Neural Machine Translation

Report for NLP final project 2018 *

Minsi Long
ml5893@nyu.edu

Fu Shang
fs1520@nyu.edu

Abstract

This paper introduces several approaches for implementation of neural machine translator (NMT). We tried different structures of the NMT, including RNN based encoder-decoder, RNN encoder-Global attention-RNN decoder, and self-attention based encoder models. To boost the performance, we tried several different structures of the attention method, implemented loader with minibatch and built beam-search algorithm based predictor. We trained our model on vi-en and zh-en corpus. We got BLEU score of 18.52 with our vi-en model and 12.50 with our zh-en model.

1 Introduction

In this paper we implemented several language models to build a machine translator.
We introduce the data preprocessing, the models and the evaluation method in Section 2, 3 and 4 respectively. In Section 5, we present our results and make some discussion, for different models under different hyperparameter settings. And we present some examples of translations in Section 6. At last we summarize our results in Section 7.

2 Data preprocessing

2.1 Cleaning and Standardization

The dataset we were provided was already a cleaned parallel corpus, with most of useless punctuation removed, splitted into corresponding sentences in the original and target languages. For Chinese, the corpus use redundant split strategy to include all possible meanings of phrases. We splitted original sentences into lists of words, lower all letters and remove redundant spaces.

2.2 Language Class Object Building

For each language, we dump the corpus and all useful informations into a specific object defined by "Lang" class. We build vocabularies for words and tokens, and get count for words for both source and target languages. We also defined '<pad>', '<unk>', 'SOS' and 'EOS' tokens.

2.3 Data Loader

We built data loader class and vocabulary collect function. First, we convert our training language pairs into Numpy ndarray based on our pre defined max length, remaining validation and test as its original length. Then we build data loader objects for train, validation and test. For each batch read, we pad all the sentences to the length of the longest sentence in that batch following the end of sentence, both in original and target language.
In training, we use batch_size=64 for all cases. In validation and test, we use fixed batch_size=1.

2.4 Pre-trained Embedding

To accelerate the convergence, we used Facebook fasttext pre-trained embedding for both VI, ZH and EN languages (Bojanowski et al., 2016). The embedding was trained on Wikipedia.
We collect embedding vectors for top 100000 most frequent words in both languages, build word-vector

*All sourcecodes could be found in our github repo: https://github.com/firstkeaster/DS_GA_1011NLP_Project/ Don't forget the underlines when copying the link!

dictionary, and build embedding matrix by comparing words in our Lang object with the pre-trained embedding dictionary. For words never appeared in pre-trained embedding, we initialize its vector with "0"s, and train it later in our model.

3 Models

The translation system is a kind of conditional auto-regressive model. Generally speaking, the translation model consists of two parts, encoder and decoder. The encoder compresses a sentence in source language into a vector (vectors) with fixed length. The decoder takes the outputs of encoder to generate the translation sentence in target language.
For encoder part, we tried bidirectional GRU-network and self-attention mechanism. For decoder part, we tried GRU-network and GRU-network with attention.

3.1 RNN based encoder-decoder

We built bidirectional GRU-Encoder and two layers GRU-Decoder models for the translation tasks. The encoder part compresses a input sentence into a hidden vector. The decoder GRU network takes the vector as the initial hidden vector and to generate the corresponding translation.

3.2 RNN based encoder-decoder with attention

We built GRU-Encoder, Attention, GRU-Decoder models for both ZH-EN and VI-EN tasks. The main idea is to build a attention mechanism to evaluate correlation between each encoder output and some specific decoder output, and get a weighted sum of all encoder outputs as additional information vector for generating the output of current decoder unit. Based on former works (Luong et al., 2015)(Bahdanau et al., 2014), we built a NN structure as shown in Figure(1).

One significant specification of our model is that for decoder, we feed encoded vector into GRU directly, and use the output hidden as the "query" for attention layer. This makes the decoder hidden provide highly homogeneous information with the encoder hiddens. After calculation of the weight and weighted sum of all encoder hiddens, we feed the concat of "weighted sum" and decoder hidden, and feed it into linears and softmax to get probability distribution of output index.

