Neural Machine Translation, English to Mandarin

Tianye Wang
Dalhousie University
Halifax, NS
tn610582@dal.ca

Zhiran Wang
Dalhousie University
Halifax, NS
zh947847@dal.ca

ABSTRACT

In this project, we use Tensor-flow sequence-to-sequence(seq2seq) model to build a neural machine translation(NMT) tool to translate a sentence from English to Mandarin. The seq2seq model contains two Recurrent Neural Network (RNN), each for encoding and decoding. To improve the performance of the model, batching, padding and dropout techniques are used in the program. The training dataset is a translated TED presentation from IWSLT. The result shows that the overall translate accuracy is below 50%, because the limit size of the training data. However, we have found that shorter sentences has higher translate accuracy.

1 INTRODUCTION

The research of machine translation starts from the '50s. Machine translation is one of the earliest research areas in AI. At that time, the main method to translate between languages relies on handcraft translation rules. However, researchers realize that the complexity of a language is hard to cover by the translation rules. Later on, people use statistic approach to translate language.

N-gram have been used in language character identification and many accurate methods were developed for long text samples, but the identification for short phrase still needs improvement. Tommi, Jaakko and Sami studied the language identification task with the test sample length between 5 to 21 characters[1]. David and Mitchell studied an algorithm that used to characterize a constituent boundary parsing by using generalized mutual information. It is a standard approach to parsing natural language by using the basic language rule, grammar[2].

From 2010, the RNN approach to NLP problems starts to outperform the statistic based model. Unlike the statistic model, the recurrent neural network can store any length of the previous words or information. The deep neural network has succeeded in character identification and in Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation[3], they prosed a new model called RNN Encoder-Decoder and it consists two recurrent neural networks. One RNN used for encodes input symbols into a fixed length vector representation, the other one is used for decodes the vector representation into the target language.

The recurrent neural network is popular for language modelling but it can only exploit a fixed length context to predict the next word of sequence which means RNNs are difficult to train[12]. LSTM language model is used to address this problem with RNN and enable the full potential of recurrent models. RNNs with long short term memory (LSTM) can provide the best result in handwriting recognition. A paper called Dropout improves Recurrent Neural Networks for Handwriting Recognition, the team discussed that the performance can be greatly improved by using dropout. In their approach, dropout is used in the network to not affect the recurrent connection, which means the potential of RNN was not affected[4]. In this project,

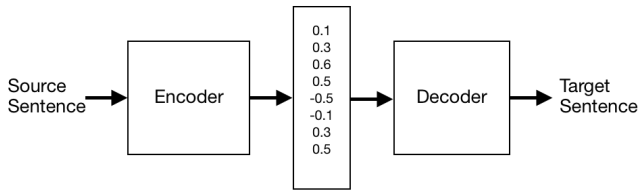

Figure 1: Encoder-decoder model - encoder convert a source sentence to a information vector and pass to the decoder to produce a translation.

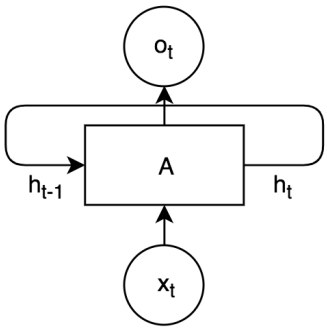

Figure 2: Example of RNN Structure

we will use Tensorflow's API to build a sequence-to-sequence language model to translate English to Mandarin.

2 BACKGROUND

2.1 Neural Machine Translation

In the traditional translation system, a model requires many steps such as preprocessing, word alignment, phrase alignment, extract phrase features, training model and weight learning. After breaking the sentence to chunks the system translates them phrase by phrase, that is called a statistical phrase-based translation system[5]. Compare to the statistical model, the concept of Neural Machine Translation(NMT) system is straightforward. In addition, studies have shown the NMT approach on the translation task obtain good results and often show better results than other states of the art statistical language modes[6]. Especially when facing the issue of local translation, NMT able to capture long-range dependencies in languages such as gender agreements[7]. An NMT system first uses an RNN reads the input sentence, converts the information about the sentence to a sequence of numbers; then, another RNN read this sequence of number and generate the target sentence. The two RNN are called Encoder and Decoder. An example of the encoder-decoder model is shown in Figure 1.

