Neural Machine Translation System Project

Hong Gao
New York University
hg1196@nyu.edu

Xiaoxue Ma
New York University
xm576@nyu.edu

Yulin Shen
New York University
ys2542@nyu.edu

Jie Yang
New York University
yangj14@nyu.edu

Abstract

In this project we explore different neural network methods to build a neural machine translation system that can intake Chinese and Vietnamese languages and translate into English. We used Recurrent Neural Network (RNN) to build a sequence to sequence model without attention, and another one with attention. In this process we tuned a few parameters, including tying to use fasttext as embedding matrix. Lastly, we used a self-attention based encoder instead of RNN. In our case, the best translation model from Vietnamese to English is the model with attention and hidden size 128, and the best translation for Chinese to English is the model with self attention and hidden size 300.

1 Introduction

Neural Machine Translation (NMT) has gained popularity in recent years since it doesn't require domain knowledge and is able to achieve good performances(Luong et al., 2015). Various models have been developed to improve state-of-the-art performance. In 2014, Sutskever et al.(Sutskever et al., 2014) developed an end-to-end sequence to sequence learning network with multilayered long short memory network (LSTM). They also found that using a reversed input sequence helped improving performance of their model. Cho et al. (Cho et al., 2014) used two recurrent neural networks (RNN) as encoder and decoder. Later in 2015, Bahdanau et al. introduced a new method based on the encoder-decoder structure, that is able to attend to the part of input useful for the output, which is called attention (Bahdanau et al., 2014). This network outperformed all the conventionnal RNN encoder decoder by at least 1 Bilingual Evaluation Understudy Score (BLEU score). In 2017, the Transformer network was introduced and reached the state-of-the-art BLEU score of 41.8 (Vaswani et al., 2017). They abandoned the RNN structure and only used attention mechanisms.

In this paper, we leverage what we have learned from the Neural Language Processing with Representation Learning class to build a neural translation machine. Based on the literature, we implement three different models to translate Vietnamese and Chinese to English respectively.

2 Data

2.1 Data Source

The raw data is from the International Workshop on Spoken Language Translation (IWSLT), which is the automatic transcription and translation of TED and TEDx talks crawled from the TED website (IWSLT2015, 2015). The dataset contains multilingual and bilingual tasks. For our paper, we used two of the bilingual sets, from Chinese and Vietnamese to English.

The datasets are provided by the class both in original format and in tokenized format, in Viet-

namese to English and Chinese to English pairs. Datasets are already separated into training, validation, and test sets.

2.2 Preprocessing and Tokenization

We use the cut function(cut_all = False) in Jieba package (Sun, 2018) to tokenize the Chinese sentences. In this way, we got cleaner tokens than the given data. We also remove some garbled in English data and cut all the sentences whose length are longer than our pre-defined max length.

2.3 Data Understanding

Take Chinese-English sentences data as an example. After tokenization, we got pairs like Figure 1.

```
我 看到 了 这 一点 ， 我 感到 那 是 很 震撼人心 的
So I saw that and it was very humbling .
```

Figure 1: Tokenized sentences pairs

As is shown in Table 1, we have almost twice of data and vocabulary for Chinese than Vietnamese.

	zh train	zh val	zh test	vi train	vi val	vi test
sentences pairs	213378	1262	1398	133318	1269	1554
vocabulary	91143	4973	4264	42152	3918	3698

Table 1: Counted pairs & words

Chinese-English sentences length distribution is demonstrated in Figure 2. Chinese language has relative short sentences than English.

3 Methods

We would like to train a few neural machine translation models on pairs of source sentence and its correct translation, and use BLEU score to evaluate the models.

This can be achieved by first applying a recurrent neural network (RNN) to encode the

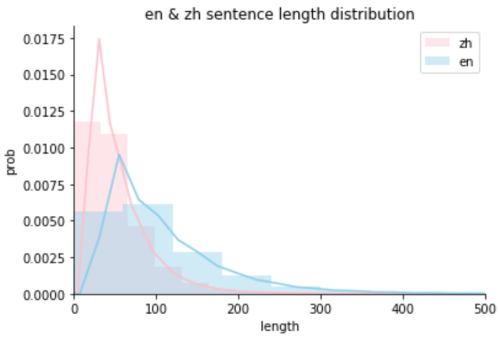

Figure 2: Sentences length distribution

source sentence into a continuous vector (context vector). Second, use a recurrent neural network as a decoder, which intakes the context vector from encoder and the previously predicted translated sentence, and output a probability of the next translated word. In a nutshell we used the following approaches:

- RNN Decoder without Attention
- RNN Decoder with Attention
- Self Attention

Then we tuned a few important hyperparameters, tried pre-trained embedding, and compared the performances across different methodologies.