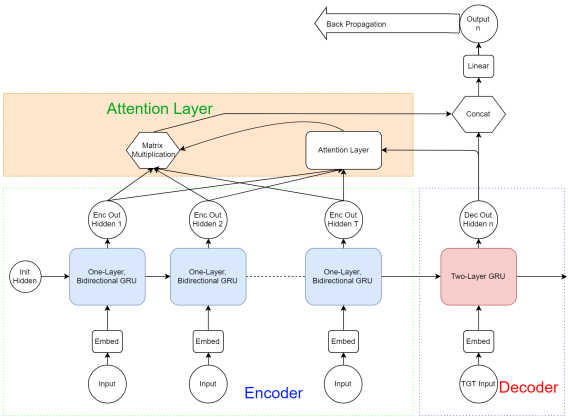

Figure 1: Attention Model Concept

For the calculation of attention score, we tried three different approaches:

$$score(h_e,h_d)=\begin{cases} dot(h_e,h_d) & Dot \\ dot(h_e,W_a h_d) & Dot-Linear \\ dot(v_a,tanh(W_a[h_e,h_d]) & Concat(withParams) \end{cases}$$

In which h_e, h_d are hidden state of encoder decoder, W_a is a linear layer, and v_a is a parameter vector.
We used bidirectional GRU as the recurrent unit in encoder, and two-layer unidirectional GRU as decoder. We initialize first_hidden of encoder with "0"s, first input of decoder with a "SOS" token, and use last_hidden of encoder as first_hidden for decoder.
We defined early-termination, comparing current val-Bleu with the historical highest Bleu minus a margin of 3.

3.3 Self-attention encoder and RNN-based decoder with attention

We built self-attntion encoder and RNN-based decoder with attention. The main idea of self-attention is similar to the attention mechanism. But sefl-attention mechanism evaluates the correlations between the input sentence words themselves. More information is in (Vaswani et al., 2017).

4 Evaluation

4.1 Evaluator

The evaluator we used in validation process of training is based on Greedy Search. For each output probability vector, we get index for only the max-probability item in the vector, then feed its embed-

ded vector into the decoder, and thus form the output sentence all the way down to a 'EOS' was predicted.

4.2 Beam Search Evaluator

We built a Beam Search evaluator to get better predictions from the trained model, compared with Greedy Search.

The basic idea of our algorithm is to use two Stacks to record "Current" K-highest-probability sequences (CUR) and "Already Ended" highest-probability sequences (END) separately, with tuple items in the form of (Probability, [Sequence]). For each predicted probability vector, we keep K highest probability token-indexes in our cur_ind stack, attach them to its corresponding preceding sequence respectively to get a K*K CUR stack, sort the current stack based on [0]th item "Probability", and keep the K-highest in CUR to be inherited to next step. Once a sequence reaches 'EOS' and still in stack, we move it to our End stack. For the next step, we simply take last token-index, feed it into decoder, and repeat. Once we finished all sequenses, we return the highest-probability sequence in the END stack, and form it back to a sentence as our output.

4.3 Loss Function and Bleu Score

We calculate NLLLoss in both training and validation set. We use raw_corpus_bleu function from sacrebleu lib. When forming references and candidates, we use just the original validation and test set WITHOUT removing any punctuations, cut off or exclude any long sentences, or add 'SOS' or 'EOS' to the sentences. Our candidate sentences are also in this form, with all '<pad>', '<unk>', 'SOS' and 'EOS' removed.

5 Result

We used Adam as optimizer, and NLLLoss as cost function in training of our models.

5.1 RNN based encoder-decoder

We tried different hidden sizes for translation tasks ZH-EN and VI-EN respectively.

The initial learning rates are $lr = 0.001$ and $lr = 0.0005$ for ZH-EN and VI-EN respectively. The learning rates decay with epochs exponentially.

The result is presented in Table 1. The training curves are in Figure 2 and 3.

ZH-EN		
hidden size	256	384
Bleu score	2.5999	5.2943
VI-EN		
hidden size	128	256
Bleu score	3.652	3.713

Table 1: Result for RNN encoder-decoder

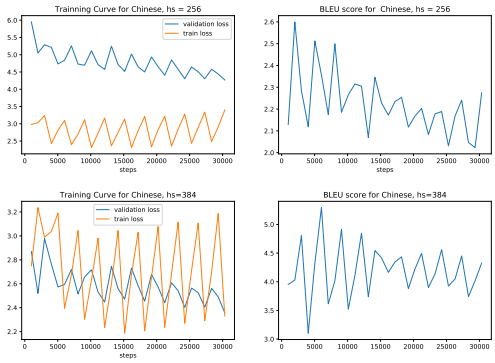

Figure 2: Training Curve of RNN-encoder-decoder for ZH-EN

5.2 Result of Encoder-Attention-Decoder Models

For learning rates and teacher forcing ratio, we tried both fixed and epoch-decay step size and ratio. The best results we got are from fixed learning rate of $3e-4$, and fixed teacher forcing ratio of 0.9.

