DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING (ELEC0141) 23 REPORT

SN: 22039170

ABSTRACT

Recurrent Neural Networks (RNNs) are powerful models that have achieved excellent performance on sequence learning tasks. Although RNNs work well whenever large labeled training sets are available, they suffer from dealing with long squences. In this paper, we present a general end-to-end approach to sequence learning that uses an encoder to map the input sequence to a context vector, and then another decoder to convert the vector to the target sequence. Then, we introduce attention mechanism enabling the decoder to focus on a certain part of the input sequence in each decoding step. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the sequence to sequence model with greedy search algorithm achieve a BLEU score of 23.57 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, beam search could improve the model performance due to its ability to explore on a larger result set. For comparison, a sequence to sequence model with beam size 2 achieves a BLEU score of 24.24 on the same test-set. Moreover, the upgraded sequence to sequence with attention model also achieves a higher BLEU score of 25.33 with greedy search. Finally, we found that stacking more than one LSTM (Multi-Layer LSTM) will improve the model's performance markedly, because LSTM networks with multiple layers have higher modeling capacity compared to single-layer LSTM networks.

Index Terms— Machine Translation, Language Model, LSTM, Attention Mechanism, Beam search

1. INTRODUCTION

Machine translation is the automated process of translating text or speech from one language to another using computer algorithms. It automates language translation using natural language processing algorithms, offering benefits such as overcoming language barriers, facilitating global communication, enabling efficient information processing, enhancing accessibility, supporting international commerce, aiding language learning and research.

Traditional machine translation model relies on phrase-based method that many small sub-components are tuned separately [1]. Neural machine translation (NMT) model, on the other hand, attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation. Neural network (NN) makes building an end-to-end machine translation system possible by its excellent ability to fit complex models by optimization algorithms such as gradient descent. In addition, high performance computing devices like GPUs and TPUs enable NN models trained on very large datasets, which could lead to better performance of the NMT model. Researchers have conducted many studies on NMT model and found that NMT model could pair with or even outperform traditional machine translation model on some translation tasks [2, 3, 4, 5].

Recurrent neural network (RNN) is specifically designed to handle sequential data, making it well-suited for tasks like machine translation. RNN can take into account the sequential nature of sentences and capture dependencies between words or characters, which is crucial for accurate translation. Thanks to RNN's ability to process sequences, one could use RNN to build an encoder taking in sentence in one language (e.g. English) and a decoder taking in sentence in targeted language (e.g. French). The aforementioned encoder and decoder could be integrated as an encoder-decoder architecture which our sequence to sequence model applies. However, its model performance is penalized from the fixed-sized context vector generated by the encoder regardless of the length of input sequence. In this case, the decoder could only receive the fixed-sized context vector holding limited information. To mitigate this problem, attention mechanism is involved in the decoding step, enabling the decoder to adaptively generate context vectors. By applying attention mechanism, the decoder could focus on certain part of input sequence in each decoding step. BLEU (Bilingual Evaluation Understudy) score is used to evaluate the quality of the output of NMT models. It is based on the idea that a good translation should have similar characteristics to one or more reference translations by human translators.

The data preprocessing, model building, and training and testing pipeline are implemented in python code taking advantage of PyTorch framework.[1] It could reproduce the experimental results described in this paper.

The rest of this paper is organized as follows. Section 2 reviews research work related to machine translation, especially neural machine translation. Section 3 explains our proposed model and the rationale behind the choice. In Section

[1]The code is provided on GitHub: https://github.com/N1ghtstalker2022/ELEC0141_Project

4, detailed description of dataset, preprocessing, implementation of our neural machine translation models, and training and validation procedure are included. In Section 5, several experiments are conducted to evaluate the effectiveness of the proposed model. Finally, in Section 6 we conclude the paper and provide future work.

