Neural Machine Translation

DDA4220 Final Project

Bao Zongbo (120090442)
School of Data Science
Chinese University of Hong Kong, Shenzhen
120090442@link.cuhk.edu.cn

Qi Xixian (120090691)
School of Data Science
Chinese University of Hong Kong, Shenzhen
xixianqi@cuhk.edu.cn

Shi Zhan (120090534)
School of Data Science
Chinese University of Hong Kong, Shenzhen
zhanshi1@cuhk.edu.cn

Abstract

In this project, we conducted experiments about Neural Machine Translation(NMT) from Chinese to English by utilizing variants of RNN, transformer, and pre-trained OPTUS-MT model. We used quantitative and qualitative evaluations to compare those models. The link of our project is available at this link.

1 Workload

All group member contributed equally to this project :

- Bao Zongbo (33%)Train RNN and transformer model, and write report
- Qi Xixian (33%): Train transformer and OPTUS-MT model
- Shi Zhan (33%): Train RNN model, conduct the five ablation experiments, and write report

2 Introduction

This study focuses on the issue of Neural Machine Translation (NMT), which has become a prominent research paradigm in machine translation due to the advancement of neural networks. Despite its potential, there are several challenges in NMT that need to be addressed to improve the translation quality:

Variable-length input sequences, which means that the model must be able to handle input sentences of different lengths. It can be challenging for some neural network architectures.

Long-term dependency problems, which refers to the difficulty of capturing long-term relationships between words in a sentence. This can lead to errors in translation, especially when dealing with complex sentences and idiomatic expressions.

Data scarcity, which is a common issue in machine translation, since large amounts of parallel data are required to train a neural network model effectively. However, high-quality parallel data is not always available, especially for less common language pairs.

To address the challenges in NMT, various neural network architectures have been proposed and developed. Recurrent Neural Networks (RNNs) have been widely used in NMT due to their ability to model sequential data [1]. Transformers models long-term dependencies and parallelizes computation, therefore gains popularity in recent years [2]. OPUS-MT, a pre-trained model that is fine-tuned on specific translation tasks, has also shown promising results in addressing the data scarcity issue [3].

Our project utilized these neural network architectures to facilitate the translation from Chinese to English. The report is structured as follows: Firstly, we employed a RNN-based model as the baseline. Subsequently, we investigated the LSTM architecture with additional layers, bidirectional LSTM, GRU, and explored the effects of activation function and teacher forcing. Finally, we evaluated the Transformer model and also incorporated the use of OPUS-MT, a pre-trained translation model, to further enhance our translation experiments.

3 Related Work

In this section, we provide an overview of several key structures utilized in NMT, including RNN, seq2seq, and transformer.

3.1 RNN and its Variants

LSTM (Long Short-Term Memory)[4] is a type of recurrent neural network that is designed to handle the vanishing gradient problem. It has an internal memory cell that can store information for long periods, allowing it to learn and remember complex patterns in sequential data.

Gated Recurrent Units (GRUs) [5] are another variant of Recurrent Neural Networks (RNNs) designed to address certain limitations of traditional RNN architectures. GRUs incorporate gating mechanisms, allowing for better long-term dependency modeling and alleviating the vanishing gradient problem.

3.2 Seq2Seq Architecture

One important framework for RNNs is Seq2Seq [6], which revolutionized NMT by providing an end-to-end approach for translating sequences. It consists of an encoder-decoder structure, where the encoder takes in the input sequence and encodes it into a fixed-length vector representation, which is then fed into the decoder network along with its own hidden state to generate the output sequence.

3.3 Transformer

The Transformer model[2] is a neural network architecture that relies solely on self-attention mechanisms without using any recurrent connections. This makes it faster to train compared with traditional RNN-based approaches while being able to handle long-range dependencies efficiently. Transformers have emerged as one of the most popular architectures for machine translation due to their efficiency at handling longer sentences and parallelizable nature across multiple GPUs or CPUs.

4 Method

4.1 Model Architecture

4.1.1 RNN-based Encoder-Decoder

We choose LSTM[4]-based encoder-decoder architecture as our baseline model. LSTM (Long Short Term Memory) is a type of recurrent neural network architecture that uses gates to selectively retain or discard information at each time step. These gates are:

- Forget gate: This gate determines which information from the previous hidden state should be forgotten and not passed on to the current state.
- Input gate: This gate controls how much new information from the input should be added to the current state.
- Output gate: This gate decides how much of the current state should be outputted as part of the final prediction.

These gates allows LSTMs to effectively capture long-term dependencies in sequential data, which made it popular in machine translation area.
During training, we found the valid loss doesn't decrease as we expected. We supposed it is resulted from the drawback of LSTM in which a wrong prediction from decoder can make bad influence on

the training inthe following steps. Therefore, we used teacher forcing technique[**?**] to mitigate this problem. The modified model is LSTM_teacher_forcing.