2.2 Recurrent Neural Networks

Recurrent Neural Networks(RNN) is one of the building blocks of an NMT model. RNN originated from Hopfield Networks by Saratha Sathasivam in 1982. Hopfield Networks is hard to implement in the '80s, so it was replaced by other traditional neural networks. With more effective RNNs being proposed, the sequence processing power of RNNs is being fully utilized[8]. Different from other neural networks RNN have connections between hidden layers. The hidden layer not only include the input but also the output from the last hidden layer. When the current words are 'sky', the previous words are 'The color of the', then there is a higher probability then the next word is blue. RNN emphasis the connection between current state and previous state. Figure 1 shows an example of an RNN structure. At every time t, RNN A will combing the current input x_t and previous hidden state h_{t-1} to generate current state h_t and output o_t.

RNN use saved history information to help prediction in the current state, like use the previous word to help to improve the understanding of the current prediction. RNN can take advantage of extracting information that the traditional neural network cannot obtain. However, RNN brings a larger challenge which is long-term dependencies. For example when predicting short sentence'The sky is blue' traditional RNN model is good enough to predict the last word 'blue' by looking at the

previous word 'sky'. On the other hand, if the sentence is 'A new car factory opened last year in Halifax, the air pollution is.....the color of the sky is grey.', when predicting the last word 'grey' short term dependency will not predict the word right. The word can be blue or grey. RNN performs badly when facing long term dependency. Long short-term memory(LSTM) is designed to solve this kind of problem in RNN. LSTM was proposed by Hochreiter and Schmidhuber in 1997[9], it is a special type of gated RNN. LSTM contains three gates called input gate output gate and forget gate. The forget gate will let the RNN delete the previous useless information and the input gate will let new information to be 'memorized' in the network. Finally, the output gate will decide what to output based on the current state and the previous output[LSTM].

2.3 Sequence-to-sequence model

Before we get into the seq2seq model, we need to introduce an important concepts of the model called embedding. In the embedding layer of the network, every word are represented by a real number vector, this vector is called word embedding. Word embedding can be understand as convert the vocabulary to a fixed dimension space. The main purpose of converting the input words into word embedding is to decrease the dimension of the input. If we do not use word embedding the input words will be pass to the RNN as one vector with size equals to the number of vocabulary. A vocabulary usually contains over 10000 words. However, after word embedding the size of input vector is between 200 1000. Another advantage of word embedding is to increase semantic information. A simple vocabulary mapping does not contain any semantic information of two adjacent words. However, word embedding convert the sparse vocabulary into a more dense vector. For example, after training with information containing dog eats food and cats eats food, the value of dog and cat after word embedding will be close to each other[10].

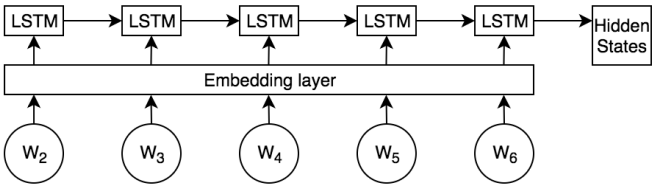

Figure 3: Seq2seq model - encoder.

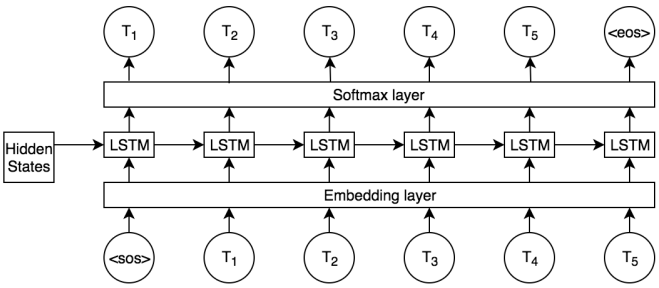

Figure 4: Seq2seq model - decoder.

