Machine Translation System Using Seq2Seq Models

Jing Wu
jw3162@nyu.edu

1 Introduction

Machine translation is a difficult problem but this field has seen significant progress in the past years. Specifically, the adoption of neural networks in this field has helped tremendously. (Ilya Sutskever and Le, 2014) proposes an end-to-end neural-network-based framework that pushes the state-of-art BLEU score on WMT-14 dataset to 34.8 from 33.3 of a phrase-based SMT system. Since (Ilya Sutskever and Le, 2014), many neural machine translation models follow the encoder-decoder framework, where the encoder tries to learn some internal representations from the input sequence, and the decoder tries to generate the output sequence based on the internal representations from the encoder. One major improvement on the original encoder-decoder framework is the adoption of the concept of "attention" (Minh-Thang Luong and Manning, 2015), which essentially incorporates more local temporal information to make decoding easier. In this project, we implemented several instances of the encoder-decoder framework, including a RNN-based encoder, a self-attention encoder, and a RNN-based decoder. We experimented our models on two datasets, of which one is Chinese to English and the other is Vietnamese to English. Our results show that the adoption of attention in the decoding process is essential to achieve good performance. Furthermore, our results also show that a model can perform well for multiple language pairs, which suggests that the encoder-decoder Seq2Seq framework is able to adapt to language-specific traits and properties.

2 Problem Formulation

The machine translation problem is essentially to find a mapping $f : \mathbf{x} \rightarrow \mathbf{y}$, where $\mathbf{x} \in \mathbb{N}^{d_{in}}$ and $\mathbf{y} \in \mathbb{N}^{d_{out}}$. The optimal mapping f^* would be the one that optimizes some evaluation metrics, which we will cover in the next section.

Specifically, in this project's context, each element of $\mathbf{x}$ and $\mathbf{y}$ represents the index number into source language's vocabulary and target language's vocabulary, respectively.

3 Evaluation Metrics

BLEU is used in this project. BLEU is short for the Bilingual Evaluation Understudy Score. The scores are calculated for sentences and averaged for the final score for a corpus.

BLEU is a modified version of precision. The number of the same true positive is limited in BLEU to prevent exploiting some easy targets. Also, BLEU is usually not computed in the word level, but a n-gram level, so that the order of translation is also paid attention to. Specifically, we used four-gram in this project.

Furthermore, since BLEU tends to prefer shorter translations becuase the denominator in the definition of precision is smaller. Therefore, a

brevity penalty is usually imposed on the modified version of precision.

In this project, we used `sacreBLEU` (https://github.com/mjpost/sacreBLEU).

4 Approach

The overall framework of this project's Seq2Seq implementations is an encoder-decoder framework. Neural networks are used to build both the encoder and the decoder. The encoder's job is to take sequences of variable lengths and transform them to some meaning representations. Then the decoder tries to emit sequences of variable lengths from the representations. The output sequences are then compared to ground truth translations to compute the loss.

We discuss various challenges and implementation details of this framework in the following subsections.

4.1 Word Embeddings

First, how to select a good representation for the input has significant impact on the framework's performance. Since from the dataset, we can only get a pair of index lists, which is extremely sparse step-wise because there is only one word out of a huge vocabulary and that makes learning very difficult. One solution is to use dense word embeddings to overcome the sparsity problem of vocabulary encoding. For example, instead of use a vector of length V, which can be more than 10,000, we use a vector of length 300 to represent the word. As word embeddings are usually encoded with semantic meanings, they are not only dense but also contains richer information than the vocabulary encoding, which has no semantic meanings.

4.2 RNN-Based Encoder

The framework of an encoder consists of two components. First, the embedding lookup component that finds the corresponding word embeddings given word indexes. Second, the encoder that transforms a list of inputs to some representations.

Recurrent neural network seems a natural fit for this type of problems. In this project, we used GRU (Kyunghyun Cho and Bengio, 2014) as the building blocks for our RNN-based encoder.