We tried different number of epochs in training, vary from 10 to 30. Most of our best models in each training come from the epoch around 10. In those cases, the model starts to overfit after 10 epochs' training. We could reduce training NLL to lower than 1, but validation Bleu score continues increasing in following epochs.

We explored with both embedding size and hidden size during hyperparameter tuning for each model structure. For hidden size, We explored from 225 to 384 for VI-EN task, and from 225 to 1152 for ZH-EN task.

As shown in Figure(4), although the model converges faster with a larger hidden size, it overfits very fast and thus cannot reach a satisfactory validation Bleu score after all. Moreover, the validation loss of hidden_size 768 model shows a very strange pattern. We'll try to figure the reason out in future researches.

In simple Dot model, we subdivided hidden size and

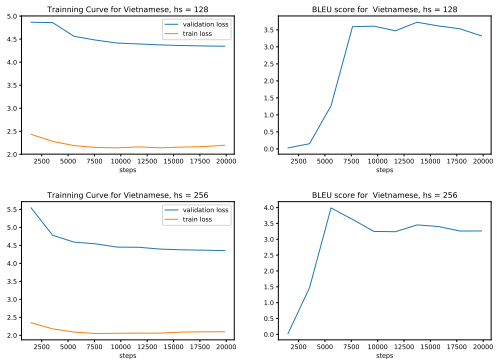

Figure 3: Training Curve of RNN-encoder-decoder for VI-EN

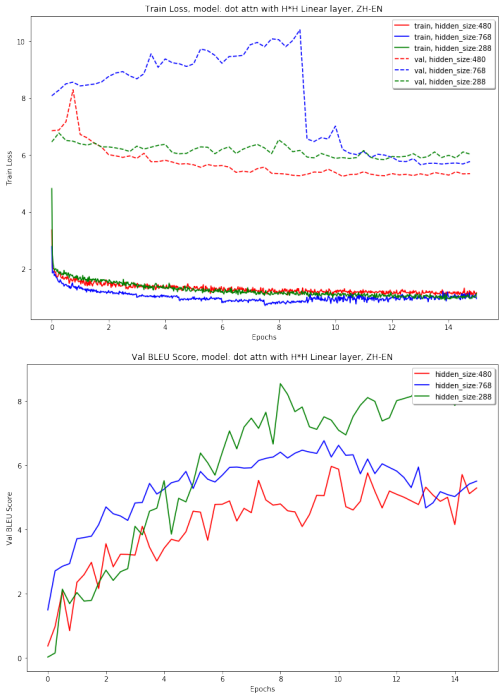

Figure 4: Training Curve of Dot-Linear Model for ZH-EN

got a eclectic hidden size of 384 which gives us the best model Figure(5). We also found NLL loss for validation set is not very relevant to the Bleu score, which suggested us to use Perplexity in the future.

To boost the performance, we tried all three model structures. Comparation between best model for VI-EN are shown in Figure(6).

Simple Dot model shows very similar performance compared with Dot-Linear model. Both models overfitted after 10 epochs' training. However, the concatenate attention model performed bad in this task.

Since overall best model for both ZH-EN and VI-

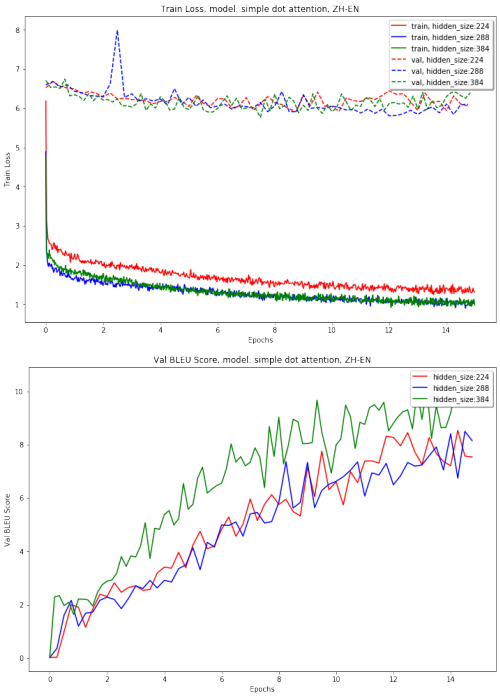

Figure 5: Training Curve of Simple Dot Model for ZH-EN

EN tasks come from the Encoder-Attention-Decoder model type, we'll report the details about those models later in 4th section of this chapter.

