A Deep Dive into Translation: 1011 Final Project

Yada Pruksachatkun (yp913), Vish Rajiv (vr1059), Zhihan Li (zl2516), Ziv Schwartz (zs1349)

Abstract

The task of this paper is to effectively build and train several types of neural translation machine models and evaluate using the Bilingual Evaluation Understudy (BLEU) Score. Machine translation is the problem of, given an input sequence in a source language, outputting a translation in the target sentence. We implement our models on two sets of language translation: Vietnamese to English and Chinese to English. Both sets of translations, Chinese to English and Vietnamese to English, were provided with a set of prepared corpora. The neural translation systems for the two language pairings we cover in this paper are (1) Recurrent neural network based encoder-decoder without attention (Vanilla Encoder Decoder), (2) Recurrent neural network based encoder decoder with global attention, and (3) Self-attention based encoder. Ultimately, we were unable to reproduce the results of the original papers with various experiments, although we were able to achieve a high BLEU score of 1.8 BLEU.

1 Related Work

The recent introduction of attention models by Vaswani et al (Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, 2017) have taken the machine translation world by storm, which focuses on introducing a new model architecture that entirely eliminates the need for recurrence and convolutions. The new model architecture, referred to as the Transformer, improved Vaswani, et al's previous BLEU score by over 2 points and established a state-of-the-art BLEU score of 41.8. The following paper, (Minh-Thang Luong, Hieu Pham, Christopher D. Manning, 2015), adapts the new attention mechanism model and implements it within a neural machine translation system, yielding an increase of 5 points in BLEU score. The self-attention encoder has thus been adapted to various domains including constituency parsing (Nikita Kitaev and Dan Klein, 2018).

Self-attention variants such as Werlen, et al (Lesly Miculicich Werlen*, Nikolaos Pappas, Dhananjay Ram, and Andrei Popescu-Belis, 2018) have also been introduced in the past year, which focused on the use of residual connections on residual RNNs.

2 Data Preprocessing

We trained on 133,317 Vietnamese-English pairs and 213,376 Chinese-English pairs, using the pre-tokenized preprocessed sentences provided, 1,261 validation pairs for Chinese-English and 1,268 for Vietnamese-English, and tested on 1,397 pairs for Chinese-English and 1,553 pairs for Vietnamese-English. We trimmed, removed and made the input sentences lowercase for English, removing noisy words such as "apos". For Vietnamese, we did not remove accents. The vocabulary sizes are 42,153 words for Vietnamese, 66,576 for Chinese, 50,970 English words (with Chinese) and 41,272 words for English (with Vietnamese). Each source sentence-translation pair are clipped at a max length of 20 words per source and target, and padded via batching.

Another preprocessing technique we experimented with was using ELMo embeddings instead of learning the embeddings as part of the model. However, we ultimately did not use ELMo in our experiments.

3 Methodology

We look to optimize machine translation on the following types of neural models, first using recurrent neural network based encoder decoder without attention, followed by recurrent neural network based encoder decoder with attention, and self-attention based encoder. We discuss our methodology below.

3.1 Vanilla Encoder Decoder

We built a Vanilla Encoder Decoder recurrent neural model without attention inspired by (Kyunghyun Cho, Dzmitry Bahdanau, Fethi Bougares, Yoshua Bengio, 2014). The Vanilla Encoder and Decoder are based on a sequence to sequence (Seq2Seq) framework. Following a classic RNN method:

$$h_t = tanh(A^\intercal h_{t-1} + U^\intercal e_t + b) \quad (1)$$

The encoder returns the final output h_T of the RNN, which is then fed into the decoder. The decoder is responsible for generating $p(y_t|y_{<t}, X)$, and similarly uses a GRU, where e_t is, when teacher-forcing, the target token. The first input is always SOS, and the first hidden state h_0 is the last hidden state from the encoder.