The output representations from the encoder is its final hidden state, which is supposed to have seen the entire sequence and thus be helpful for the decoder.

We implemented the RNN-based encoder using one-layer bidirectional GRU cell with a hidden size of 128.

4.3 RNN-Based Decoder

Recurrent neural network is also a good choice for building the decoder as we want to emit sequences of variable lengths.

Similar to the RNN-based encoder, the RNN-based decoder is built with GRU cells. Note that the initial hidden state to the RNN-based decoder is the final hidden state of the RNN-based encoder. To simplify the connection mechnism between the encoder and decoder, we used the same hidden size for GRU cells in both the encoder and the decoder.

There is one problem about the naive RNN-based decoding process. In the context of machine translation, the representation passed to decoder is a mixed version of the input sequence, which might have lost some of the temporal information, namely the order of input words, and thus when the decoder is decoding, it might be difficult for it to translate in the right order.

To solve the problem, (Minh-Thang Luong

and Manning, 2015) proposes a mechanism that pays special attention to certain part of the input sequence while emitting the output sequence. For example, in order to translate "I woke up in the morning", the decoder should pay more attention to the part around "I" when decoding "I", and more attention to the part around "in the morning" when decoding "in the morning". This mechanism resembles better how humans do translation, by focusing a small part of the input sequence at a time.

Specifically, to implement the concept of "attention", here are the steps to follow.

First, compute attention scores. Attention score is a similarity measure between the decoder hidden state and encoder hidden state at a certain time. In this project, we used dot product to compute the attention score.

Second, we normalize the attention scores by applying a softmax layer.

Third, we compute a sum of the encoder's hidden state at all steps, weighted by the preceding softmax scores. Therefore, the resulting vector is an attention-adjusted view of the entire input sequence.

Finally, we concatenate the weighted sum with the decoder hidden state, and use the resulting vector to compute the decoder's output at this step.

We implemented the RNN-based decoder using one-layer bidirectional GRU cell with a hidden size of 128.

4.4 Self-Attention Encoder

One problem of the RNN-based encoder is that, while matrix multiplications can be computed in parallel in GPU, recurrent neural network needs to be computed sequentially due the nature of the model.

(Ashish Vaswani and Polosukhin, 2017) proposes a new encoder-decoder structure, named *Transformer*, which includes a module called *Self-Attention*. Unlike RNN-based models, this model does not compute sequetially, instead, this model computes sequence data in one step. Self-attention structure resembles bidirectional RNN in that both try to incorporate global context information into one step's output. Self-attention does that more efficiently as it is easier for parallel computing.

Another advantage is that while extracting internal representations of inputs, the encoder is considering attention-based information already, because of which the entire system is very likely to be better than the one that just uses the simple dot product in the decoder's attention mechanism.

In this project, we implemented the self-attention encoder using different architecture hyper-parameters than those in (Ashish Vaswani and Polosukhin, 2017). We used 6 stacked encoder block, 8 attention heads, 300 for hidden size, 64 for query size, 64 for key size, and 64 for value size. For the feed-forward sub-network, we used 1200 for the first fully-connected layer's hidden size.

Furthermore, since we only replaced the encoder with self-attention-based components and our decoder is still RNN-based, we need to decide what is the "hidden state" from the encoder to our decoder. In this project, we extract the last step's output from the last encoder block's output as the initial hidden state for the decoder.

4.5 Beam Search

Beam search is a way to help the decoding process. When we do decoding in the greedy way, we feed the most likely word, based on current step's output, to the next step. This approach is not optimal in that the global optimum does not necessary make locally optimal decisions.

Instead of using the last best prediction for

the next step, we use last k-best predcitions to compute the next step. Suppose the vocabulary size is V, then there would be V likelihood outputs for each of the k inputs. We then select the top k most likely sequence from the kV candidate sequences.

Although beam search does not gaurantee global optimum, it should be better than greedy search as it explores more candidate sequences.