The seq2seq model is similar to the NMT system we discussed before. It has a encoder and a decoder. However, the structure of the encoder and the decoder are different than the NMT system. The encoder of seq2seq have two layers, the first layer is the embedding layer, and the second layer is a LSTM RNN. The decoder of seq2seq model also have a embedding and a LSTM layer, the different part is that the encoder have a softmax layer to generate the probability of the words in the output.

In the training process, the encoder read the input sequence one by one, then pass the hidden layer state of the LSTM to the decoder as the input sequence. The first input sequence of the decoder is a special character called '$< sos >$' means the start of the sentence. The words predicted in each step is the target language output. Finally, the last words of the predicted sequence is another special character called '$< eos >$' means the end of the sentence. Figure 2 is an example of a seq2seq encoder model. Figure 3 is the decoder of the seq2seq model.

2.4 Perplexity, log perplexity and cross entropy

The common evaluation of a language model is perplexity. The lower the value of perplexity the better the language model is. Equation (1) shows how to calculate the perplexity of a sentence S.

$$\begin{aligned} perplexity(S) &= p(w_1, w_2, w_3, ..., w_m)^{1/m} \\ &= \sqrt[m]{\frac{1}{p(w_1, w_2, w_3, ..., w_m)}} \\ &= \sqrt[m]{\prod_{i=1}^{m} \frac{1}{p(w_1, w_2, w_3, ..., w_m)}} \end{aligned} \tag{1}$$

In other words, perplexity measure characterizes the ability to predict a language sample in a language model. For example, if we know that a sentence $S(w_1, w_2, w_3, w_4)$ will be in our testing dataset, the higher the probability of this sentence we get from the model the better our model fits the dataset. From equation (1), we can observe that perplexity can be considered as a weighted branching factor or average branching factor. Perplexity is the number of choice of the next text predicted from the model. For example, if a model has perplexity of 10, that means when the model predicts the next word there are 10 words you can choose from. When training a language model, log perplexity is preferred. Equation (2) shows how the log perplexity is calculated. Compare equation (1) to equation (2), rather than taking square root and multiplication in equation (1), taking sum can speed up the calculation process in equation (2).

$$log(perplexity(S)) = -\frac{1}{m} \sum_{i=1}^{m} log(w_i | w_1, w_2, ...w_{i-1}) \tag{2}$$

In statistics, log perplexity can be treated as the cross-entropy between a predicted distribution of the dataset and the ground truth of the dataset. Cross-entropy describe the distance between two probability distribution. Assume $u(x)$ and $v(x)$ are two probability distribution of x, then the cross-entropy H of u and v is defined as the expected value of $-log(v(x))$ of distribution u in equation (3) [12].

$$H(u, v) = E_u[-logv(x)] = -\sum_{x} u(x) logv(x)) \tag{3}$$

2.5 Batching

In order to process a large volume of data efficiently, batch processing was implemented. Unlike the real-time processing, which will process the data right after it entered into the machine, batch processing collects all the data and data related information from the transaction and store everything into a group. When the group of data reached a preset size, it will be processed all at once. In this project, the training data was not randomly selected sentence pairs, they are related to each other in the paragraph and in the article. The machine must be able to pass the message to the next sentence in order to make the whole translation more accurate. If there is not limited to the size of the model, the best option is to connect all the sentence in the article and train as one big sentence. Unfortunately, it is not possible because entering a whole article into the training model would cost gradient explosion and occupies too much memory resources. The solution to this problem is to slicing the one big sentence into several sub-sentence with the same fixed length. Every time the machine finished process a sub-sentence, the final stats will be carried to the initial value of next sub-sentence[13].

