Seq2Seq Translation Report

– Machine Translation from Chinese to English

Abstract

The seq2seq is the mainstream framework for neural machine translation. Most of today's commercial machine translation systems are based on it. In this assignment, we will use the Chinese and English text provided by NIST to train a simple Chinese-English translation system.

In this assignment, we first compared the performances of single/bi-directional GRU/RNN/LSTM models and chose the best network to proceed. Hyperparameter tuning experiments were run on hidden size and layer amount to find the best learner model under our task and data. The best performance without attention enabled has achieved a BLEU score at 0.8728 on the validation set .

Among the networks we tested, GRU always has the best learning quality, whilst the RNN architecture is the worst. Using bidirectional networks significantly improves the performance of unidirectional networks. Bidirectional enhances context understanding by processing data in both forward and backward directions. The hidden state is a combination of information from both directions, enriching the model's comprehension. In speech recognition, Bidirectional RNNs capture nuanced patterns, improving accuracy.

However, still, we could notice that the translation can miss semantic meanings. One of the major improvements could be made based on how we separate tokens from the source Chinese language. Currently, we extract Chinese tokens on a byte-base, yet using third-party segmentation modules could be expected to make significant progress, since the byte-based tokens are clearly missing grammar and semantic meanings, compared to subwords.

Another enhancement would be added in the attention mechanism. At the time we handed in this assignment, we were still debugging the attention part of codes. It was a pity we did not successfully implement such a mechanism, and we are definitely going to try it out in the future.

1 Introduction

Fundamentally, machine translation requires mapping an input sequence (words in the source language) to an output sequence (words in the target language). Recurrent Neural Networks (RNN) effectively handle this type of sequential data. An important difficulty in machine translation is that there is no one-to-one correspondence between input and output sequences. That is, sequences are often of different lengths, and word correspondences can be nontrivial (e.g., words that are directly translated from each other may not appear in the same order).

To solve the above problem, we will use a more flexible architecture called seq2seq model. The model consists of two parts: encoder and decoder, both of which are RNN. The encoder takes as input a sequence of words in the source language and outputs the final hidden state of the RNN layer. The decoder is similar except that it has an additional fully connected layer (with softmax activation) that defines the probability distribution of the next word in the translation. In this way, the decoder essentially serves as a neural language model of the target language. The key difference is that the decoder uses the output of the encoder as its initial hidden state, rather than a vector of zeros.

We will investigate how different encoder/decoder networks can affect the translation quality, and if adding in attention can significantly improve the translation model performances.

In addition, many thanks to LiWei Hou, for helping me look at the code base and debug whilst changing the original dataframe from 1 sample at an iteration to minibatches.

2 Model Architecture

RNN, GRU and LSTM are examined as the encoder/decoder network. Three different networks have similar architectures, the below section will use the LSTM as an example to showcase the model flowchart.

The code implementation is inspired by [9].

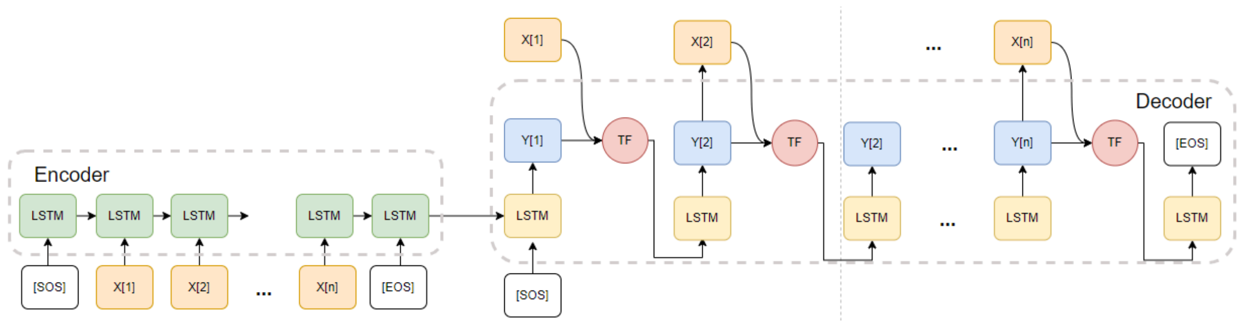
2.1 Data Preprocessing

A pipeline to preprocess date has been provided with the assignment code framework.