5.3 Result of Self Attention Encoder Attention Decoder Models

The self attention encoder models don't produce "hidden vector" directly. We took the last two vectors of the output of encoder, as the hidden vector to feed into the decoder. The dimension of the output of encoder is equivalent to the hidden size of the decoder. As referred in (Vaswani et al., 2017), we set the hyperparameters of encoder as: the encoder layers $N = 6$, each layer has 8 heads. The inner-layer of the feed forward function between layers of encoder has dimensionality as $dff = 2048$.

The learning rate starts from 0.0001 and decays with epochs exponentially. The decay rate is 1.05. The results are presented in Tabel 2. The training curves are in Fig 7 and Fig 8.

5.4 Overall Best Models

For VI-EN and ZH-EN tasks, both best models are from the "Encoder-Attention-Decoder" structure. Our best model's hidden size, training epochs,

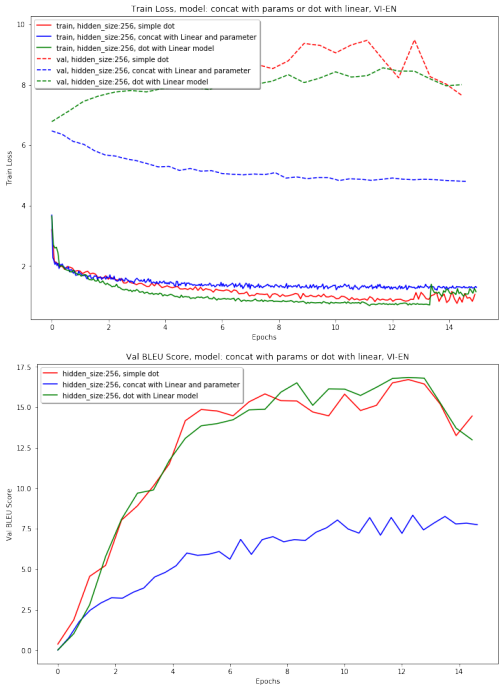

Figure 6: Training Curve of Multiple Models for VI-EN

ZH-EN		
hidden size	512	720
Bleu score	5.5338	5.1712
VI-EN		
hidden size	512	720
Bleu score	3.2241	4.4482

Table 2: Result for self-attention encoder-decoder

validation Bleu, test Bleu, test Bleu with beam search are presented in Tabel 3.

VI-EN					
Model	Hid	EP	VBleu	TBleu	TBleuBS
DotLinear	256	7	16.84	16.42	18.05
ZH-EN					
Model	Hid	EP	VBleu	TBleu	TBleuBS
SimpleDot	384	17	10.71	10.51	11.97

Table 3: Result for Overall Best Models

6 Translation Presentation

Respectively, we list some translation examples for both ZH-EN and VI-EN.

First we present 6 examples for ZH-EN. First we take a look at 3 short sentences. The translations are very close to the reference sentences. The translations of the 6 long sentences contain the key words

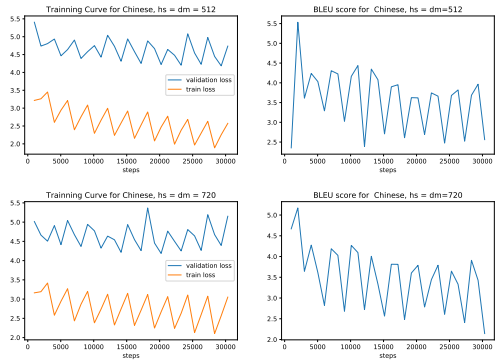

Figure 7: Training Curve of self-attention encoder for ZH-EN

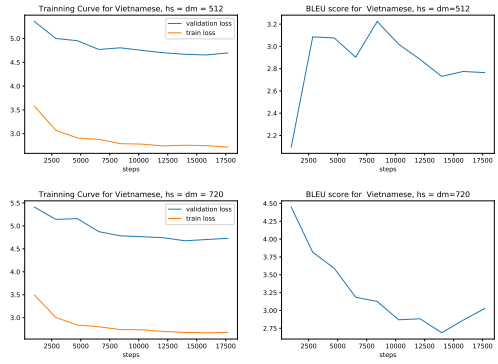

Figure 8: Training Curve of self-attention encoder for VI-EN

but are not as fluent as the reference sentences. And the translations of long sentences look like missing some words at the end.