2. LITERATURE SURVEY

With the development of neural networks (NN), many applications start to utilize NN trying to obtain better performance than traditional methods. Since Bengio et al. identified the feasibility of modelling the conditional probability of a word given a fixed number of preceding words by NN, much research and work have been conducted to the application of NN to machine translation task [6]. Schwenk applied feedforward neural network to the phrase-based statistical machine translation (SMT) system. The feedforward NN is used to compute the score of a pair of source and target phrases as an additional feature in their SMT system [7]. More recently, Recurrent Neural Network (RNN) have been found to be able to enhance the performance of SMT model by learning phrase representations [3]. It is demonstrated that a standard phrase-based SMT system could benefit from the use of NN-based model as part of the machine translation system. Kalchbrenner and Blunsom proposed the Recurrent Continuous Translation Model which converts sentences into vectors by Convolutional Neural Networks (CNN) and models the generative architecture with Recurrent Neural Networks (RNN) producing representations for targeted translations, but it does not consider the ordering information of words within sentences [8]. Collectively, the above approaches indicate the feasibility of involving neural networks in traditional SMT models either by generating new features or rescoring the list of candidate translations. Although these approaches were shown to improve the performance of translation over former machine translation systems, researchers attemped to seek the possibility of build an end to end machine translation system based solely on neural networks which also is known as neural machine translation (NMT) system. Unlike traditional statistical machine translation (SMT) approaches, which often rely on handcrafted linguistic rules and feature engineering, NMT models could directly map the source language to the target language, without explicitly relying on intermediate representations. Sutskever et al. proposed an end to end NMT system using layers of Long Short Term Memory units(LSTM) [2]. Specifically, their NMT system applies encoder-decoder architecture where sentences are encoded into fixed-length vectors which are later decoded back to variable-length translated sentences and achieves a performance close to phrase-based SMT system on English-to-French translation task. A follow-up research by Cho and Bengio found that the performance of encoder-decoder based NMT model could be improved by allowing the decoder to search for parts of a source sentence which are relevant to a target word [9]. In their NMT model, an alignment model is applied to decoder architecture to let the decoder decide parts of the source sentence to attend to, which is also known as attention mechanism. Gehring et al. found that the using of CNN as encoder in the sequence to sequence model instead of RNN could achieve competitive performance on multiple translation tasks. CNN allows the encoder to encode the source sentence simultaneously, which eliminates the computational constraints introduced by recurrent networks [5]. Kaiser and Bengio, on the other hand, proposed an extended model of active memory which updates the memory of encoder in parallel rather than remain unchanged in attention mechanism [4]. It matches existing attention models on neural machine translation and generalizes better to longer sentences.

3. DESCRIPTION OF MODELS

Machine translation is one of the classic natural language processing tasks. The encoder-decoder architecture has been applied to build NMT system thanks to its ability of dealing with two sequences with different lengths.

To model the machine translation process, one could describe the problem in a probabilistic perspective. Given a source sentence represented as a sequence of words, denoted as $X = (x_1, x_2, \ldots, x_T)$, and a target sentence represented as $Y = (y_1, y_2, \ldots, y_{T'})$, the probability model calculates the conditional probability of generating the target sentence given the source sentence:

$$P(Y|X) = \prod_i P(y_i|y_1, \ldots, y_{i-1}, X)$$

In this formula, $P(y_i|y_1, \ldots, y_{i-1}, X)$ represents the probability of generating the i-th target word, y_i, given the previous target words, $y_1, y_2, \ldots, y_{i-1}$, and the source sentence, X.

In the following section, recurrent neural network is introduced first Then the plain sequence to sequence model is discussed, and then the more advanced sequence to sequence with attention model is introduced with its potential advantages over the plain sequence to sequence model.

3.1. Recurrent Neural Network

A Recurrent Neural Network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior, which makes it particularly suited to tasks involving sequence prediction like machine translation.

Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs, which allows them to capture information about what has been calculated before. Given an input sequence $(x_1, x_2, \ldots, x_T)$, a

standard RNN could output a sequence $(y_1, y_2, ..., y_T)$ by iterating two following equations:

$$h_t = \text{sigm}(W_{hx}x_t + W_{hh}h_{t-1})$$
$$y_t = W_{yh}h_t$$

, where h_t is the hidden state of RNN at time t, W_{hx}, W_{hh}, and W_{yh} are linear transforms provided by RNN, and sigm represents the sigmoid activation function. The working rational of an RNN could be demonstrated in Fig. 1

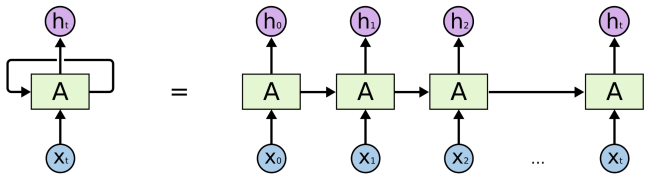

Fig. 1. RNN with an unrolled representation. A represents a chunk of neural network performing linear transformations.

The machine translation task could be regarded as a sequence learning task which requires the model to learn from the input sequences. An Encoder-Decoder architecture is applied to map the input sequence to a fixed-sized vector using one RNN, and then map the vector to the output sequence with another RNN.

3.2. Sequence to Sequence Model

The sequence to sequence model consists of two main components: an Encoder and a Decoder, both of which are implemented as Long Short-Term Memory (LSTM) instead of traditional RNN, since traditional RNNs suffer from long term dependencies due to the vanishing and exploding gradient phenomena during backpropagation [2].

In our model, Long Short-Term Memory (LSTM) networks are used as a special kind of RNN, capable of learning long-term dependencies for sequences (i.e., learning to connect past information to the present task, such as predicting the next word based on a series of previous words).

In an LSTM, the hidden layer updates are replaced by a memory cell which contains three crucial components: an input gate, a forget gate, and an output gate. These gate units, coupled with a memory cell maintaining cell state and hidden state, control the flow of information inside the network. The gates selectively allow information to flow in and out of the cell, enabling the LSTM to maintain or erase its internal state, thereby effectively capturing temporal dependencies of different lengths.

Mathematically, the LSTM updates its memory cell C_t and hidden state h_t based on the current input x_t, the previous hidden state h_{t-1} and the previous cell state C_{t-1}. At each time step t, the input gate i_t decides the new information to be stored in the cell state.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$\tilde{C}_t$ represents the candidate cell state.

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

The forget gate f_t determines the extent to which the LSTM keeps the previous state.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

Then, the cell state is computed by combining the information of the previous cell state modulated by the forget gate and the candidate cell state modulated by the input gate.

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Finally, the new hidden state h_t is calculated by applying tanh activation function to the C_t with an output gate controlling the effectiveness of the cell state.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$

In the equations above, σ denotes the sigmoid function, tanh is the hyperbolic tangent function, W_i, W_C, W_f, and W_o are weight matrices of linear transformations, b_i, b_C, b_f, and b_o are bias vectors, and $[h_{t-1}, x_t]$ denotes the concatenation of the previous hidden state and the current input. We define $\theta = \{W_i, W_C, W_f, W_o, b_i, b_C, b_f, b_o\}$ as the parameters of an LSTM unit.

In our sequence to sequence model, LSTM units are used as the building block of both encoder and decoder. A diagram of LSTM units is shown in Fig. 2.

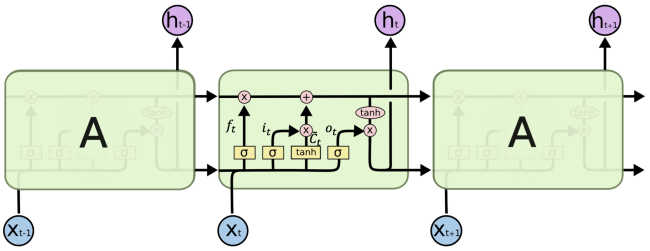

Fig. 2. The diagram shows the structure of LSTM (denoted as A) and how it processes input sequences and generates new sequences.

