Machine translation project

Yifan Zhao
yifanzhao@umass.edu

1 Introduction

Machine translation aims to take one sequence of words in one language as input and output the same sequence of words in another language. Machine translation has many practical applications and plays a more important role in cultural exchange, education and scientific research with the development of internet and globalization.

In the past, several approaches have been proposed to tackle the issue of how to build an effective and reliable way to translate one language to another language by the computer. The most promising and accurate way is the statistical machine translation(Peter F.Brown, 1993) that learn a decent probabilistic model from the large source and the target corpus. But it is extremely hard to obtain a sound statistic machine translation system due to the complexity and irregularity of the alignment between two languages. Another drawback of this approach is that the system is divided into two separate components in nature since it is based on the Bayes rule and needs to calculate the product between the translation model and the language model. Thus, the system needs lots of feature engineering and even human resources to update and maintain.

Machine translation has made impressive progress since deep learning becomes a dominant approach in Artificial Intelligence. In recent years, neural machine translation(NMT) has been introduced into the research area of artificial intelligence and natural language processing and has achieved the state-of-art result in translation task. Different from the statistical machine translation which dominant in this task for the past several decades, neural machine translation is an end-to-end system that can learn the dependency and relation of two or more languages automatically through a large amount of data and expensive computation. But there do exist some disadvantages of the neural machine translation. It is difficult to adjust and control the whole training process except altering the hyperparameters of the defined model due to the lack of understanding and interpreting of the system.

In this project, we make use of existing neural machine translation model and translate the German sentences to English sentences. we will extend the seq2seq(Ilya Sutskever and Le, 2014) machine translation model to have a better performance in German to English translation, and use it as our baseline. Furthermore, we will apply the transformer(Ashish Vaswani, 2017) to get a better result in the machine translation task. The transformer is totally based on attention mechanisms and enjoying the benefit of recurrence and convolutions in the simple encoder-decoder neural networks. We are given IWSLT 2016 corpus as our default dataset and use BLEU metric to evaluate the result of our model.

2 What you proposed vs. what you accomplished

In our proposal, we propose to use the basic seq2seq model without the attention mechanism as our baseline model, and our approach will be the extension of the baseline model that includes the Bilinear attention and the beam search. But it seems that our method cannot reach a good result due to its insufficiency capacity to interpret and perceive subtle relations in the complex sentences. Thus, we decide to apply the transformer as our approach and the model indeed outperform the original seq2seq model with the attention. But their do have some goals that we could not achieve in the project due to lack of time and compute resources.

One is that we used to decide to modify the

basic transformer model that adds an additional bilinear attention layer between the encoder and decoder of our transformer model. In the basic transformer model, we have eight encoders and decoders respectively. We could make the eight output of our encoders to form an additional attention layer which may help the model find the indirect dependencies between the sentence pairs. The transformer is a self-attention model that each word in the specific sentence will look at other words in the sentence in the attention layer, but the multiplicative attention is attention between the encoder and the decoder. The combination of the two types of attention may get a good result in our translation task.

Another idea that we do not implement in our project is to use other state-of-art pre-trained word embeddings models like ELMO(Peters et al., 2018) or BERT(Devlin et al., 2018) as our corpus data embeddings and compare the performance of each word embeddings. In theory, it will give us a reasonable improvement in the machine translation result since these embeddings have been trained in the vast datasets and deep neural networks and are more representative to describe the words in the sentences.

3 Related work

In 2014, a seq2seq machine translation model was introduced and drawn attention in many NLP researchers. The seq2seq model is an end-to-end model that consists of two stacked LSTM networks: the encoder and the decoder. The encoder takes the input sequence as input and transforms them into a fixed-size vector which compresses the information about the input content. The decoder uses the vector produced by the encoder as a seed to generate the output sequence.

In order to get better translation results in long sentences, some attention mechanism approach(Minh-Thang Luong, 2015) is proposed to make use of the observation that different parts of the input sentence may have different weight when generating different parts of the output sentence. Thus, the decoder of our machine translation system can look up the source language information directly, and it allows the decoder to generate better prediction compared to the system that does not utilize the attention. It also helps our system to solve the gradient vanishing and exploding issue which is inevitable and harmful in the recurrent neural networks. Furthermore, the attention mechanism can assist us in interpreting our machine translation system by inspecting which part of the source sentence the decoder is focusing on each step. Another major issue in machine translation is large output vocabulary that may produce lots of unknown words. To address the problem, a paper(Minh-Thang Luong, 2016) suggests a hybrid model that make up with word and character-based approaches. The system translates common words using word-level approach and switches to the character-level approach when encountering rare words. Another mechanism that can improve the performance of our system is to use the beam search(Markus Freitag, 2017) instead of the greedy search. Different from the greedy search that produces the target sentence by taking the argmax word in the vocabulary on each step, the beam search keeps track of some potential partial candidates and generate the target sentence that has the maximum probability in the end. Although the beam search cannot guarantee to find the best target sentence, it has better performance than the greedy search and more efficient than the exhaustive search.