3.1 FastText

In addition to the simple RNN encoder decoder network, we also use FastText as pre-trained word representations to explore the best model. In order to do this, three FastText files for English, Chinese, and Vietnamese were downloade respectively from https://fasttext.cc/. There are about 1 million English words and about 2 millions Chinese and Vietnamese words in the files. In practice, we load 100,000 words in each file as they should be enough to cover common words. Words out side of this vocabulary are coded as <UNK>. We set embedding size as 300.

3.2 Sequence to Sequence without Attention

Architecture

In this model, we use two RNN to transform one sequence to another. An encoder network condenses an input sequence into a vector, and a decoder network unfolds that vector into a new sequence (PyTorch, 2017). Our codes are based on the Lab8 in class, PyTorch tutorial (PyTorch, 2017) and Practical Pytorch (Robertson, 2018). The architecture is shown in Figure 3. First we encode the source sentences into context vectors. Then we decode the context vectors with greedy search to get the final predicted sentences. In our application, we use an one layer GRU as our RNN model in both encoder and decoder. We set our max length to be the length of the sentence whose length is longer than 99% of training data.

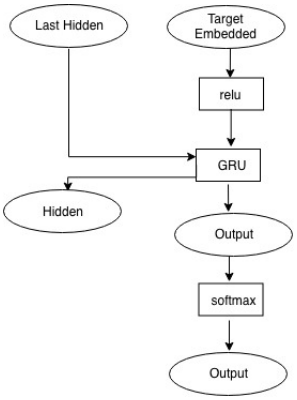

Figure 3: Decoder without Attention

Parameters

When using fast text, because the embedding size is fixed at 300, only hidden size was tuned for both 500 and 300. Hidden size of 500 yields the best result, and the result is significantly better than hidden size 300 whose BLEU score is 6.54 for Vietnamese and 6.16 for Chinese on validation data. For hidden size 500, we get bleu score 8.15 for Vietnamese and 6.71 for Chinese on validation data. When using random initialized embedding and hidden size of 500, the validation BLEU score is 8.67 on Vietnamese and 6.75 on Chinese.

Results

The best simple RNN model is with random initialized embedding, the performances of which is 8.35 for Vietnamese and 6.00 for Chinese on test set.

One of the training curve example is shown in Figure 4. In the beginning, the result seems pretty good which has BLEU score above 20. However, we found that keeping <SOS>and <EOS>in predicted and labelled sentences may lead to higher BLEU score because each sentence would have at least two correct words mentioned above. After deleting them, the score decreases significantly. In addition, although the FastText method uses less training time, it does not perform very good. Moreover, a simple RNN model compress a sentence regardless of the sentence length, the results might improve when we incorporate attention mechanism in the model.

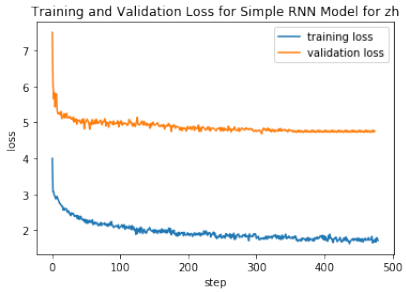

Figure 4: Simple RNN for zh

3.3 Sequence to Sequence with Attention

Architecture

Based on the previous RNN encoder decoder model structure, the second model included a attention mechanism. Attention is added in the decoder, which allows the model to 'pay attention' to particular parts of the input during decoding process(Bahdanau et al., 2014). In detail, we use the same GRU encoder with model 3.2. The last hidden state of the encoder and target embedding are

passed through another GRU layer, the product of which is combined with encoder output to produce attention weights. We explored two methods to calculate the attention weights, linear and dot product, to compare the performance. After that, matrix multiplication is applied between the weights and encoder output, which is then concatenated with GRU output and feed into a linear layer for the final prediction.