The **encoder** processes the input sequence $(x_1, x_2, \ldots, x_T)$ into a fixed-length context vector c. This is accomplished by applying an recurrent transformation parameterized by a function f that updates the internal hidden state h at each time step t. Hence we have

$$h_t = f(x_t, h_{t-1}; \theta_e)$$

, where x_t is the input at time step t, h_{t-1} is the previous hidden state, and θ_e are the parameters of the LSTM. Since

only the final hidden state of the encoder is used as the initial hidden state of the decoder, the context vector c here should be h_T where T represents the last time step of the encoder.

The **decoder** is another LSTM, designed to output a sequence whose length may be different from that of input sequence. Given the context vector c, the decoder generates the output sequence $(y_1, y_2, ..., y_{T'})$. Similar to the Encoder, the Decoder also applies a recurrent transformation parameterized by a decoder function g that updates the hidden state of the decoder s and produces an output y at each time step t'. Each decoding step could be represented mathematically as

$$s_{t'} = g(y_{t'-1}, s_{t'-1}, c; \theta_d)$$

, where $y_{t'-1}$ is the previous output, $s_{t'-1}$ is the previous decoder hidden state, c is the context vector, θ_d are the parameters of the decoder. After the hidden state of current time step is computed, the predicted output for time step t' could be generated by a linear transformation followed by a softmax function

$$y_{t'} = \text{softmax}(W_s s_{t'} + b_s)$$

, where W_s and b_s are parameters of a linear transformation and the softmax function transforms the vector generated by the linear transform into a probability distribution, ensuring all output elements are between 0 and 1 and sum to 1.

The architecture of the sequence to sequence model is shown in Fig. 3

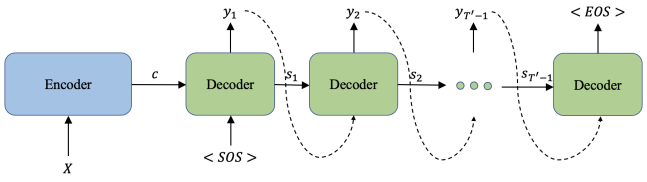

Fig. 3. The diagram shows the structure of sequence to sequence model. It is noteworthy that the input in the first decoding step is <SOS> (start of sentence) and once the output of one decoding step becomes <EOS> (end of sentence) the translation process ends. Both <SOS> and <EOS> are put into vocabularies of both input sentences and target sentences.

3.3. Sequence to Sequence with Attention

Bi-LSTM is used for the encoder instead of LSTM to let the computed information of each word (i.e. hidden state) include not only preceding words but also the following words. A simplified Bi-LSTM diagram is demonstrated in Fig. 4.

The Bidirectional Long Short-Term Memory (Bi-LSTM) architecture is a variant of recurrent neural networks (RNNs) designed to capture bidirectional dependencies in sequential data. In a Bi-LSTM, the forward LSTM processes the input sequence $(x_1, x_2, \ldots, x_T)$ in the natural order, while the backward LSTM processes it in reverse order. At each time step t, the forward LSTM updates its hidden state $\overrightarrow{h}_t$ and cell state $\overrightarrow{c}_t$ using the input x_t and the previous hidden and cell states:

$$\overrightarrow{h}_t, \overrightarrow{c}_t = \text{LSTM}(x_t, \overrightarrow{h}_{t-1}, \overrightarrow{c}_{t-1})$$

Similarly, the backward LSTM updates its hidden state $\overleftarrow{h}_t$ and cell state $\overleftarrow{c}_t$ using the input x_t and the previous hidden and cell states:

$$\overleftarrow{h}_t, \overleftarrow{c}_t = \text{LSTM}(x_t, \overleftarrow{h}_{t+1}, \overleftarrow{c}_{t+1})$$

The final hidden state representation of the Bi-LSTM at each time step is obtained by concatenating the forward and backward hidden states:

$$h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$$

where $[;]$ denotes concatenation. h_t represents the hidden state that encode both forward and backward contextual information.