For each sentence in the source and the target language, three special symbols are introduced here, "SOS" means "Start of sentence" and "EOS" means "End of sentence". They are added to both ends of the input text to control the decoding process. “Padding” is used to fill the spaces if the sentence is not enough for a predefined maximum sequence length.

In order to improve the ability of computers to understand text-based content in a better way[1], we need to embed the text information extracted from the language and associate each unique token in our dataset with an index (an integer). The source language, Chinese, in our case, built the vocabulary token on a byte base. Whilst the target language, English, is built on a word base. A more advanced technique of extracting Chinese tokens could be invoking some third-party modules that tackle Chinese word segmentation, such as Jieba[2] and THULAC[3]. This is expected to be important for the Chinese language, since tokens segmented based on syntax will contain more semantic information。

2.2 Seq2Seq without Attention



Above is a diagram that showcases the architecture of the LSTM Seq2Seq model.

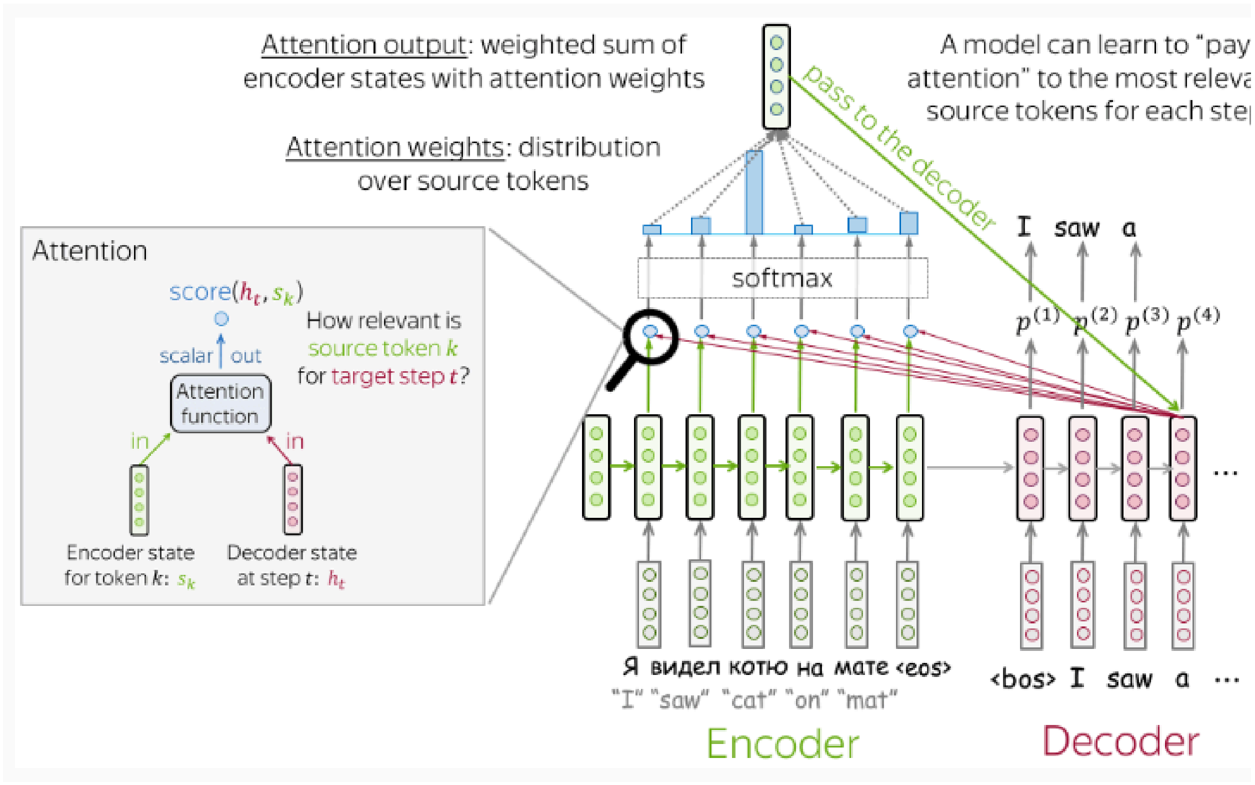
The source sentence, as the input X(embedded), is passed into the encoder. At each time-step, the input to the encoder is both the embedding of the current word, as well as the hidden state from the previous time-step. The hidden state is the representation of the sentence so far.