The performance of the machine translation system can be further improved by using the pre-trained word embeddings introduced recently. There has some word embedding algorithms used in the past like word2vec(Tomas Mikolov, 2013) or Glove(Jeffrey Pennington, 2014), but they do have several drawbacks. The word in these representations only has one single embedding regardless of in which context the word occurs or what specific meaning the word is supposed to convey. ELMO, which refers to embeddings from language models, learns the word embeddings by the relative context of the word instead of the fixed window size that used in the previous embedding algorithms. ELMO learns the context-specific word embeddings by combining and weighting various bidirectional LSTM word representations. Another famous pre-trained word embedding model is BERT, Bidirectional Encoder Representations from Transformers. In order to achieve a bidirectional understanding of the sentences but not feed future information of the sentence to the model, BERT utilizes the masking technique that predicts the mask words in each task.

4 Dataset

We use the provided IWSLT 2016 corpus as our dataset.The dataset consists of five different components: The German and the English training files and the German and English validation flies in addition to the test file. The training file has almost 200000 English-Germany sentence pairs and the validation file has more than 7800 language pairs. The test file includes around 2800 German sentence that supposed to be translated into English by our model. We do not touch the validation and test file in this section since it will be considered cheating.

4.1 Data preprocessing

We need to pre-processing the words in the description corpus before the process of word embeddings. Since there have many words in other language and uncommon words that are unlikely to appear in many situations, we use <unk>to denote the word that is not in the vocabulary and convert all the letters in words into lower case. We also adopt <pad>token to pad the sentence in order to make the sentences in one batch have the same length.

Since storing and processing strings are more difficult than dealing with the integers on the computer. All the words in the description corpus have been tokenized and initialized by a unique one-hot vector which only has numeric values in the original vector. We add two additional tokens that are annotated as <s>and </s>to our vocabulary, which represent the beginning and the end of our output word sequence. When passing these words to our model, we represent each word index using a one-hot vector and learning the parameters of the vector through backpropagation.

4.2 Data sample

There are sentences couples in training dataset that make our machine translation task hard, and our model cannot learn a lot in these pairs.

Some examples:

ENGLISH: 3,1415, 2657, 753, 8567, 24972 – – 85871, 25871, 3928, 5657, 2592,5624.

GERMAN: 3.1415, 2657, 753, 8567, 24972 – – 85871, 25871, 3928, 5657, 2592, 5624.

ENGLISH: 95.218.564?

GERMAN: 95.218.564?

The example sentence pairs above shows an identical match in English and German. Since digital numbers are considered rare words in our embedding, the model cannot get much information in this example and even earn some noise during the training process. Instead of applying word tokenization, we could make use of the character-based approaches as described in the related work section to solve the issue.

Another examples:

ENGLISH: David Gallo: This is Bill Lange. I'm Dave Gallo.

GERMAN: David Gallo: Das ist Bill Lange. Ich bin Dave Gallo.

ENGLISH: David Gallo: Audience: Seven. AB: Seven.

GERMAN: David Gallo: Publikum: 7. AB: 7.

The first case involves some kind of identical matching but other words need to be translated from German to English. The model is not supposed to learn the identical matching in the machine translation task without the copy mechanism(Stephen Merity, 2017). The second language pair has some flaws that the number '7' and the word seven imply the same thing but does not represent uniformly in this example.

5 Baselines

In our baseline model, we use the bidirectional LSTM(Sepp Hochreiter, 1997) as the encoder and unidirectional LSTM as the decoder, and we make use of the bilinear attention with beam search to gain better performance on our machine translation task. In fact, there have several attention variants we can utilize like scaled dot attention or additional attention, but they are more computational expansive than the simple bilinear attention approach and do not provide significant improvement in our baseline architecture(Denny Britz, 2017). The Figure 1 shows the baseline architecture.