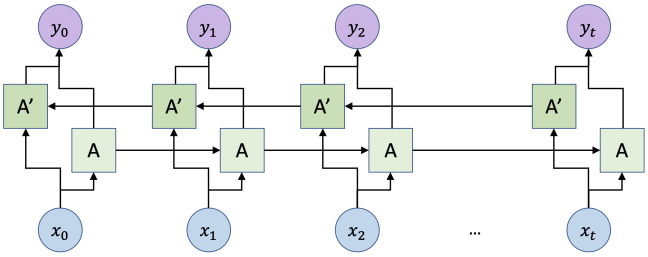

Fig. 4. The diagram shows the structure of Bi-LSTM. It is noteworthy that A and A' represent neural units for foward and reverse LSTM respectively.

One main weakness of the original Sequence to Sequence model is that it encodes the whole sequence into a fixed-sized vector to serve as the context vector for the decoder. As a consequence, it may lose a large amount of information of the long input sequence.

The Sequence to Sequence with Attention uses a weighted sum of hidden states of the encoder as the context vector, which mitigates the aforementioned problem. The weights reflecting the relevance of each source word at each decoding step, are calculated adaptively by applying attention mechanism.

At decoding time step t', the context vector $c_{t'}$ is computed by

$$c_{t'} = \sum_{t=1}^{T} \alpha_{t't} h_t$$

, where h_t contains information about the whole input sequence with a strong focus on the parts surrounding the t-th word of the input sequence, and the weight $\alpha_{t't}$ is formularised by the softmax function

$$\alpha_{t't} = \frac{\exp(e_{t't})}{\sum_{k=1}^{T} \exp(e_{t'k})}$$

with

$$e_{t't} = a(s_{t'-1}, h_t)$$

where a represents a score fuction measuring the extent to which the features around t-th input matches the (t'-1)-th hidden state of the decoder.

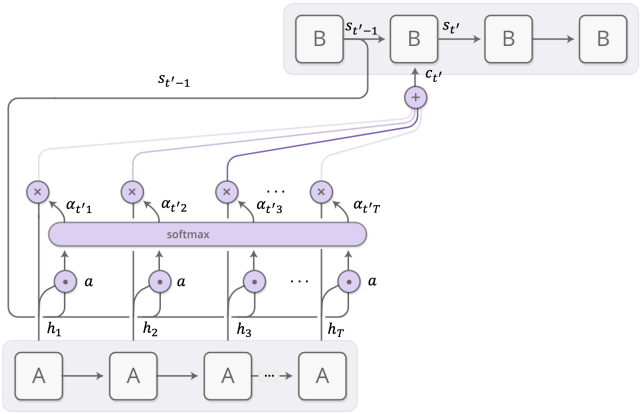

Fig. 5. The diagram shows the architecture of sequence to sequence with attention model. A represents Bi-LSTM units used for encoder. B represents LSTM units used for decoder. The attending LSTM generates a query describing what it wants to focus on. Each hidden states of encoder is computed with the query by a score function denoted by $\odot$, describing how well it matches the query. The scores are fed into a softmax to create the attention distribution.

