, we also hope to try this paradigm to see if there will be some improvements. In this project, we specify five novel transformer-based models for trying the pre-training and fine-tuning mode.

T5

T5 [14] is a highly pre-trained and unified model. It considers all NLP tasks as text-to-text problems and pre-trains them uniformly so that all downstream tasks can be performed on a single model. It is based on transformer but removes the bias of normalization layer and not using fixed position encoding but the relative position between query and key instead (i.e. the offset value).

Transformer-align

Transformer-align [15] supposes that the past fixed-length vectors encoded in encoder-decoder structure reduces the performance of this structure, such as ability to correspond to long sentences in the evaluation. Therefore, it allows the model to improve the performance by soft-searching for parts of the original sentence that are relevant to the target sentence. The model encodes the input sentences to a suitable length each time and then selects the most relevant subset of them for decoding and translation.

We also tried other pre-trained models: bert2bert [16], a transformer-based model with both encoder and decoder as pre-trained bert; FSMT [17], a model using byte pair encoding; Deep-shallow [18], a structure that improves the speed and performance of an autoregressive model by deepening the number of layers of the encoder.

5 Experiment

Dataset

We use dataset **Multi30k** provided by Pytorch, which involves bilingual text of German and English. It is consist of 30k short sentences. We select German as our source language, and English as our target language (i.e. translate from German to English). The dataset is divided into training and validation sets, which contains 29,000 and 1,000 German-English sentence pairs respectively.

Baseline model

We use the classic Transformer as our baseline model.

Pre-trained models

We compare five pre-trained models, namely T5, Transformer-Align, Bert-to-Bert, FSMT and Deep-Shallow. These models have different model structures and pre-training tasks. We use their default tokenizers to process incoming and outgoing sentences.

Training strategy and parameters

The baseline model is trained directly using our dataset. We stop training when the loss on the validation set have no further decrease (i.e. converge);

The pre-trained models are fine-tuned by our dataset. The pre-trained weights are only used as an initialization, and we don't freeze any part of the model, as we want the model to learn the features of the our dataset better. These five models are only trained with the same number of epochs as the baseline case, even if they have not converged on the validation set at the stage.

Training parameters are set to be the same for all the cases. We specify Cross Entropy Loss as the loss function and AdamW as the optimizer, which gives an adaptive learning rate.

Evaluation

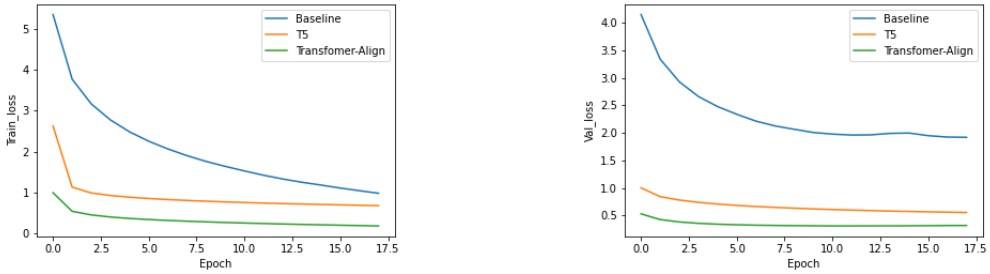
We use **SacreBleu** to evaluate the translation performance of the model. SacreBLEU provides hassle-free computation of shareable, comparable, and reproducible BLEU scores. Comparability is a very important feature relative to BLEU, since each model uses a different tokenizer maybe with subtle changes. We hope that our translation performance can be uniformly compared across multiple tokenizers.

6 Results and Discussions

We evaluate the loss at each epoch on both training set and validation set, and Bleu score on validation set, and make comparisons between the baseline and other proposed models towards the above metrics. Plots of the loss function and the Bleu score are demonstrated in Fig.1 and Fig.2 respectively.

For the training loss in Fig.1a, it can be found that the loss value decreases with epoches and all the three models achieve a relatively low loss in the end. While the pre-trained models initially have a relatively low loss value. And it should be noted that the pre-trained models only train the same number of epochs as the baseline and are not necessarily converged. If continue training, they have the potential of getting a better result.