- (**Z**) 我们正在起飞
  (**T**) we're taking off .
  (**R**) We're taking off .
- (**Z**) 我妈妈是十个个里最年轻的
  (**T**) my mom was 10
  (**R**) My mom was the youngest of her 10 kids.
- (**Z**) 我永远忘不掉不掉
  (**T**) i'll never forget that.
  (**R**) I never will forget it.
- (**Z**) 大概十五还是二十二十分二十分钟十分十分钟分钟之后她站起起来穿过过房房间牵住我的手对我说过来布莱恩莱恩我们得谈谈
  (**T**) it's about 15 minutes later , and after 10 minutes , she stood up through the room , and i said , "brian , we've got to
  (**R**) And after about 15 or 20 minutes of this ,

she got up and she came across the room and she took me by the hand and she said , "Come on , Bryan . You and I are going to have a talk."

- More examples are in Appendix.

Now we present 6 examples of VI-EN. For both long and short sentences, the translations are better than ZH-EN. This may comes from the Vietnamese vocabulary size (35530) is smaller than Chinese vocabulary size (88426). The model can fit better.

- (**V**) Cám_ơn rất nhiều
  (**T**) thank you very much .
  (**R**) Thank you very much .

- (**V**) Bây_giờ , nghĩ về những sự lựa_chọn của chính bạn
  (**T**) now , think about your own choices .
  (**R**) Think about your own choices .

- (**V**) Đây là cửa_hàng với tên Drager ' s
  (**T**) this is the store with the name of the " star."
  (**R**) It was a store called Draeger's .

- (**V**) Gần đây , tôi đã khảo_sát với hơn 2.000 người Mỹ , và trung_bình số lựa_chọn mà người châu mỹ điển_hình đã làm là khoảng 70 lần trong 1 ngày
  (**T**) recently , i've been investigating over 2,000 americans , and the average number of choices that the american americans have done is about 70 times a day .
  (**R**) I recently did a survey with over 2,000 Americans , and the average number of choices that the typical American reports making is about 70 in a typical day .

- More examples are in Appendix.

7 Conclusion and Perspective

We tried three different structures and several variants of neural Seq2Seq language model. We performed hyperparameter tuning on multiple parameters and tried different training strategies.
A fixed learning rate and a very high teacher forcing ratio seems mandatory for a good model. The Seq2Sqe models are very sensitive to hidden size, embedding size and other parameters, high volume of tuning work is needed in every specific task.
We can get satisfactory VI-EN translations with our model, the translations for long sentences are impressive. Our ZH-EN model is not that robust in handling sources with various semantics.
We should try more changes and innovations with the structures of attention models referring the newest discoveries.

Contributions of team members

Mingsi Long is mainly responsible for building data preprocessing, RNN encoder-decoder models and self attention models. Fu Shang is mainly responsible for building evaluation and encoder-attention-decoder model. We two team members worked closely together in debugging, training and tuning models.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Appendix

More translated exmpales in Chinese-English translation task done by our best model:

- (**Z**) 这上面说本飞机有很多精密部件协作保障你飞行安全这是不是不是让你很有信心
  (**T**) it says, well, ben plane has a lot of sophisticated parts, collaborative security, and you can fly, and you
  (**R**) It says, 'This plane has hundreds of thousands of tiny parts working together to make

you a safe flight.' Doesn't that make you feel confident?"

- (**Z**) 我们已经体会到作为一个老师你的话是有影响影响力的如果你是个很慈善的老师你的教导就格外意味意味深长深长
  (**T**) we've experienced that as a teacher , you have a influence , and if you're a great teacher , you teach it , you
  (**R**) And I think what we've learned is that , if you're a teacher your words can be meaningful , but if you're a compassionate teacher , they can be especially meaningful .

More translated examples in Vietnamese-English translation task done by our best model:

- (**V**) Cười Nếu bạn nghĩ đến phim "Avata "nếu bạn nghĩ về việc mọi nó đã lay_động mọi người như thế_nào - - đừng để_ý đến câu_chuyện Pocahontas làm_gì - - Tại_sao lại bị lay_động bởi biểu_tượng đến$_th$
  (**T**) if you think about film , " if you think about things that have moved people like – don't pay attention to the story ?
  (**R**) And if you think about " Avatar " if you think of why people were so touched by it – never mind the Pocahontas story – why so touched by the imagery ?

- (**V**) Và những nhà khoa_học đã dẫn ra tất_cả những nhiệm_vụ khác_nhau mà những CEO đó tham_gia và họ đã tốn bao nhiều thời_gian để thực_hiện những quyết_định liên qua đến những nhiệm_vụ đó
  (**T**) and the scientists have led all sorts of different tasks that ceos have been involved and they've spent a lot of time to do these decisions.
  (**R**) And these scientists simply documented all the various tasks that these CEOs engaged in and how much time they spent engaging in making decisions related to these tasks .