Once the final work is processed by the encoder, we use the final output hidden state as a vector representation of the entire source sentence. At each time-step, the input to the decoder is both the embedding of the predicted word from the last timestep, as well as the hidden state from the previous time-step. Note that we added a teacher forcing mechanism(red) for the decoder. If the teacher forcing is enabled, the input for the decoder would be the word that corresponds to the ground truth of the previous time step, as well as the last hidden state.

2.3 Seq2Seq with Attention

Traditional Seq2Seq model's encoder compressed the whole source sentence into a single vector. This can be very hard to represent the actual sequence of words - the number of possible source sentences (hence, their meanings) is infinite. When the encoder is forced to put all information into a single vector, it is likely to forget something[4].

This single vector representation also brings in limitations for the decoding process, since the decoder sees only one representation of source. However, at each generation step, different parts of the source can be more useful than others. This is the reason we introduce attention mechanisms.


Attention output: weighted sum of encoder states with attention weights
A model can learn to "pay attention" to the most releva source tokens for each ste
pass to the decoder
Attention weights: distribution over source tokens
softmax
Attention
score(h_t, s_k)
scalar out
How relevant is source token k for target step t?
Attention function
in
in
Encoder state for token k: s_k
Decoder state at step t: h_t
I saw a
$p^{(1)}$ $p^{(2)}$ $p^{(3)}$ $p^{(4)}$
Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"
<bos> I saw a ...
Encoder
Decoder

The above diagram is referenced from [4].

At each decoder step, attention receives a decoder state and all encoder states, and compute the attention scores. For each encoder state , attention computes its "relevance" for this decoder state. Formally, it applies an attention function which receives one decoder state and one encoder state and returns a scalar value as the attention score. The attention weights are calculated by applying a Softmax over the attention scores. And finally the attention output is the weighted sum of encoder states with attention weights.

At the time we handed in this assignment, we were still in the debugging stage, therefore the final submission will not include this part. But this is definitely something we will try to make work and test in the future.

3 Parameter Tuning and Model Comparison

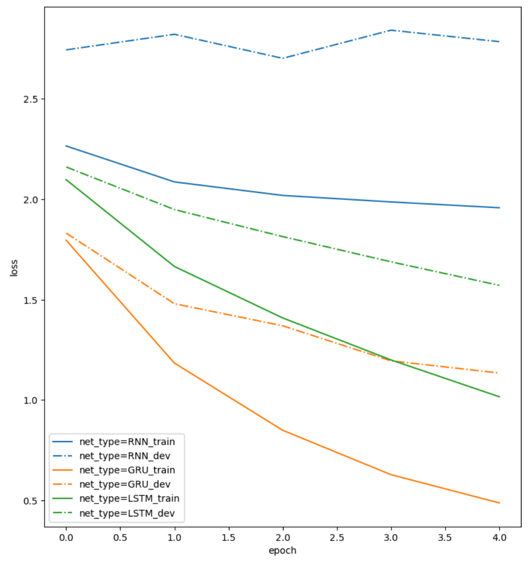
We will investigate how different encoder/decoder networks can affect the translation quality, and if adding in attention can significantly improve the translation model performances.

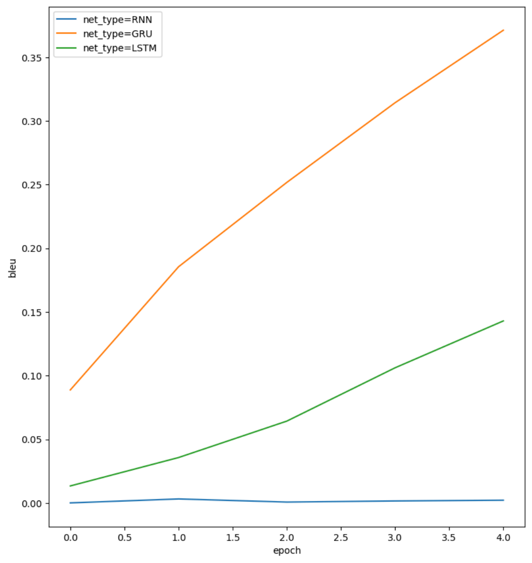
Loss and BLEU scores are used to evaluate the model performance. BLEU (bilingual evaluation understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Quality is considered to be the correspondence between a machine's output and that of a human[5]. Although it is one of the most popular evaluation metrics for NLP tasks, it has some limitations. BLEU score heavily relies on n-grams and may not capture the overall meaning or fluency of the translated text accurately. It may also penalize translations that are longer than the reference translations, which can be unfair in some cases[6].