4.6 Training

To train the encoder-decoder model, we feed the ground truth to the decoder step by step, and compute cross entropy loss using the current step's output distribution against the true next word. We stop when the ground truth target sequence ends.

Suppose the target sequence is $\mathbf{y} \in \mathbb{R}^{T \times V}$, and the predicted sequence is $\hat{\mathbf{y}} \in \mathbb{R}^{T \times V}$, where $\mathbf{y}^{(t)}$ is the one-hot encoding at step t, and $\hat{\mathbf{y}}^{(t)}$ is the likelihood distribution at step t. We define our loss as

$$Loss(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{t=1}^{T}\sum_{v=1}^{V} \mathbf{y}_v^{(t)} log(\hat{\mathbf{y}}_v{}^{(t)}) \quad (1)$$

4.7 Inference

During inference, the maximum length for output sequence is set to 100 in our implementation. The final output sequence is the best sequence from beam search and then trimmed to the ⟨eos⟩, which is the end of speech token. Also, any other special tokens are removed from the predicted sequence.

5 Data

To build the vocabulary before training our models, we pre-processed the given token-based datasets. First, we added the following special tokens: ⟨unk⟩, ⟨start⟩, ⟨eos⟩, ⟨pad⟩. Then, we went through the training data and filtered out infrequent tokens. We kept thoese tokens that appear at least 3 times in the training data.

After pre-processing, we have 17731 unique tokens for Vietnamese, 28811 unique tokens for English, and 47132 unique tokens for Chinese. The filtering step is important, because our vocabulary size is significantly smaller than the FastText vocabulary size, otherwise we would have out-of-memory errors because the entire word embedding matrix is just too large to fit in ordinary machines.

6 Experimental Setup

6.1 Pre-Trained Embeddings

We used FastText word embeddings in this project. FastText word embeddings were trained using the Skip-Gram approach, which uses context words to predict the center word. The embedding dimension is 300 for all three languages. We created random vectors as initialization for the special tokens, and make all the word embeddings trainable. Furthermore, some words that appear in our training datasets do not appear in the vocabulary of FastText vacabulary. We also initialize a random word vector for each of such words. For example, there is no apos;ve, namely "'ve", in FastText vocabulary.

6.2 Implementation

Github link to the model implementations and data/training/testing pipelines: https://github.com/jw3474/nlp_final_project.

This project is implemented using `pytorch`. From this project, we found that implementing the end-to-end Seq2Seq framework and all the related decoding algorithms is non-trivial.

6.3 Training Setup

We used Adam optimizer to train all our models. Learning rate was set to 10^{-3}. Batch size was

set to 64 and max sequence length was set to 100. Beam search width was set to 5.

We trained all our models on one Nvidia Tesla K80 GPU, and each batch takes about 1 second.

Each model was trained for 3 epochs.

7 Results

	Chinese	Vietnamese
RNN	16.90	15.99
RNN with Attention	21.88	23.03
Self-Attention Encoder	16.48	16.08

The above table shows our results for the two language pairs with several encoder-decoder variations. The numbers are BLEU scores computed over the entire testing corpus. In general, the performance of a model on the two language pairs is similar while different model architecture can lead to very different performance.

7.1 Impact of Attention

Our results show that the impact of the attention mechanism during decoding on the performance is enormous. For Vietnamese to English, using attention improves BLEU score by 7.04. For Chinese to English, using attention improves BLEU score by 4.98. The results verify the intuition that in order to find out the correct meaning of a word in certain context, we need to also take a look at other places in this context, not only relying on a language model, which is conditioned only on the previous words.

7.2 Training Process

Figure 1: Training Loss and Validation BLEU

Here we show an example plot of the training log for RNN with attention. We see that the training loss first drops significantly and then drops very slowly. One pattern we notice with the training loss plot is that it plateaus and then suddenly drops for several times, which indicate the loss surface might be made up with many flat regions, which are connected with cliffs. This pattern shows the difficulty of training our model because it is difficult for the optimizer to escape any flat regions. Furthermore, the validation BLEU score seems keeping climbing towards the end of the plot, which indicates that there are more potentials to the model's performance if we have time to train it for longer.