Since the context vector is not fixed in contrast with the original sequence to sequence model using the fixed context vector, the former equation could be reformularized as

$$s_{t'} = g(y_{t'-1}, s_{t'-1}, c_{t'}; \theta_d)$$

. We then pass $c_{t'}$, $s_{t'}$, and $y_{t'-1}$ through the linear layer L, to make a prediction of the next word in the target sentence, $y_{t'}$. This is done by concatenating them all together.

$$y_{t'} = L(c_{t'}, s_{t'}, y_{t'-1})$$

4. IMPLEMENTATION

4.1. Dataset and Preprocessing

In this project, English-French sentence pairs of Europarl v7 dataset from WMT 2014 is used [1]. Concretely, it contains 2,007,723 parallel English and French sentences. There are 50,196,035 English words and 51,388,643 French words in total. Due to limited computing resources, ten percent of the original English-French sentence pairs are sampled and then split into train, validation, and test set with a ratio of 8:1:1, as it is shown in Table 1 and Table 2. From the dataset, we further extract sentences with less than twenty tokens.

Table 1. Number of samples for train, validation and test set

Dataset	Train	Validation	Test	Total
No. of Samples	160,617	20,077	20,078	200,772

During preprocessing, the manipulations towards English and French are identical, so here we make general descriptions for simplicity. All sentences are tokenized into single words, (e.g. English sentence "Resumption of the session" could be tokenized to separated words: 'Resumption', 'of', 'the', 'session'). Then the vocabulary set is built based on words whose frequency over the dataset is larger than ten. '<pad>', '<sos>', '<eos>', and '<unk>' are also included in the vocabulary to represent padding, start of sentence, end of sentence and unknown words, respectively. In addition, a python class called "TranslationDataset" which extends the pytorch "Dataset" class is created with all sentences and vocabularies.

Table 2. Three samples of English and French sentence pairs of the dataset.

Samples	English	French
One	Resumption of the session.	Reprise de la session
Two	Please rise, then, for this minute' s silence.	Je vous invite à vous lever pour cette minute de silence.
Three	Madam President, on a point of order.	Madame la Présidente, c'est une motion de procédure.

It is argued by Sutskever et al. that reversing the order of input sequence fed into encoder will improve the performance of the sequence to sequence model because it is easier to establish connections between the input sequence and output sequence by making possible word pairs closer [2]. With this thought, we also build a dataset with all sentences reversed.

4.2. Model Implementation

Both sequence to sequence model and its updated version with attention is implemented in PyTorch thanks to torch.nn module which provides a large variety of linear and non-linear transformations. The embedding layer is built to convert words into 300 dimension vectors holding features of words. We tried two ways of vectorization of words: one is to use pretrained word2vec embeddings provided by Google [10], and the other is to train the embeddings from scratch. Concretely, PyTorch class nn.Embedding is used as embedding layer where word represented as a single number could be mapped to its corresponding vector representation.

In terms of encoder implementation, class nn.LSTM is used for building LSTM and Bi-LSTM for sequence to sequence model and its updated version with attention, respectively.

Decoders are also built by nn.LSTM, although attention block is added to the decoder for sequence to sequence with attention model. Concretely, for attention block, the scaled dot-product operation is chose as the score function computing the correlation of hidden states of encoder and decoder[11]. The decoder generates the next word in one time step. Instead of using teacher forcing method which discards the preceding generated word and input the actual word for the next decoding step, we use the predicted word or actual word by teacher forcing with a fixed probability.

4.3. Training and Validation

The code for training and validation is implemented in PyTorch. For training and validation process, the corresponding dataloaders are built with a batch size 128. Batch-wise padding is applied to ensure sentences within a batch holding the same length, allowing efficient processing of multiple sequences in parallel. Mini-batch gradient descent strategy is applied to update parameters of the model, (i.e. model parameters are updated based on the gradients computed on each mini-batch with 128 samples). Concretely, for mini-batch gradient descent, torch.optim.SGD optimizer provided by PyTorch is used with a learning rate of 0.01 and an weight decay rate of 0.001 to prevent overfitting. We use negative log-likelihood loss as the loss function (nn.NLLLoss fucntion in PyTorch), since for each decoding step, our language model generates the probability of all words in the vocabulary and compare it with true target word.