3.1 Comparison between RNN/GRU/LSTM (No Attention, No Bi-direction)

Although we removed too long sentences, there are still around 89935 pairs. Therefore, in order to iterate the experiments faster, we only train the model for 5 epochs.

The batch size is set to 50, with the initial learning rate set to 0.001. The layer of encoder or architecture is set to 2, with the bi-directional mechanism disabled.

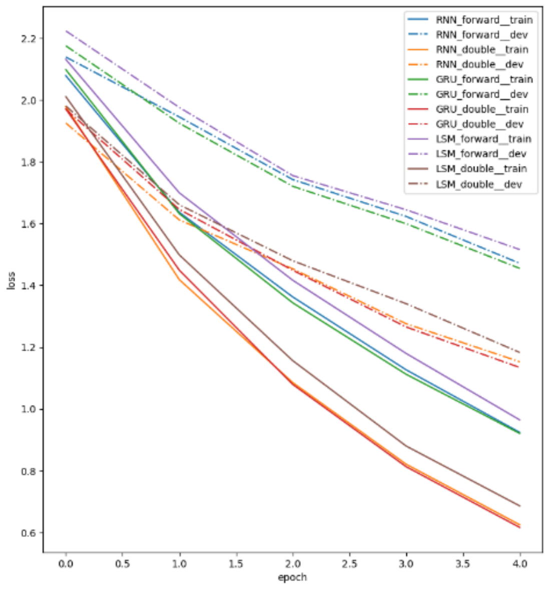


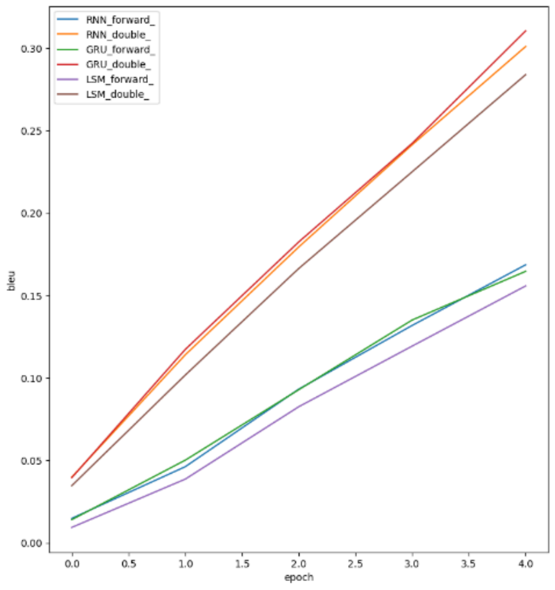


<u>Observations:</u>

1. Chart 1 is a visualization of the loss of the model on the training set and validation set, and Chart 2 is the BLEU score translated by the model on the validation set.
2. GRU is the best performer among the three models tested, and the BLEU score of its translated sentences far exceeds the other two networks.
3. The translation model based on RNN is the slowest to converge, and its loss on the verification set has almost no significant decrease, and there is an obvious overfitting trend in only 5 epochs.
4. The translation score of the RNN-based framework on the verification set is infinitely close to 0, and the model has not learned enough information to make effective inferences.
5. The performance of LSTM is far inferior compared to GRU, but obviously exceeds RNN. It can be seen from the loss curve that after two epochs, the loss reduction speed of LSTM and GRU on the training set and verification set is close, but in the first two epochs, the convergence speed of GRU is much higher than that of LSTM.
6. In the later section, we will further examine if enabling a bidirectional mechanism could change the current performance rank, although we highly suspect that GRU will still be the best.

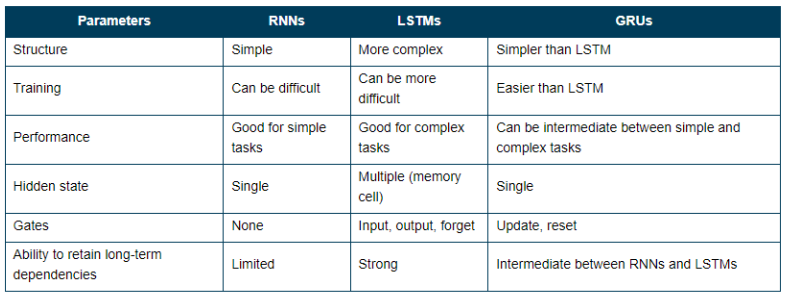
3.2 Comparison between RNN/GRU/LSTM (No Attention, Bi-direction)