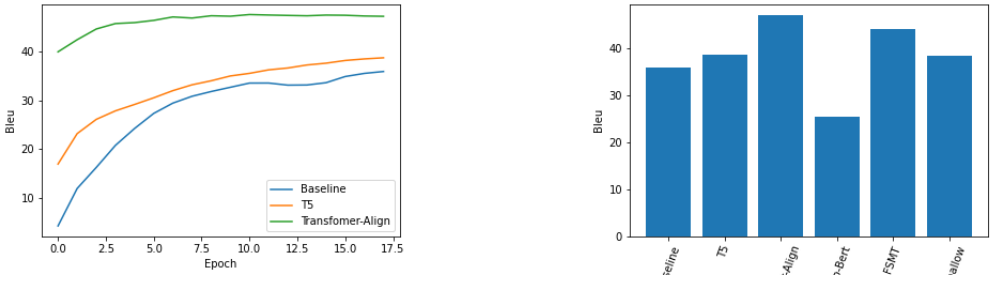
And for the validation loss in Fig.1b, the loss of baseline is much higher than that of the pre-trained model. The performance on the validation set indicates the generalization ability. And it is clear that the generalization



(a) loss trend for training set

(b) loss trend for validation set

Figure 1: The prediction loss of the model decreases successfully as the number of epochs increases. Comparing with the baseline, the pre-trained models are obviously performance better.



(a) Bleu score trend **(b)** Final Bleu score comparison

Figure 2: Comparison of Bleu score at different epochs and between different models.

Table 1: Examples of translation using different models for 3 sentences in Multi30k

Methods	Translated Sentences
Labels	A group of men are loading cotton onto a truck. A man sleeping in a green room on a couch. A boy wearing headphones sits on a woman's shoulders.
Baseline	A group of men loading pool into a truck. A man is sleeping on a couch in a green room. A boy with headphones sitting on his shoulders while sitting on a woman.
Transformer-Align	A group of men load cotton onto a truck. A man is sleeping on a couch in a green room. A boy with headphones is sitting on a woman's shoulders.
T5	A group of men load cotton on a truck. A man sleeps in a green room on a sofa. A boy with headphones sits on a woman's shoulders.
Bert-to-Bert	A group of manners ladt cotton on a truck. A sleep in a green room on a sofa. A young person with headphones sits on the shoulders of a frau.
FSMT	A group of men load cotton onto a truck. A man sleeps on a sofa in a green room. A boy with headphones sits on a woman's shoulders.
Deep-Shallow	A group of men loads cotton on a truck. A man sleeps in a green room on a sofa. A boy with headphones sits on a woman's shoulders.

ability of the baseline is not as good as that of the pre-trained model. The reason may lie in that the pre-trained models have prior knowledge, while the baseline is only trained on our dataset, thus having a risk of overfitting.

Fig.2a shows the Bleu score changing with the number of epoches. The scores of pre-trained models always exceed that of the baseline, especially for *Transformer-Align*, indicating that pre-trained models have a better translation performance. Also looking at the final Bleu of the baseline and five pre-trained models shown in Fig.2b, we can intuitively see that the paradigm of pre-training and fine-tuning can generally achieve better results in translation tasks compared to the baseline that does not apply a pre-training mode.

To sum up, the advantages of the pre-trained model over the baseline lie in two aspects. First, pre-trained models can start with lower losses and achieve lower final losses. Second, pre-trained models have a better generalization ability. For the translation task, the pre-trained model can not only promote the translation performance, but also reduce the work in the training phase to a certain extent.

In order to more intuitively feel the translation capability, we give examples of translation of three German sentences using different models. The result is shown in Table 1, and the true English label is given.

Attention visualization We also give visualization of attention weights for the encoder-decoder attention. Weights of different heads in the last layer are shown in Fig.3. Although different heads function on different aspects, we found that as the learning proceeds, they all learned the alignment of words with the same meaning between languages to some degree. And this effect is the most significant for *Head 0*.