7.3 Sample Translations

Here we present some translation examples for Chinese to English. The predicted sentences are from RNN-based encoder and RNN-based decoder with attention.

Example 1

Reference: he just wanted to share a rainbow

Predicted: he 's just wanted to share a picture

Example 2

Reference: even cats were watching this video

Predicted: even a cat , you see the video

Through these examples, we see that the model seems to have at least some vague understanding of the source sentence. It can get some key words correct, but the syntax and the logic in the translation are not satisfactory yet. This pattern resembles that a person who just wakes up from a dream can only remember the outline of the dream but not the details. Indeed, in the RNN-based model used to generate these translations keep hidden states as their "memories". The problem is that while "memories" can keep note of important words and concepts, it is difficult to use them to recover the exact syntax and logic.

7.4 Impact of Sentence Length

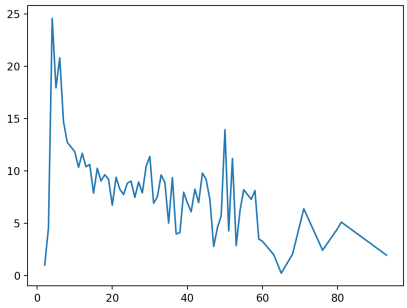

Figure 2: Performance by Sentence Length

Apart from computing BLEU for the entire testing corpus, we also computed BLEU for each sentence in the testing corpus. 2 shows that BLEU score peaks at short sentences and then keeps declining as the sentence become longer. Indeed, long sentences are very difficult not only for neural machine translation models but also for human beings. More specifically, for long sentences, it might be difficult to "memorize" the very long input sequence in the hidden state passed from the encoder to decoder. Also, a long sequence makes it hard for the attention mechanism to work as the longer the sequence, the more distractions appear when the model computes attention scores.

7.5 Performance of Self-Attention Encoder

The performance of our self-attention encoder with RNN-based decoder is disappointing. Although (Ashish Vaswani and Polosukhin, 2017) shows *Transformer*, which consists of a self-attention-based encoder and a self-attention-based decoder, delievers state of art results, our results show that self-attention encoder's performance is only on par with RNN-based encoder without attention in decoding.

We hypothesize that the reason is that self-attention encoder was designed to pair with self-attention-based decoder, which imposes a specific way to utilize the output from the encoder. On the contrary, as the RNN-based decoder expects a "hidden state" from the encoder, it is not straightforward as to how to extract the "hidden state" from the self-attention-based encoder.

8 Conclusion

In this project, we implemented several variations of the encoder-decoder framework for machine translation from scratch. Our results show that attention mechanism is great technique that significantly improves the performance in terms of BLEU score. Also, we found that neural-network-based Seq2Seq can be easily applied to different language pairs without altering the model architecture based on language-specific traits and properties.

For future works, more control variable experiments could be helpful to help us understand more about the encoder-decoder Seq2Seq framework. For example, we can answer is bidirectional GRU actually better? How many GRU layers are optimal? What about the number of hidden unit in each of the GRU layers?

9 Team Contribution

Name	NetID	Percentage Contributed
Jing Wu	jw3162	100%

References

Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Łukasz Kaiser Ashish Vaswani, Noam Shazeer and Illia Polosukhin. 2017. Attention is all you need. *arXiv:1706.03762*.

Oriol Vinyals Ilya Sutskever and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *NIPS Proceedings*.

Caglar Gulcehre Dzmitry Bahdanau Fethi Bougares Holger Schwenk Kyunghyun Cho, Bart van Merrienboer and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*.

Hieu Pham Minh-Thang Luong and Christopher D Manning. 2015. Effective approaches to attentionbased neural machine translations. *arXiv:1508.04025*.