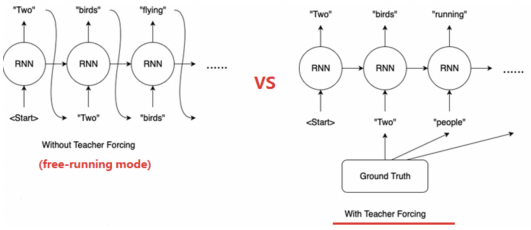

Figure 1: Teacher Forcing Example

Teacher forcing involves using the true output sequence in some time step as input to the decoder during training. As shown in Figure1, in the third time step, we didn't use the output "birds" from the previous step, but used "people" instead. During training, this approach can help improve convergence and stability by reducing error propagation over multiple steps. It also allows for faster and more efficient learning since it provides immediate feedback on whether or not predictions are correct.
We also did some other attempts to get a better performance:

- Change LSTM into GRU model
- Pile 5 LSTM layers both in encoder and decoder
- Adopt Bidirectional LSTM to model bidirectional infoamation better

Therefore, we have trained 5 diffrent versions of seq2seq variants, including single directional LSTM as the baseline model, LSTM with teacher forcing method, GRU, LSTM with additional 5 layers, and bidirectional LSTM.

4.1.2 Transformer

As transformer[2] model is getting popular recent years, we also used it in our project. It relies on self-attention mechanisms to encode and decode input sequences, allowing for more efficient processing compared with traditional recurrent neural networks like LSTMs. In the Transformer architecture, both the encoder and decoder consist of multiple layers of self-attention blocks followed by feed-forward neural networks. Self-attention allows each word or token in the sequence to attend to all other words or tokens in the same sequence, capturing global dependencies between them without requiring explicit alignment information.

4.1.3 Pre-trained model

We also used pre-trained model as a comparison. The model we chose is OPUS-MT proposed in [3].The model is based on the state-of-the-art transformer model with standard setups(6 self-attentive layers in both encoder and decoder network, and 8 attention heads in each layer).We used the API provided in huggingface (transformers.AutoModelForSeq2SeqLM)

5 Experiments

5.1 Evaluation method

- Perplexity (PPL): Suppose a sentence is consists of words, i.e., $s = w_1 w_2 \cdots w_n$, where w denotes the word. Then,

$$PPL(S) = P(w_1 w_2 \cdots w_n)^{-\frac{1}{N}} \tag{1}$$

- BLEU:

$$BLEU = BP \times \exp(\sum_{i=1}^{N} W_n \times \log P_n) \tag{2}$$

5.2 Experimental details

5.2.1 Dataset

Our training and validation sets are translation2019zh. The training and validation set is 800k and 20k. All the experiments are conducted via V100.

5.3 Results and analysis

5.3.1 Quantitative Results

This section presents an analysis of the quantitative results obtained from our models, including the training time of each model, metrics such as loss, BLEU score, and perplexity (PPL).

The loss curves of both the RNN and Transformer models, during both the training and validation phases, are illustrated in Figure 2. OPUS-MT, with its distinct loss criteria and encapsulated implementation within the Hugging Face API, is not included in the figure. Its performance and evaluation are considered separately in our analysis.

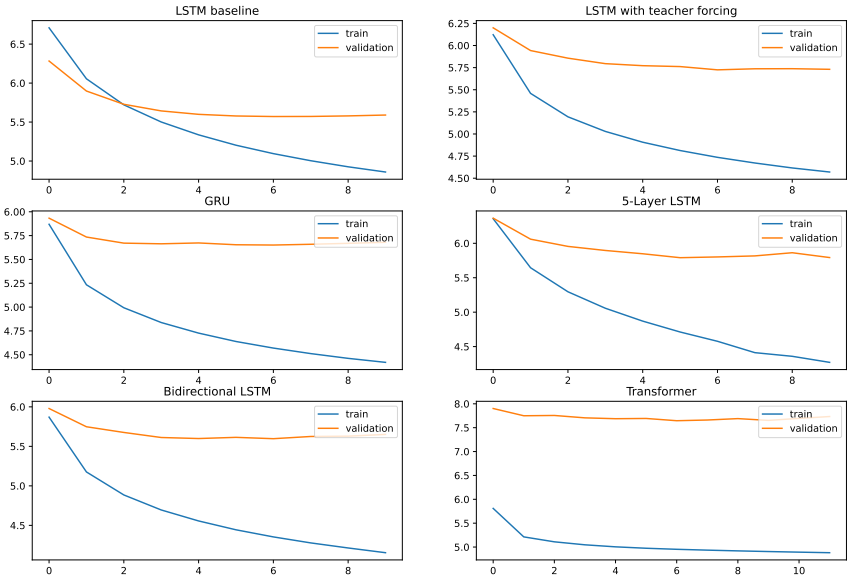

Figure 2: Visualization of the training loss and validation loss of RNN and Transformer models.