Observations:

1. As expected, GRU is still the best-performing network structure, and its BLEU score for inference on the validation set significantly exceeds that of other networks.
2. Using bidirectional networks significantly improves the performance of unidirectional networks. Bidirectional enhances context understanding by processing data in both forward and backward directions. The hidden state is a combination of information from both directions, enriching the model's comprehension. In speech recognition, Bidirectional RNNs capture nuanced patterns, improving accuracy[7].
3. Surprisingly, after using the bidirectional RNN structure, the model performed better than the LSTM results. Even the performance of bidirectional RNN and bidirectional GRU is extremely close.
4. Below is a diagram comparing these three networks:

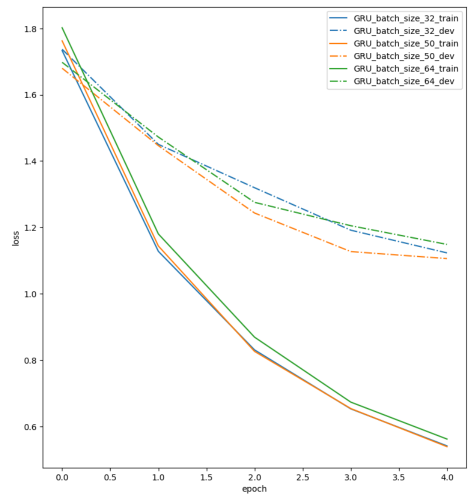


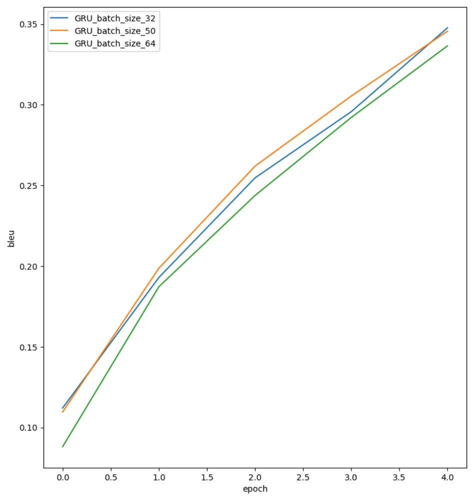
Parameters	RNNs	LSTMs	GRUs
Structure	Simple	More complex	Simpler than LSTM
Training	Can be difficult	Can be more difficult	Easier than LSTM
Performance	Good for simple tasks	Good for complex tasks	Can be intermediate between simple and complex tasks
Hidden state	Single	Multiple (memory cell)	Single
Gates	None	Input, output, forget	Update, reset
Ability to retain long-term dependencies	Limited	Strong	Intermediate between RNNs and LSTMs

5. We will choose GRU with bi-directional for later parameter tuning.

3.3 Batch Size

We will use bi-directional GRU with 5 epochs to test the batch size.

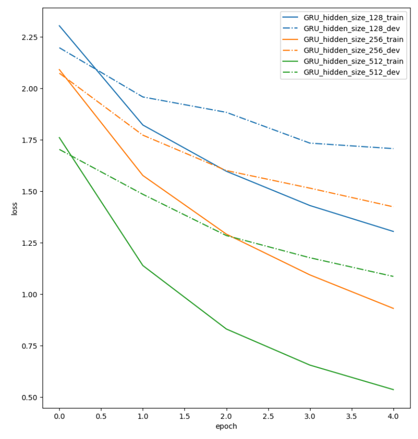


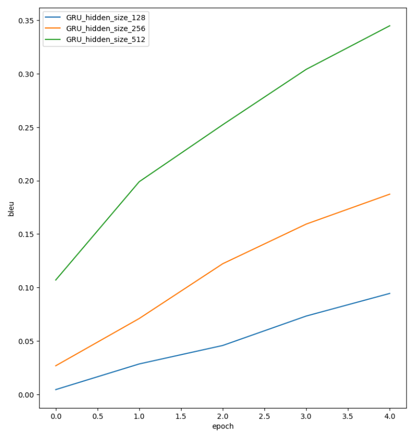


It can be seen from the figure that when the batch size is set to 32 and 50, the performance of the model is relatively close. Although the performance of batch size 50 has been better than that of batch size 32 in the first four epochs, the BLEU score of batch size 32 in the last epoch significantly exceeded that of batch size 50, and there is still an upward trend. So we will use batch size 32 as the basis for subsequent parameter adjustment.