2.6 Padding

In text data processing, the length of each sentence is different and unlike images, it is unable to re-size all the sentences to the same size. The most common way to address this problem is padding. In the machine translate training data, each pair of a sentence were trained as independent data. Because of the length of each sentence was different, when all these sentences being entered into the

same batch, it is necessary to extend the length of shorter sentence to the longest in the batch. This process is called padding. For example, if there are two sentences in the batch, $A_1A_2A_3A_4$, and B_1B_2. After padding, the data will become $A_1A_2A_3A_4$ and B_1B_200[14].

2.7 Dropout

Dropout is a system and method technique to address and reduce overfitting problem in neural networks. When a learning machine has an overfitting problem, its function usually fit a limited set of data points. In reality, statistics and data study always have some error or random noise within it, and people usually drop those noise data to make the result more accurate. Similar to dropping noise data in statistics, machine learning model also drops data sets that too far away from the function. Figure 5 shows two sets of data and try to use a curve to separate them. Although it didn't completely separate blue and red dots, it is the line of best fit. The green line shows when overfitting happens, the line forced to try to separate all the dots. As a result, it successfully separates the data but the line itself means nothing to the chart and unable to fit additional data or predict future observation reliably. Therefore in this project, it is important to apply dropout to ensure the performance of translate machine and accuracy[15].

3 EXPERIMENT SETTING UP

The experiment are developed in Python 3.6.7. We use codecs and collection package to preprocess the dataset and everything else are in Tensorflow 1.14.

3.1 Dataset

The training dataset used in this project comes from International Workshop on Spoken Language Translation(IWSLT). IWSLT is an organization associated with an open evaluation campaign on spoken language translation, they provide many

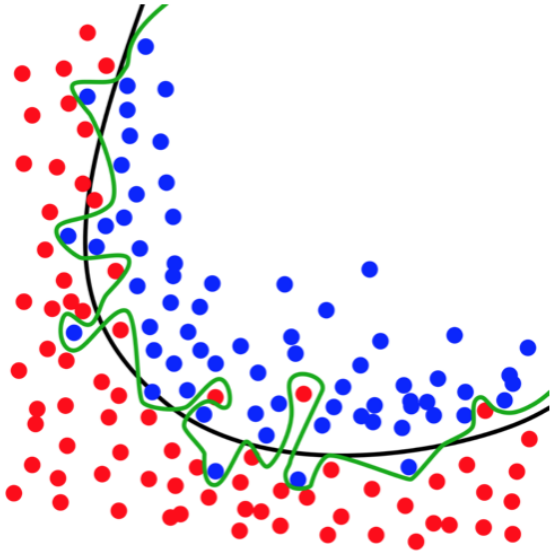

Figure 5: Overfitting Example[16].

training data with many languages. The reason this project didn't use datasets from Workshop on Statistical Machine Translation, WMT for short, is because the size of dataset from WMT are huge and took too much time for training. The dataset used in this project contains about 210, 000 sentences and its originally the subtitle of a TED presentation[17].

3.2 Data Preprocessing

In data preprocessing, the task is to prepare the data such that the encoder can read the data in its's input format in order to generate information vector and pass it to the decoder. Since the dataset contains two files of TED talk translated transcripts between English and Mandarin, the first step is to generate vocabulary file of the dataset and sorted them by their frequency in the training data. Before generating vocabulary file, the words with low frequency are removed. Also, three special character '$< unk >$' (unknown), '$< sos >$', and '$< eos >$' are added at the beginning of the vocabulary file. In this experiment we choose to have 5000 vocabulary size for mandarin and 10000 vocabulary size

English Vocabulary	Mandarine Vocabulary
<unk>	<unk>
<sos>	<sos>
<eos>	<eos>
,	的
.	，
the	我
to	。
of	是
and	们
a	—

Figure 6: Vocabulary Example.

for English. An example of generated vocabulary file are provided in Figure 5.