For our study, we implement the Decoder with ReLU activation layer after embedding and a GRU. The output is finally passed through two linear layers before a softmax activation layer for the output.

3.2 Adding Attention

For the attention model, we used the same RNN encoder with batching, while incorporating the Luong Attention mechanism to the decoder. The mechanism is as follows: the encoder outputs (the set of hidden states corresponding to each step in the source sequence) are passed to the decoder.

The decoder first runs through a recurrent timestep (GRU) (using a token from the target sequence) and outputs a decoder hidden state, which conceptually represents what has been translated *so far*. This hidden state and each of the encoder outputs are then passed into the Attention mechanism in order to score each of the encoder outputs for relevancy to the decoder's next word (to be translated). We use dot scoring as our attention function, which is shown below, using the query-key generalization. The decoder hidden state is represented as the query vector and the encoder outputs of a single source sentence are stored as columns in Q.

$$Score(Q, K) = Q^T \cdot K \quad (2)$$

We also implemented two other methods for attention scoring, as suggested by the paper: general and concat. The *general* method transforms each encoder output vector with a matrix multiplication, followed by a dot product with the decoder hidden state. The *concat* method concatenates the decoder hidden state with each encoder output, transforms each concatenated vector using a weight matrix, uses tanh activation function on the resulting tensor, and finally performs a dot product of this tensor with another vector v, a learned parameter for the model.

The output of the attention function results in a list of attention energies, across each of the encoder outputs for a given source sentence. We linearly combine these encoder outputs with their attention energies in order to get a final context vector, a representation of the most relevant *part* of the source sentence for the next word being translated.

In order to obtain our conditional distribution of the next word over the vocabulary, we concatenate this context vector and hidden state returned by the decoder, transform the result with a matrix multiplication over the vocabulary, and then softmax the resulting vector. This completes the forward pass through the Encoder and Luong Attention Decoder for a given source sentence. We based our implementation off of (pyt).

3.3 Self Attention Based Encoder

We then implemented and ran the full attention mechanism from (Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, 2017).

The main difference is that the attention energy distribution is calculated using the same sequence as the key K, value V, and query Q vectors when creating the context vector as outlined in 3.2. Furthermore, instead of a GRU implementation, a fully-connected feed forward layer is used to calculate the encoder hidden states. In our implementation, our self-attention encoder is paired with the standard no batching decoder. Within the Decoder, the inputs are passed through forward propagation, feeding input representations through an embedding layer and passing that embedding layer through an initial ReLU activation layer.

The input representations are passed through an EncoderLayer, a core Encoder, and finally a SupEncoder. The SupEncoder is a super class of Encoder, that learns embeddings and uses the encoder class during forward propagation to take in and process masked src sequences. The parameters are different from layer to layer although the

linear transformations are the same across different positions.

In order for the neural model to utilize the order of the sequence, some information is required regarding the relative and absolute positioning of the sequence tokens. This is achieved by incorporating positional encodings to the input embeddings before passing into the encoder. For this model, each dimension of the positional encoding corresponds to a sinusoid where the different frequencies are defined as:

$$PE(pos, 2i) = sin(pos/10000^{2i/d_k}) \quad (3)$$

$$PE(pos, 2i+1) = cos(pos/10000^{2i/d_k}) \quad (4)$$

As an input to the super Encoder class, the representation is passed through a layer made up of the self-attention, normalization layer and feed forward mechanism. This output is then used as an input within the core Encoder module, that is essentially a stack of N identical clone layers. We trained the model on a small subset of training set of 4 and 6 duplicate attention layers where each input are forward propagated through each layer in turn. However, the accuracy does not have a significant change. That may be because we only pass in one sequence at a time and the training size is relatively small when we are training the model with multiple attention layers. We choose our N to be 1 for the model on entire training set due to time constraints instead of the 6 layers specified in the paper due to time constraints.