3.4 Hidden Size

We will use bi-directional GRU with 5 epochs to test the hidden size. The batch size is set to 32. The hidden dimension sizes examined are: [128, 256, 512].





The dimension of LSTM hidden states can have an impact on the performance of the model. A larger dimension can potentially allow the model to capture more complex patterns in the data, leading to better performance in tasks such as language modeling, machine translation, and speech recognition. However, increasing the dimension also increases the number of parameters in the model, which can lead to longer training times and increased computational requirements. Additionally, the optimal dimension may vary depending on the specific task and dataset, so it's often a matter of experimentation to find the best setting for a particular application[8].

As can be observed from the diagram above, the greater the hidden dimension is, the faster the model converges, and the better translation the model inference on the validation set. Therefore, we will use 512 as the hidden dimension in the later experiments.

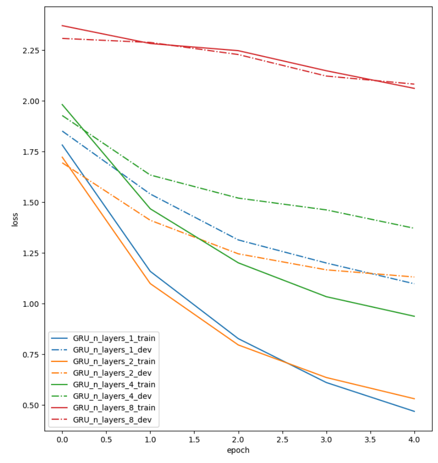
In addition, since we can see that 512(the greatest dimension we have in the test) has the best performance, we could assume that the best solution would be a value greater than 512. However, due to the limitation of time, we did not run further testing.

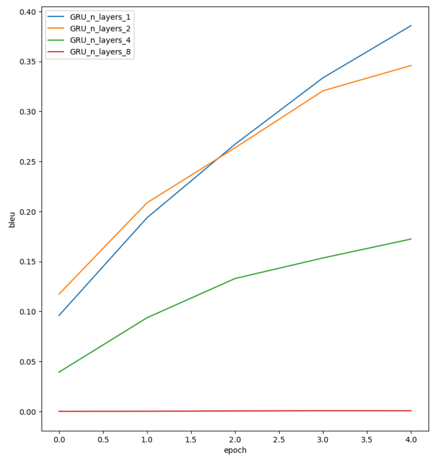
3.5 Number of layers

We will use bi-directional GRU with 5 epochs and 512 as the hidden size. The batch size is set to 32. Layer amounts examined are: [1, 2, 4, 8].

Observations:

1. When the number of GRU layers is set to 1 or 2, the performance of the two on the validation set is close. The model with a layer number of 2 initially performs better than a model with a layer number of 1, but becomes worse than the model with a layer number of 1 at about the 3-4 epoch.
2. The deeper the number of layers in the model, the worse the performance of the model. Especially when the number of layers is set to 8, you can see that there is almost no change in the loss curve and BLEU curve of the model. Deeper neural networks can sometimes perform worse than shallower ones due to the vanishing gradient problem and overfitting. The vanishing gradient problem occurs when the gradients of the loss function become extremely small as they are back-propagated through the network, making it difficult for the lower layers to learn effectively. This can hinder the training of deeper networks. Additionally, deeper networks can be more prone to overfitting, especially when the amount of training data is limited.