The next step is to map each sentence in the dataset to the vocabulary by their vocabulary position. After the mapping, the word with low frequency in each sentence will be change to '< *unk* >'. Then, '< *eos* >' is added to the end of each sentence. For example, a sentence in mandarin ['谢', '谢', '大', '家', '。', '<eos>'](means thank you in English) will become a number sequence ['226 226 30 61 6 2\n']. The next step is filtering, empty sentence and sentence length greater than 50 words will be removed from the dataset. Finally, apply data batching and padding after a random shuffle of the dataset.The batch size is 100.

3.3 Model Specification

In this experiment, we use the seq2seq model in Tensor-flow to perform the translation task. The structure of the encoder and decoder are similar to the seq2seq model. Except that we use a two layer LSTM instead of the original one layer LSTM in seq2seq model. First, in our model, we define the LSTM structure of the encoder and decoder with 1024 hidden layer size and 2 layers of LSTM. Next, we add word embedding layer with dropout

```
Source language: The weather is good . <eos>
Source language number sequence: [60, 1990, 13, 149, 4, 2]
Target language number sequence: [ 1 344 170 7 80 3 148 2]
Target language: <sos>气候是好的.<eos>
```

Figure 7: Testing Example.

rate of 0.2 for both encoder and decoder. Next, we add softmax layer to the decoder. We also include a shared parameters between softmax layer and embedding layer to improve the performance of our model. As discussed in section 2.3 we use log perplexity as our loss function. In addition, we use gradient decent as our optimizer.

3.4 Training and Testing

The network is trained on a Nvidia GTX1070 GPU with 8GB memory. The training steps are straight forward, The first step is hyper-parameter initialization, as we discussed in the model specification, we define the hidden size of the LSTM, the number of layers in the LSTM, the English vocabulary size, the Mandarin vocabulary size, batch size, number of epoch, and the dropout rate. After the hyper-parameter initialization, we initialize our seq2seq model with the hyper-parameters. Then, we define the input data and initialize a session in Tensor-flow to train the data. During the training, in a single iteration we train the model until it go through all the data in the dataset, and print cost for every 100 steps. For every 200 steps we save the model to a check-point file for testing the model. The check-point file can use to test the model even the model is still training. To test our model we need to select some English sentence and convert them as input to decoder. After define the input we define the decoder model and load the check point data to the model. Next, the input are pass through the decoder to decode the target language translation. Finally, convert the number sequence generated from the decoder beck to target language sentence. Figure 6 is an example of a decoded translation.

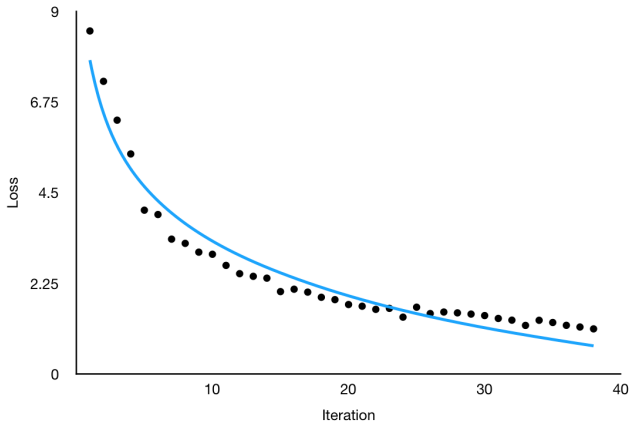

Figure 8: Convergence Graph

Source language: The weather is good .

Target language: 气候是好的.

Source language: write report takes a lot of time .

Target language: 报告写很多时间。

Source language: Today I 'm going to talk about a project in language translation .

Target language: 今天，我来说一下一个关于语言翻译的项目。

Figure 9: Translation Example

4 EXPERIMENT RESULT AND CONCLUSION

Figure 8. shows the convergence graph generated from the loss training step of the model with 40 iterations. It shows that log perplexity is decreasing over time. From the graph, we can observe that the model is improving over time. In the beginning, we have log perplexity around 8 which means our model has about 300 words to choose in a prediction. After 10 iterations the loss decrease below 3. The model shows a significant improvement from the beginning, the number of word choice decrease from 300 to 8 in a prediction. Finally, after 40 iterations the loss decrease to 1.119 which means that there are two choices for the model to decide what is the next output. Overall the training process of our model takes about 24 hours to train 40 iterations.