Dropout which is a regularization technique commonly used in neural networks is applied to our . It helps prevent overfitting and improves generalization by randomly dropping out (setting to zero) a fraction of the neurons during training.

The learning curve of the sequence to sequence model is shown in Fig. 6. During the first 20 epochs, both the train loss and validation loss monotonically decrease, although the rate of decrease gradually slows down. Meanwhile, the validation loss is very close to the validation loss, indicating no salient overfitting problem in the current state of the model. However, from the figure we can also argue that the model now is still underfitting since there is trend of descent of both train and validation loss, which means more epochs of training are needed to make the model converge.

In this project, computation of model training is carried out on GPUs. Training data is divided among four NVidia 80Gb Tesla A100 GPUs to train our model in parallel (i.e. processing different portions of the data simultaneously). By data parallelism, the training process could be accelerated and therefore efficiency could be improved.

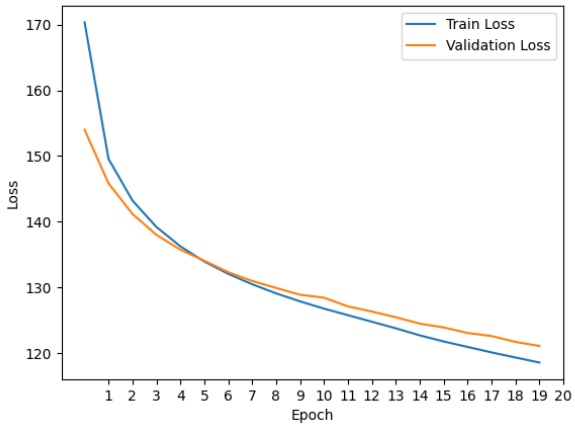

Fig. 6. The learning curve for sequence to sequence model.

5. EXPERIMENTAL RESULTS AND ANALYSIS

When evaluating our model, two decoding strategies are applied: greedy search and beam search. Greedy search is simply use the predicted word (the prediction having the largest probability) of the last step as the input of the next decoder unit. Beam search allows for exploring a larger search space by considering multiple candidate sequences simultaneously. This could help in finding more diverse and potentially better solutions. BLEU score is applied to measure the correctness of the machine translation of our model. As shown in 3, beam search could make the model perform better and the larger the beam size the better translation results. The reason could be by maintaining a set of top-scoring partial hypotheses beam search can help avoid getting stuck in local optima. This allows for exploring alternative paths and prevents premature convergence to local optimum.

Table 3. Different decoding strategies tested on the same model (sequence to sequence). Beam search (2) denotes a beam search strategy with beam size two.

Model	Decoding	BLEU score	Sec/sentence
Seq2Seq	Greedy search	23.57	0.02
Seq2Seq	Beam search (2)	24.24	0.08
Seq2Seq	Beam search (4)	25.02	0.15
Seq2Seq	Beam search (8)	25.87	0.33
Seq2Seq	Beam search (16)	26.02	1.24

Detailed ablation study is conducted to justify our methods (Table 4). It is found that our sequence to sequence model with attention performs better than the plain seq2seq model. Also, model using pretrained English embeddings which are frozen during training has lower BLEU score than the one training embedding from scratch, which indicates

that although pretrained word embeddings were trained on very large dataset using many high performance computing devices its generalization feature limit its performance on specific task like machine translation. In addition, model with multiple layers of LSTM generates higher BLEU score, since deep LSTM networks are capable of capturing hierarchical dependencies in sequential data. Each LSTM layer in the network can learn and model different levels of abstraction in the data, allowing for the representation of complex patterns and long-term dependencies.

Table 4. A comparison of experimental results for our two models. Seq2Seq refers to our sequence to sequence model. Seq2Seq refers to our sequence to sequence with attention model. Embedding refers to the word embedding. LSTM Layer No. refers to the number of layers applied to deep LSTM for both encoder and decoder.