The training time for the six models is depicted in Figure 3, with the X-axis representing the training time and the Y-axis representing the number of parameters. Interestingly, despite the increased model complexity of the 5-layer LSTM, its training time remains moderate.

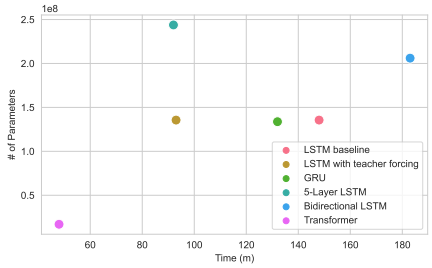

Figure 3: Training time and number of parameters of RNN and Transformer models.

We can obtain the following results:

- GRU demonstrates better performance and requires less training time compared to single-directional LSTM. This advantage can be attributed to GRU's innovative gating mechanism.

- The method of teacher forcing can significantly reduce the training time of NMT models, and can be applied to all the models evaluated in this study.
- Increasing the depth of LSTM models tends to reduce the likelihood of overfitting, as indicated by the continued decrease in training loss. But it will also result in an increase in model complexity.
- The bidirectional LSTM with teacher forcing generally exhibits superior performance, in spite that it will cost a longer training time.

Interestingly, Transformer exhibits comparatively poorer performance, which may be attributed to the relatively small size of the dataset. With a limited amount of data available for training, the Transformer model may struggle to fully capture the underlying patterns and complexities of the task.

The table for BLEU and PPL scores are presented as follows.

Metric	LSTM orig	**LSTM bi**	GRU	LSTM 5	LSTM tf	Transformer	**OPTUS-MT**
PPL	327.5	**262.8**	284.7	284.6	269.5	1508.2	**202.4**
BLEU	0.18	**0.26**	0.25	0.25	0.28	0.04	**0.40**

Table 1: PPL and BLEU scores for RNNs and Transformer models

In terms of PPL and BLEU scores, we observe that OPUS-MT achieves the best performance, while the bidirectional LSTM exhibits the highest performance among the evaluated models. This finding is consistent with the results obtained from the loss analysis.

5.3.2 Qualitive Results

This section presents a case analysis of several translation examples. The results are presented in Figure 4. Interestingly, we find that models with better PPL and BLEU scores do not necessarily perform better in real translation tasks.

Translation Examples

source: 祝同学们最后的大作业顺利

RNN (baseline):	<sos> <unk> the to to to to . . . <eos>
LSTM with teacher forcing:	<sos> <unk> Your company 's the to the the of the . . <eos>
GRU:	<sos> <unk> <unk> the best of the the . <eos>
Bidirectional LSTM:	<sos> <unk> The good of of the students is the final . . <eos>
LSTM with 5 layers:	<sos> <unk> The lessons of the in the . <eos>
Transformer:	<sos> <unk> <unk> the the the the . . . <eos>
OPTUS-MT:	<sos> Good luck for the students final project . <eos>

Figure 4: Training time and number of parameters of RNN and Transformer models.

For instance, although the Transformer model has significantly worse PPL scores compared to the baseline model, the translation results are similarly poor, producing mostly meaningless output.

6 Conclusion

In this project, we trained the variants of RNN, transformer, and OPTU-MT modeland evaluate their result by quantative and qualative perspective. We found Bidirectional LSTM performed best in all 5 versions of RNN variants, which may due to its great ability to extract the information from two direction. The transformer model unexpectedly performed poorly, which may due to the lack of training data. The pre-trained model OPTUS-MT outperformed significantly compared with other models, which is expected as it was SOTA model. The future work might be modified the transformer model by simplifying the model and add more training data to avoid overfit.

References

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

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

[3] Jörg Tiedemann and Santhosh Thottingal. Opus-mt–building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*. European Association for Machine Translation, 2020.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

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

A Appendix

A.1 Data Processing

We transform a sentence into a sequence of tensors before fed it into the model. The step is as follows:

We first tokenized the sentences to get a list of tokens with the help of in-built tokenizers of spacy("zh_core_web_sm" and "en_core_web_sm").Then, a vocabulary is built by collecting all unique tokens from the corpus of training data and only the tokens that appreared at least 20 times are added into the dictionary. Some special tokens(i.e. <unk>,<sos>,<eos>,<pad>) are also included. Each word is assigned an unique ID that are used to represent the word as a one-hot word vector. After inserting the ID of <sos> and <eos> in the beginning and the end of the sentences, we concatenate a batch of tensors and fill the gap with <pad> ID to get a tensor that are ready to fed into the models.