We choose to use the embedding size of 256, hidden size of 256 and beam size of 5 to train our model since these parameters are safe and reasonable for machine translation tasks. In general,

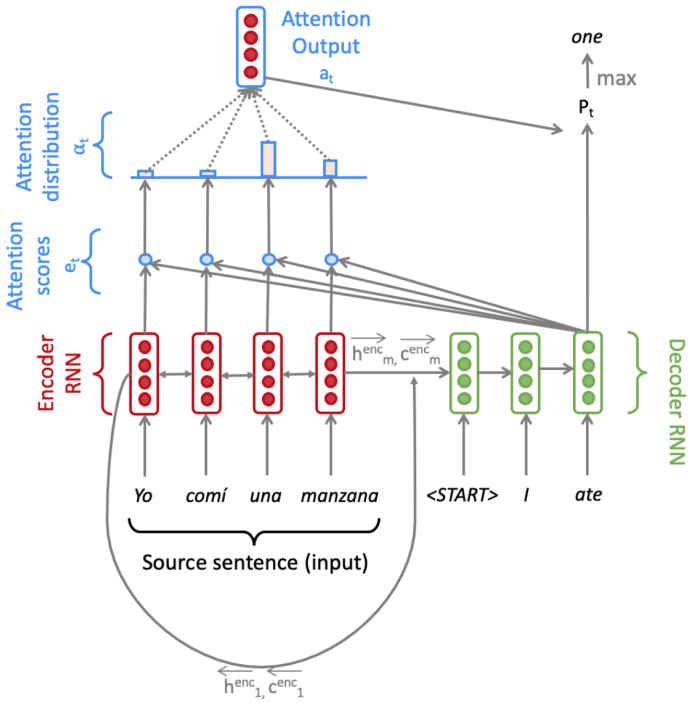


Figure 1: baseline architecture(adopted from cs224n ppt)

word embeddings have more capacity if the dimension increases and thus can help improve the accuracy of the translation, but we cannot use large embedding dimensions since it will cause Cuda memory error in the task.

5.1 Encoder

Our bidirectional Encoder has both forward(→) and backward(←) LSTM paths. Since every LSTM cell has to produce the hidden state vector, the final hidden state vector for the encoder hidden state in a specific step is the Concatenation of the hidden states in both directions. We also apply the same rule to generate the cell state vector for the encoder.

$$h_i^{en} = [h_i^{en}\leftarrow; h_i^{en}\rightarrow] \quad (1)$$

$$c_i^{en} = [c_i^{en}\leftarrow; c_i^{en}\rightarrow] \quad (2)$$

5.2 Dncoder

The hidden state vector and the cell state vector in the decoder are more difficult to generate than in the encoder due to the bilinear attention mechanism included. To begin with, we need to initialize the first hidden and cell state vector for the decoder by linking the final state of corresponding vector states in both directions.

$$h_0^{de} = [h_{final}^{en}\leftarrow; h_{final}^{en}\rightarrow] \quad (3)$$

$$c_0^{de} = [c_{final}^{en}\leftarrow; c_{final}^{en}\rightarrow] \quad (4)$$

The other hidden and cell state vectors are produced by the previous vectors and the word with attention in the particular LSTM step. The vector $\bar{y}_t$ is composed of the word embedding in the step and the output vector generated by the bilinear attention.

$$h_{de}^t, c_{de}^t = L(\bar{y}_t, h_{de}^{t-1}, h_{de}^{t-1}) \quad (5)$$

The result of the bilinear attention can be calculated over all the hidden state vectors in the encoder, and we finally cumulative vector indicating the relation between the target word and all the source words.

$$a_{t,i}^- = (h_{de}^t)^T W^{attn} h_{en}^i \quad (6)$$

$$a_t' = Softmax(a_{t,i}^-) \quad (7)$$

$$a_t = \sum_i^n a_t' h_{en}^i \quad (8)$$

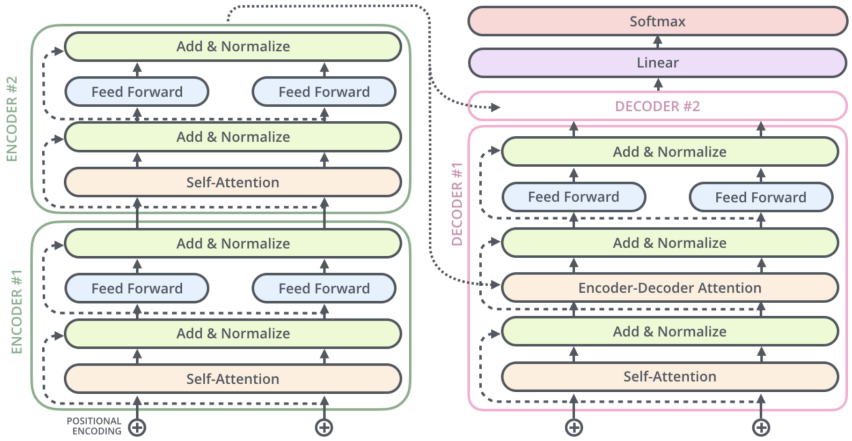


Figure 2: baseline architecture(adopted from Jay Alammar's blog)