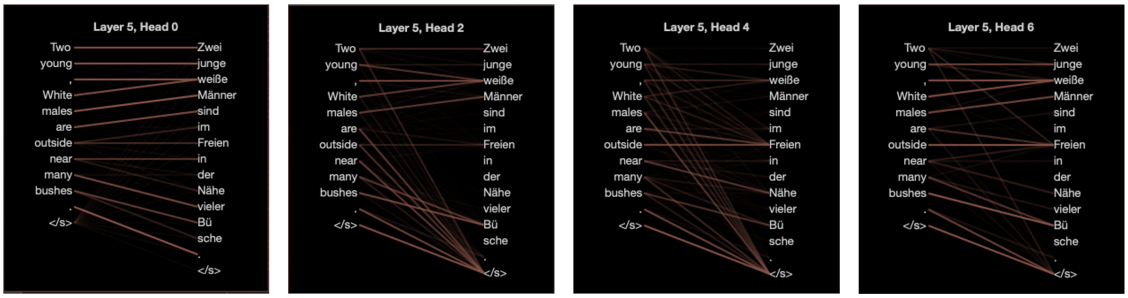


Figure 3: Visualize the cross attention weight of different heads in the last layer of the transformer-align model.

7 Future work

Due to time and hardware constraints, we did not complete all of the expected experiments. So far, the structure of the pre-trained models we have used is basically a sequence to sequence form, and the pre-training tasks contain generation tasks. We have also tried other pre-trained models and pre-training tasks, but we have not fully completed these experiments. We document our unfinished and hoped-for experiments here.

Pre-training tasks. If the pre-trained model itself contains a generation task, it even contains a translation task itself. In this case, our experiments have verified that pre-trained models can achieve a good performance on downstream tasks. But we also want to explore the effect of pre-trained models that only use nature language understanding (NLU) tasks in the pre-training phase, such as Roberta, XLMR.

Pre-trained models. Pre-trained models with sequence to sequence structure are undoubtedly very suitable for machine translation tasks, which is also the structure we use in our experiments. We also hope to explore the performance of pre-trained models of other structures on downstream tasks (e.g. only encoder, only decoder), or what kind of performance will be generated when the depths of encoder and decoder are inconsistent.

References

[1] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems 27 (2014).

[2] Gehring, Jonas, et al. "Convolutional sequence to sequence learning." International conference on machine learning. PMLR, 2017.

[3] Sundermeyer, Martin, et al. "Translation modeling with bidirectional recurrent neural networks." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[4] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[5] Tan, Zhixing, et al. "Neural machine translation: A review of methods, resources, and tools." AI Open 1 (2020): 5-21.

[6] Yang, Shuoheng, Yuxin Wang, and Xiaowen Chu. "A survey of deep learning techniques for neural machine translation." arXiv preprint arXiv:2002.07526 (2020).

[7] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[8] Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).

[9] Lewis, Mike, et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." arXiv preprint arXiv:1910.13461 (2019).

[10] Liu, Yinhan, et al. "Multilingual denoising pre-training for neural machine translation." Transactions of the Association for Computational Linguistics 8 (2020): 726-742.

[11] Conneau, Alexis, et al. "Unsupervised Cross-lingual Representation Learning at Scale." Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020.

[12] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

[13] Papineni, Kishore, et al. "Bleu: a method for automatic evaluation of machine translation." Proceedings of the 40th annual meeting of the Association for Computational Linguistics. 2002.

[14] Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." J. Mach. Learn. Res. 21.140 (2020): 1-67

[15] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

[16] Rothe, Sascha, Shashi Narayan, and Aliaksei Severyn. "Leveraging pre-trained checkpoints for sequence generation tasks." Transactions of the Association for Computational Linguistics 8 (2020): 264-280.

[17] Ng, Nathan, et al. "Facebook FAIR's WMT19 news translation task submission." arXiv preprint arXiv:1907.06616 (2019).

[18] Kasai, Jungo, et al. "Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation." arXiv preprint arXiv:2006.10369 (2020).

[19] West, Jeremy, Dan Ventura, and Sean Warnick. "Spring research presentation: A theoretical foundation for inductive transfer." Brigham Young University, College of Physical and Mathematical Sciences 1.08 (2007).