The paper uses multihead attention, which uses h key, value, and query vectors to calculate attention. The attention is calculated across h equal parts of the hidden size d_k, dividing the hidden size into batches by number of heads h we defined to be 8. The separate and independent attention outputs are then concatenated and linearly transformed across all clone layers within the encoder.

Unlike Vaswani's model, since we did not batch our self-attention model, we took the mean of the second dimension, whose size is the number of the sequences, from the output of our encoder to pass into the vanilla decoder. This effectively squashes the R^{mxn} output into a R^m output, where N is the sequence length.

3.4 Full Self-Attention Translation System

To work with the EncoderDecoder framework for full connected self-attention encoder and decoder, we create training, validation and test dataset files that has different format from the previous dataset being processed. We implement the TranslationDataset class from torchtext package and create different vocabulary for source and target sequences from the training dataset. We tempt to run a small subset of 100 training pair samples and 20 validation pair samples. Further work could be focused on training on the entire dataset.

This is a translation system containing self-attention in both encoder and decoder while the decoder also incorporates attention over the encoder output. This model builds on the previous configuration of self-attention with $N = 6$ duplicate layers and incorporates new training and optimization functions. Instead of using negative log loss as the criterion, we use KL divergence utilized by the label smoothing step.

$$KL(P \parallel \hat{P}) = -\sum_{i=1}^{n} \log \frac{P_i}{\hat{P}_i} \quad (5)$$

Unlike the distribution for 0-1 target vectors, the smoothing class create a distribution over the confidence of words correctly predicted which improves the BLEU score.

The entire architecture for this system utilizes a joint EncoderDecoder module that serves as the baseline for both the Encoder and Decoder. The full model utilizes the same core Encoder and EncoderLayer from 3.3 and now implements this process for the Decoder as well. In addition, the searching method being used is not beam search but greedy search.

The core Decoder creates a generic N attention layers decoder with masking. We mask the tokens after what has already been translated because we do not want to use the future translation tokens to predict the current token translated. After computing attention based on masked target sequence, we connect the attention with input in SublayerConnection class and normalize the connected output. The connected self-attention outputs then corporate with encoder output to be passed in another multihead attention layer and feed forward layer. Both layers incorporate residual connection and normalization. The MultiHeadedAttention, SublayerConnection and LayerNorm method for the Decoder are implemented in the same manner as

the Encoder. Along with this, the PositionWiseFeedForward and PositionalEncoding methods are implemented within the Decoder as well. Our implementations are based off of the implementations of (Alexander Rush, 2018).

4 Experiments

For our experiments, we used a hidden size of 256, running for 4 epochs, calculating validation score with every batch iteration. We used the AdaDelta optimizer. We use beam search with K = 5 (exclusive, without including 1-4), and also set the maximum generation length to be equal to the length of the input sentence. We also experimented with learning rates from 3e-4 to 1, with surprising results, shown in our ablation study of the attention decoder. As clarification, we only preformed the high learning rate on the Luong attention decoder model, not the self-attention encoder model. All other models had a learning rate of 0.01.

4.1 Evaluation Metric

We used negative loss likelihood for our train loss, only accounting for the loss with respect to non-PAD tokens in our reference sentence in the batching process. For all our models, we trimmed the length of sentences to be 30, and used *sacrebleu's* raw corpus bleu function to calculate the validation and test accuracy. BLEU calculates the geometric mean of the precision of the predicted and reference sentences.

5 Results

For the Vietnamese to English translation, we were able to achieve a BLEU score was 1.6 and 1.2 for Chinese to English translation (see Table 1) with the Luong Attention Encoder model and a learning rate of 1.0.

5.1 Ablation study on learning rate

We experimented with the learning rate of the Luong attention decoder model, and trained our models over 4 epochs. Our results are in Table 2.