Figure 9. shows some test example of the translation from English to Mandarin. We compare the translation results to a set of translated languages pairs to evaluate the accuracy of the translation. We use 50 translated sentence pairs to compare to the generated translations. Unlorturnaly, the accuracy of the test result is only about 50 percent. An interesting finding is the model can translate short sentences very well. For example, the sentence' I can finish the assignment', 'The weather is good' and even 'Today I'm going to talk about a project in language translation'. Most of the translated results are grammatical correct. However, when the structure of gets complicated the accuracy of the translation decreases. To test the accuracy of the model, 50 translated sentence pairs are compared to the translation result in the model. The result shows that the model has an accuracy of 50 percent.

To sum up, In this project we combine the concept of seq2seq model, log perplexity, batching, padding, dropout, and gradient descent optimization together to build our neural machine translation model. We use shared parameters between the embedding layer and softmax layer to increase the performance of our model. The training process of the model brings down the log perplexity to 1.119. However, the testing of a machine translation program is different than evaluating a language model. The decoder decodes the message without the knowledge of the prediction is correct or not. The only way to test its accuracy is to manually check each sentence or use an automated process to check.

5 REFERENCE

[1] Tommi V., Jaakko J., Sami V., (2010). Language Identification of Short Text Segments with N-gram Models. Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta

[2] David, M., Mitchell P., (1990). Parsing a Natural Language Using Mutual Information Statistics. 1990 AAAI-90 Proceedings.

[3] Cho, K., Merrienboer, B. V., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)

[4] Pham, V., Bluche, T., Kermorvant, C., Louradour, J. (2014). Dropout Improves Recurrent Neural Networks for Handwriting Recognition. 2014 14th International Conference on Frontiers in Handwriting Recognition.

[5] P. Koehn, F. J. Och, D. Marcu. Statistical Phrase-Based Translation. https://aclanthology.info/pdf/N/N03/N03-1017.pdf

[6] W. De Mulder, S. Bethard, M.-F. Moens, A survey on the application of recurrent neural networks to statistical language modelling (2015) Computer Speech and Language, 30 (1) , pp. 61-98.

[word embedding] D. Karani Introduction to Word Embedding and Word2Vec. https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

[7] Seq2seq basics : https://google.github.io/seq2seq/

[8] S. Sathasivam. Logic Learning in Hopfield Networks. Modern Applied Science, 2009.

[9] S. Hochreiter, J. Schmidhuber. Neural computation 9 (8), 1735-1780, 1997. 17177 , 1997. Deep learning in neural networks: An overview. J Schmidhuber.

[10] D. Karani Introduction to Word Embedding and Word2Vec. https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

[11]P. Tardy. H. learner. In NLP, why do we use perplexity instead of the loss? https://www.quora.com/In-NLP-why-do-we-use-perplexity-instead-of-the-loss/answer/Paul-Tardy

[12] W. De Mulder, S. Bethard, M.-F. Moens, A survey on the application of recurrent neural networks to statistical language modelling (2015) Computer Speech and Language, 30 (1) , pp. 61-98.

[13] M. Walker, Batch vs. Real Time Data Processing, http://www.datascienceassn.org/content/batch-vs-real-time-data-processing

[14] M. Dwarampudi, N V S. Reddy, Effects of padding on LSTMs and CNNs, https://arxiv.org/pdf/1903.07288.pdf

[15] W. Zaremba, I. Sutskever, O. Vinyals. Recurrent Neural Network Regularization https://arxiv.org/abs/1409.2329

[16] Chabacano. diagram showing overfitting of a classifier. https://en.wikipedia.org/wiki/Overfitting

[17] Dataset: Web Inventory of Transcribed and Translated Talks https://wit3.fbk.eu/download.php?release=2015-01