Model	Embedding	LSTM Layer No.	BLEU score
Seq2Seq	Pretrained	One	23.57
Seq2Seq Attention	Train from Scratch	One	24.24
Seq2Seq	Pretrained	Two	25.02
Seq2Seq Attention	Train from Scratch	Two	25.87
Seq2Seq	Pretrained	Four	26.02
Seq2Seq Attention	Train from Scratch	Four	27.42

Here in Table 5, some examples of translations from English to French are displayed. It shows that our model could generate descent translations, although in some place it is not accuracy.

Table 5. A comparison of experimental results for our two models. Seq2Seq refers to our sequence to sequence model. Seq2Seq refers to our sequence to sequence with attention model. Embedding refers to the word embedding. LSTM Layer No. refers to the number of layers applied to deep LSTM for both encoder and decoder.

Source	An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.
Reference	Le privilege dadmission est le droit dun medecin, en vertu de son statut de membre soignant dun hopital, dadmettre un patient dans un hopital ou un centre medical afin dy delivrer un diagnostic ou un traitement.
Seq2Seq	Un privilege dadmission est le droit dun medecin de reconnatre un patient a lhopital ou un centre medical dun diagnostic ou de prendre un diagnostic en fonction de son etat de sante.
Seq2Seq Attention	Un privilege dadmission est le droit dun medecin dadmettre un patient a un hopital ou un centre medical pour effectuer un diagnostic ou une procedure, selon son statut de travailleur des soins de sante a lhopital.
Google Translate	Un privilege admettre est le droit dun medecin dadmettre un patient dans un hopital ou un centre me dical pour effectuer un diagnostic ou une proce dure, fonde e sur sa situation en tant que travailleur de soins de sante dans un hopital.

6. CONCLUSION

In this paper, we presented a general end-to-end approach to sequence learning using recurrent neural networks (RNNs). We introduced an encoder-decoder architecture with attention mechanism that allows the model to effectively handle long sequences. The results of our experiments on an English to French translation task from the WMT'14 dataset demonstrate the effectiveness of our proposed approach.

The sequence to sequence model with greedy search achieved a BLEU score of 23.57 on the entire test set, while penalized on out-of-vocabulary words. Furthermore, the use of beam search, multilayer LSTMs could also improved the model's performance, with four layer LSTM resulting in a BLEU score of 27.42.

Additionally, the incorporation of attention mechanism in the sequence to sequence model further enhanced the translation quality, leading to a higher BLEU score of 25.33 with greedy search. We observed that the attention mechanism allowed the decoder to focus on specific parts of the input sequence, improving the overall translation accuracy.

Moreover, we found that the use of Multi-Layer LSTM (stacking more than one LSTM layer) significantly improved the model's performance. The increased modeling capacity of multi-layer LSTM networks compared to single-layer LSTM networks played a crucial role in capturing complex patterns and dependencies in the data.

Overall, our study demonstrates the effectiveness of the proposed approach in the machine translation task. The combination of encoder-decoder architecture, attention mechanism, and Multi-Layer LSTM provides a robust framework for achieving high-quality translations and addressing the challenges of dealing with long sequences in sequence learning.

7. REFERENCES

[1] Philipp Koehn, "Europarl: A parallel corpus for statistical machine translation," in *Proceedings of machine translation summit x: papers*, 2005, pp. 79–86.

[2] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[4] Łukasz Kaiser and Samy Bengio, "Can active memory replace attention?," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[5] Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin, "A convolutional encoder model for neural machine translation," *arXiv preprint arXiv:1611.02344*, 2016.

[6] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent, "A neural probabilistic language model," *Advances in neural information processing systems*, vol. 13, 2000.

[7] Holger Schwenk, "Continuous space translation models for phrase-based statistical machine translation," in *Proceedings of COLING 2012: Posters*, 2012, pp. 1071–1080.

[8] Nal Kalchbrenner and Phil Blunsom, "Recurrent continuous translation models," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1700–1709.

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

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.