6 Takeaways

We learned a lot about how to more effectively use and debug PyTorch, especially on GPU with batching, and especially on models which train for relatively long periods of time (i.e. overnight), which was a first experience for us. We became very familiar with tensor operations and how to debug out-of-memory errors, which were caused by memory leaks in our code and were improved by garbage collecting and deleting references to variables. In retrospect, we also should have experimented other optimizers such as Adam. We also learned the importance of learning rate adjustment on final model performance. Moving forward, we will take these lessons to improve the effectiveness of our reproducibility work.

Model	**BLEU Score (Vi-En)**	**BLEU Score (Zh-En)**
Vanilla Encoder Decoder	0.7	0.5
Vanilla Encoder with Luong Decoder	1.6	1.2
Self-Attention Encoder with Vanilla Decoder	0.05	0.8

Table 1: BLEU Score performance on the testing data

Learning Rate	**BLEU Score (Vi-En)**	**BLEU Score (Zh-En)**
1.0	1.9	1.35
0.1	0.10	0.14
0.001	0.02	0.0009

Table 2: Maximum BLEU score achieved with respect to learning rate on validation dataset

7 Future Work

There is much more work to be done. Other than doing ablation studies on learning rate and hidden size on the models that we have run, we also would like to fully run some of the models we have implemented in our GitHub but not fully run on the entire dataset. These include the full self-attention translation model, Bahdanau attention decoder, and bidirectional GRU encoder, and the use of various preprocessing variants such as ELMo (Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer, 2018). We test the full self-attention translation model on the Vietnamese-English dataset with 200 training samples and 50 validation samples, the resulting BLEU score is 5.6, however, we would like to

run this model further for the entire dataset to report significant results. We also would like to explore the effect of various beam widths (for beam search) and number of cloned layers in attention on performance.

Furthermore, we would like to explore memory networks applied to the NMT problem. Cheng, et al (Jianpeng Cheng, Li Dong and Mirella Lapata, 2016) introduced memory networks in conjunction with LSTMs with respect to machine reading - this could be modified to enhance the performance for the machine translation problem.

8 Conclusion

In summary, the best neural translation model that we implemented was with the Luong Attention Encoder module with learning rate of 1.0 and Adadelta optimizer. After our ablation study, it was surprising to find that the learning rate of 1.0 ended up maximizing our BLEU scores. We would expect that with this type of task, a lower learning rate would perform better. Although we achieved a maximum of 5.6 BLEU score with the full attention model, we do not include that in our results section as we did not run it for as many epochs as the other models as to standardize our results. For the Vietnamese to English translation, the BLEU score on the testing set was 1.8 and 1.2 for the Chinese to English translation,

References

Seq2seq pytorch tutorial.

Alexander Rush. 2018. The annotated transformer.

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

Jianpeng Cheng, Li Dong and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *EMNLP 2016*.

Kyunghyun Cho, Dzmitry Bahdanau, Fethi Bougares, Yoshua Bengio. 2014. Learning phrase representations using rnn encoderdecoder for statistical machine translation. *EMNLP 2014*.

Lesly Miculicich Werlen*, Nikolaos Pappas, Dhananjay Ram, and Andrei Popescu-Belis. 2018. Self-attentive residual decoder for neural machine translation. *NAACL 2018*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. 2018. Deep contextualized word representations. *NAACL 2018*.

Minh-Thang Luong, Hieu Pham, Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. *arXiv preprint arXiv:805.01052*.

A Contributions

Yada served as the project manager for the group. She did literature review, preprocessed the data, set up the training/testing pipeline including batching, worked on and ran the vanilla encoder-decoder, the self-attention encoder, and bidirectional encoder experiments, and helped write the report, as well as collaborated with Vish on the ablation study. Vish worked on the vanilla encoder-decoder, batching the training pipeline, greedy and beam search, the Luong attention decoder, ablation study, and the report write-up. Eva worked on the self-attention encoder, fully connected translation system script and their write-ups, and Ziv worked on leading the report write-up and structure and compiled related papers on the models.
Master Code Repository:
`https://github.com/pruksmhc/FinalProject1011`