In order to generate the output vector o_t that is concatenated with the next word embedding y_{t+1}, we need to combine the attention output a_t with the hidden state vector in the decoder. And we pass the concatenated vector into various regularization and generalization steps like dropout(Nitish Srivastava, 2014) and the tanh function.

$$u_t = [a_t; h_{de}^t] \tag{9}$$

$$o_t = D(Tanh(W_u u_t)) \tag{10}$$

6 Approach

Although the baseline model seems sophisticated and competitive, the transformer architecture attracts more and more attention among the natural language processing researches in recent years. We are curious about what performance can be achieved in the transformer architecture and we use the existing nlp framework allennlp(Gardner et al., 2017) to fulfill the implementation of the transformer model. The Figure 2 shows the transformer model architecture.

The transformer also has the encoder and the decoder like other machine translation model, but it makes use of the self-attention instead of the recurrent neural network for training and testing processes. Every encoder in the transformer model is composed of two components, the self-attention layer, and the feedforward layer. The decoder has an additional encoder-decoder layer between these two layers since it needs to get the information of the source sentence. We trained the transformer model for around 5 hours in the google cloud platform which provide a NVIDIA Tesla P100 gpu for training process.

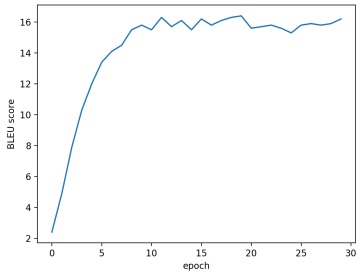
6.1 Self-attention

The key component in the transformer model is the self-attention mechanism. It helps the word in a specific position is aware of the other words in the input sentence. Thus it will get better encoded since it has a greater understanding of the whole sentence.

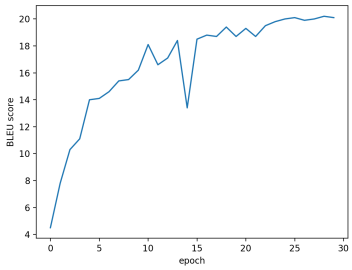
Each word in the self-attention process has three vectors, the key vector V_k, the query vector V_q and the value vector V_v. In each attention steps, every word takes the dot product of its query key with other words' key vector, and obtain an immediate score vector. And we scale the score by dividing the square root of its dimension to avoid large value dominant the probability distribution which leads to inappropriate gradient flows. We pass the scaled the score vector into softmax function and compute the product between the result of the softmax and the value vector to get the final attention vector that will be fed into the feedforward layer.

$$V_{atte} = softmax(\frac{V_q V_k^T}{\sqrt{d_k}})V_v \tag{11}$$

To archive parallelism and generalism in the training time, the transformer model introduces multi-heads attentions. Every word has multiple pairs of the three vectors and calculate each pair in the self-attention layer and concatenate the output vectors of these pair to obtain the eventual result vector. It expands the capacity of the model but

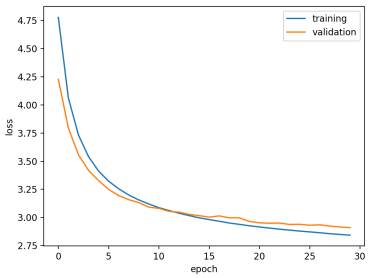


(a) baseline

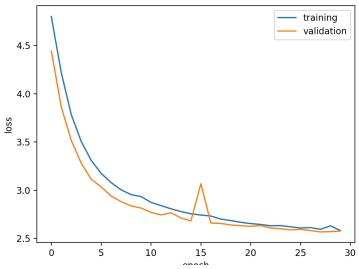


(b) transformer

Figure 3: BLEU score for baseline and transformer



(a) baseline



(b) transformer

Figure 4: loss for baseline and transformer

does not increases the computational time since they can be trained in parallel.

6.2 training techniques

Since the self-attention does not encode the position information of the source sentence other than the recurrent neural networks, the transformer needs additional positional encoding trick. The transformer model utilize the periodical functions like sin and cos. The embedding of each word has to add a position vector generated by the positional function based on their order in the sentence. These position vector produced by these functions can encode not only the position but the distance information.