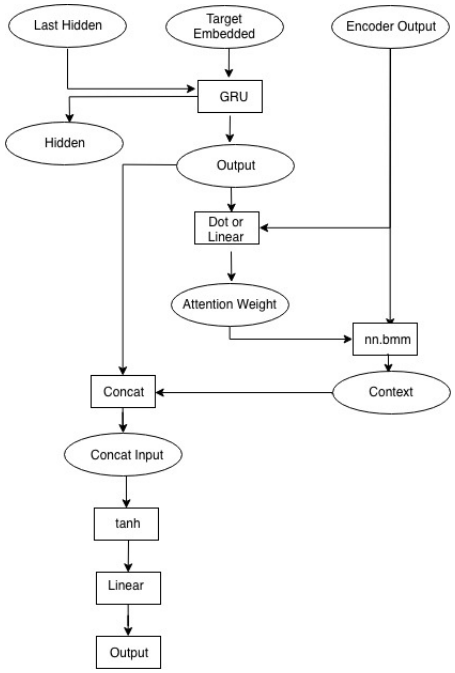

Figure 5: Decoder with Attention

Parameters

We highly referenced the Pytorch tutorial for building the initial models(PyTorch, 2017).

(a) Embedding

Two types of embedding are used, FastText and random initiation. FastText uses embedding size of 300. Random uses embedding size of 128, 300, and 500.

(b) Attention Mechanism

We use general linear layer and dot product to generate attention weights. For the former, FastText with 500 hidden size and random initialization with 300 and 500 hidden size are used. For the dot method, we used hidden size 128.

(c) Other

All the models use teacher forcing rate of 1, batch size of 2, dropout rate of 0.1, and learning rate of 0.0001. We imposed maximum sentence length of 70 on validation and test sets, while no limit was set on training set.

On validation set, randomly initiated embedding performs surprisingly better than FastText on BLEU score. With the same hidden size 500, FastText has BLEU score of 3.54 on Chinese and 6.88 on Vietnamese, while random embedding has 6.72 on Chinese and 11.03 on Vietnamese. Performance on different hidden size (300 and 500) indicate the optimal hidden size might vary between Vietnamese and Chinese, where Chinese translation benefits more from bigger hidden size. Among all the parameters tuned, attention weights from a dot product reached the highest BLEU score 16.43 on Vietnamese and the highest score 8.15 on Chinese with a much smaller hidden size.

Results

Comparing to using FastText, random initiated weights takes much longer to train. Training and validation losses of one of the best performing models are showed in Figure 6. In the models, we experienced a large drop in training loss starting from the second epoch. This might due to the frequent updating of the weight matrix.

The best performing model, attention with a dot method and hidden size 128 is compared with the other two models on the test set. Results are in Table 2. We are able to reach BLEU score of 15.72 on Vietnamese and 8.87 on Chinese.

3.4 Self Attention

Architecture

In this model, we use a self_attention based encoder based on Transformer (Vaswani et al., 2017), then combine with our decoder with attention. Our codes are based on The Annotated

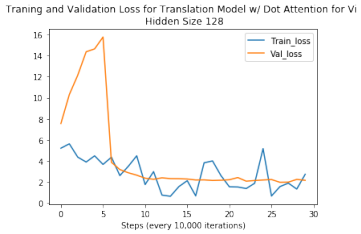

Figure 6: Loss for Model with Attention for Vietnamese

Transformer (Rush, 2018). The model architecture is shown in Figure 7.First, we encode the source sentences into embedding vectors. Since our encoder does not contain RNN or CNN architecture, the network doesn't know the order of a token. A positional matrix is added to inject information about the token positions in a sequence (positional encoding). Dropout was applied to the sum of embedding and the positional encoding.Word embedding is then fed into 6 identical layers. Each layer consists of two sub-layers. The fist one is a multi-head attention cell, where we separate our embedding vector into several pieces and perform attention separately, then concatenate the output back together. All of the keys, values, queries come from the output of previous layers, which allows the encoder to get exposed to information from all positions. The second sub-layer is a position-wise fully connected feed-forward network, which applies to each position respectively. We also conduct a residual connection around each of the two sub-layers, followed by layer normalization. So the output of each sub-layer would be LayerNorm(x+Sublayer(x)).After that we feed the encoder output to our original attention decoder mentioned in section 3.3.

Parameters

(a) teacher_forcing_ratio

We explored the teacher_forcing_ratio as 0, 0.5, 1.0, respectively. When ratio is 1.0, the model yields the best result (BLEU score 8.67 for Vietnamese and 8.96 for Chinese on validation data). When ratio is 0, the model cannot give a reasonable translation output at all. It shows that teacher forcing learning is important in our translation system since our own prediction may not be very nice during training.