Another technique is the residual network(Kaiming He, 2015) between each component in the encoder and decoder architecture. The residual network is very popular in the computer vision tasks and it can help alleviate the gradient flow issue in the deep neural networks. Since the transformer is a deep and large neural network, the residual network inserted between the encoders and decoders really relieve the issue of gradient vanishing.

6.3 result comparison

The Figure 3 shows the BLEU(Kishore Papineni, 2002) score achieved by each model and the Figure 4 shows the training loss and the validation loss during the training process. The baseline has the BLEU score of 16.6 and the transformer has the BLEU score of 20.2.

From the Figure 3, we can observe that the transformer model still learns from the training data since the BLEU score still increases, but the BLEU score of the baseline model does not increase after several epochs. We think the reason is that the self-attention mechanism in the transformer model is more subject to the internal relationship of the words in sentences and has better gradient flows in the training time. Another Figure shows that the baseline model will overfit the training data after a few epochs but this is not the case in the transformer model. In other words, the transformer model has more capacity than the baseline model.

But We also discover that the transformer model seems less stable than the baseline model since it has a sudden peak in the middle of the figures.

7 Error analysis

One typical issue I found in both my baseline and transformer predictions is that they have a large amount of words denoted <unk>.

Some examples:

GOLD: There are even research fields such as comparative religion and comparative literature.

baseline: There are even unk unk unk unk and unk unk .

transformer: There are even more unk like unk and unk unk .

GOLD: And my favorite, "Poor Gumby-mouth terrorist.

baseline: And my favorite , " unk unk unk .

transformer: And my favorite , " unk unk unk .

In our opinion, some reasons could lead to the issue. First, our model is not competent enough to learn the rare and compound words. The unknown words are usually professional nouns or advanced verb, our model could not interpret these high-level words during the decoding process. Another reason could be a lack of corresponding training data for our model to understand these words internally. And we can use the copy mechanism or character-based approach to alleviate the issue.

And there are cases that the transformer model performs better than the our baseline model.

Some examples:

GOLD: That's what psychologists call an "Aha!" moment.

baseline: That 's what 's called a unk call .

transformer: This is what 's called a psychologist moment .

GOLD: So what do you think happens when you pat a twentysomething on the head and you say, "You have 10 extra years to start your life"?

baseline: So what happens when you 're going to tell someone in your mind , and you say , " Do you get 10 years later , something about your life

transformer: So what happens when you put someone in the head of the head , and you say , " You get 10 years in order to make something from your life ? "

The results above show our transformer model has more capacity to understand and interpret words and can gain better translated sentences.

8 Conclusion

In this project, we have explored the traditional seq2seq model and the transformer model for machine translation task. Through the project, we have deepened the understanding of each machine translation model and gain some practice in applying deep learning to natural language processing.

For future work, we could implement the copy mechanism as described in the last section to improve the performance of our model. Another idea we want to try is to test the pre-trained word embeddings like ELMO and BERT and compare the capacity with the randomly initialized word embeddings.

References

Ashish Vaswani, Noam Shazeer, N. P. J. U. L. J. A. N. G. L. K. I. P. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Denny Britz, Anna Goldie, M.-T. L. Q. L. (2017). Massive exploration of neural machine translation architectures. In *Proc. of ACL*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. S. (2017). Allennlp: A deep semantic natural language processing platform.

Ilya Sutskever, O. V. and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*.

Jeffrey Pennington, Richard Socher, C. D. M. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*.

Kaiming He, Xiangyu Zhang, S. R. J. S. (2015). Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*.

Kishore Papineni, Salim Roukos, T. W. W.-J. Z. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceeding of ACL*.

Markus Freitag, Y. A.-O. (2017). Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*.

Minh-Thang Luong, C. D. M. (2016). Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proc. of ACL*.

Minh-Thang Luong, Hieu Pham, C. D. M. (2015). Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Nitish Srivastava, Geoffrey Hinton, A. K. I. S. R. S. (2014). Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*.

Peter F.Brown, Vincent J. Della Pietra, S. A. D. P. R. L. M. (1993). The mathematics of statistical machine translation: Parameter estimation. In *Computational Linguistics - Special issue on using large corpora: II*, pages 263–311.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

Sepp Hochreiter, J. S. (1997). Long short-term memory. In *Neural computation*.

Stephen Merity, Nitish Shirish Keskar, R. S. (2017). Regularizing and optimizing lstm language models. In *arXiv 1708.02182*.

Tomas Mikolov, Kai Chen, G. C. J. D. (2013). Efficient estimation of word representations in vector space. In *arXiv preprint arXiv:1301.3781*.