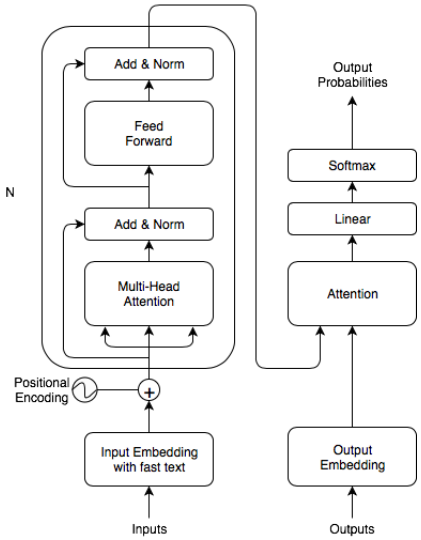

Figure 7: self attention mechanism

(b) ff_size

We try to use different value of ff_size (120 and 1200), which is the inner-layer dimension within feed-forward network. The result is similar to each other, around 9 for both Chinese and Vietnamese on validation set.

Results

In order to save some training time, we tune this self-attention model with fast text pre-trained word embedding, After hyper-parameter tuning, we got a BLEU score of 8.62 for Chinese and 9.56 for Vietnamese on the test data. And the training and validation loss is shown in Figure 8, a step contains 100 batches.

The result is not ideal, even compared with simple encoder-decoder architecture with attention. This result reminds us that simply putting the output of self-attention encoder into decoder with attention may not be a reasonable process. Maybe a combination of self-attention encoder with self-

Method	hs*	vi	zh
simple RNN	500	8.35	6.00
RNN with attention	128	15.72	8.87
Self-attention	300	8.62	9.56

Table 2: BLEU Score in Test Dataset (* Hidden Size)

attention decoder will perform better.

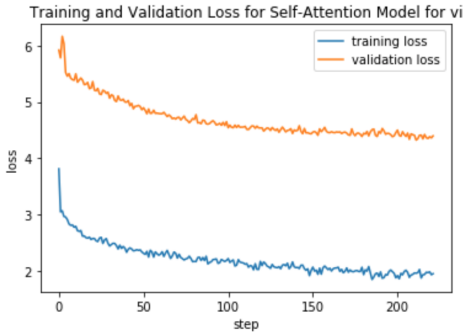

Figure 8: self attention for vi

4 Conclusion

4.1 Examples

Below is an example of the real translation from vietnamese vs. what's translated from our attention model (with hidden size = 500):

Target: *And there is this stunning silence.*

Prediction: *And there's a silence of a good*

4.2 Discussion

Please see Table 2 for the comparison of the BLEU scores. In general, attention models perform much better than RNN encoder-decoder network. Self-attention model reached the highest BLEU score among all the models in Chinese, but doesn't perform as well in Vietnamese. Attention model reaches the highest score in Vietnamese and outperform self-attention by 7.1 score.

To sum up, attention mechanism could help to reach higher BLEU score. FastText saves significant amount of time in training, but would sacrifice the performance. Ideally, models with different structure and parameters should be trained for different languages to ensure the best performance.

5 Future Work

As potential improvements in the future, we suggest performing a full transformer model with beam search for the self-attention mechanism. In addition, ensemble different models to check if it outperforms a single model.

6 Github Link

https://github.com/IceRakka/Neural-Translation-System

7 Contribution

Hong Gao: Create data loader. Improve upon base code and build RNN encoder-decoder network and attention RNN models with random initialized weights.

Xiaoxue Ma: Tokenize Chinese-English data. Prepare and analyze data. Build self-attention based encoder model.

Yulin Shen: Preprocess the data with FastText and build its dataloader. Build simple RNN and attention RNN models with greedy search evaluation.

Jie Yang: Build the RNN encoder and decoder with attention using greedy search to make it work on one single batch.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

IWSLT2015. 2015. The International Workshop on Spoken Language Translation (IWSLT) kernel description.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

PyTorch. 2017. Translation with a sequence to sequence network and attention.

Sean Robertson. 2018. practical-pytorch. https://github.com/spro/practical-pytorch.

Alexander Rush. 2018. The annotated transformer. https://github.com/harvardnlp/annotated-transformer.

Junyi Sun. 2018. Jieba. https://github.com/fxsjy/jieba.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*, Montreal, CA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.