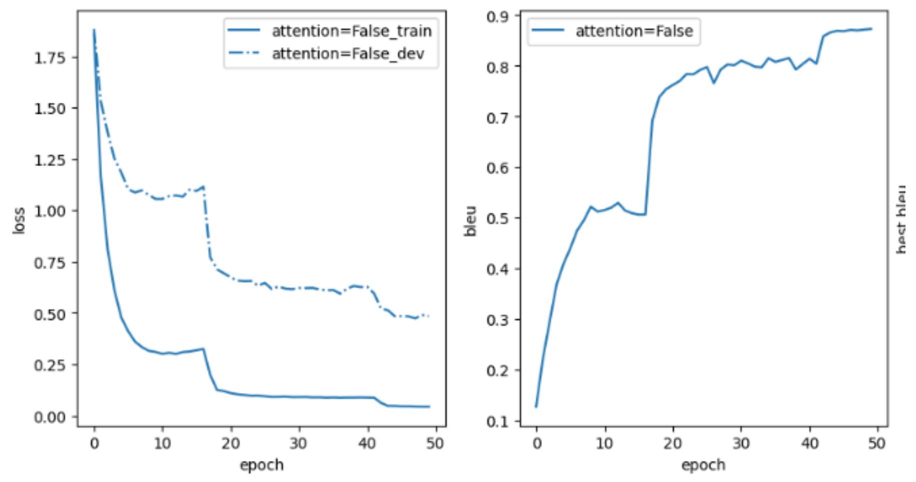




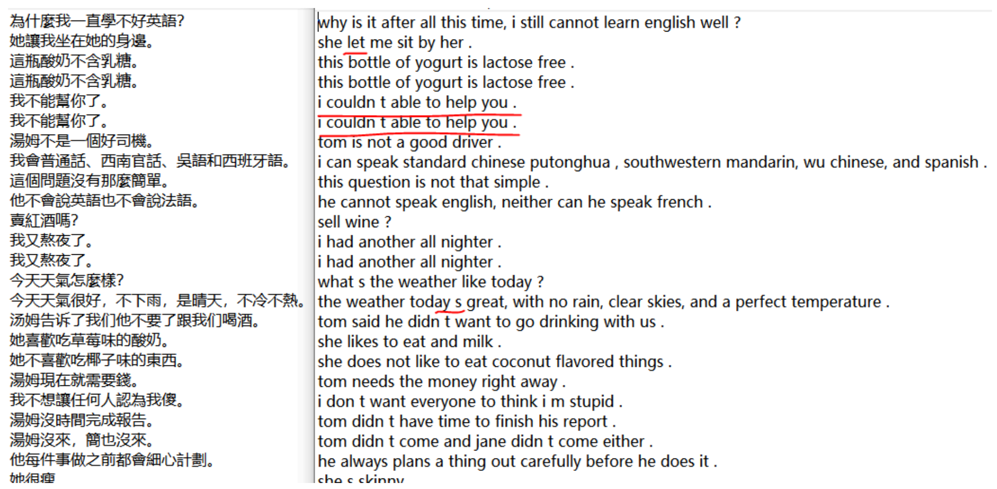
4 Final Model

Based on the above section, we finally decide on the hyperparameters for our final model without the attention mechanism involved. The parameters are:

hidden_size : 512, n_layers: 1, Bidirectional: enabled, batch_size: 32, network: GRU, teacher_foorcing_ratio: 0.5, and gradient_clipping: 5.0.



The best performance within 50 epochs has a BLEU score at 0.8728 on the dev set. Below are some translation pairs from the test set.



為什麼我一直學不好英語?	why is it after all this time, i still cannot learn english well ?
她讓我坐在她的身邊。	she let me sit by her .
這瓶酸奶不含乳糖。	this bottle of yogurt is lactose free .
這瓶酸奶不含乳糖。	this bottle of yogurt is lactose free .
我不能幫你了。	i couldn t able to help you .
我不能幫你了。	i couldn t able to help you .
湯姆不是一個好司機。	tom is not a good driver .
我會普通話、西南官話、吳語和西班牙語。	i can speak standard chinese putonghua , southwestern mandarin, wu chinese, and spanish .
這個問題沒有那麼簡單。	this question is not that simple .
他不會說英語也不會說法語。	he cannot speak english, neither can he speak french .
賣紅酒嗎?	sell wine ?
我又熬夜了。	i had another all nighter .
我又熬夜了。	i had another all nighter .
今天天氣怎麼樣?	what s the weather like today ?
今天天氣很好，不下雨，是晴天，不冷不熱。	the weather today s great, with no rain, clear skies, and a perfect temperature .
汤姆告诉了我们他不要了跟我们喝酒。	tom said he didn t want to go drinking with us .
她喜歡吃草莓味的酸奶。	she likes to eat and milk .
她不喜歡吃椰子味的東西。	she does not like to eat coconut flavored things .
湯姆現在就需要錢。	tom needs the money right away .
我不想讓任何人認為我傻。	i don t want everyone to think i m stupid .
湯姆沒時間完成報告。	tom didn t have time to finish his report .
湯姆沒來，簡也沒來。	tom didn t come and jane didn t come either .
他每件事做之前都會細心計劃。	he always plans a thing out carefully before he does it .
她很瘦	she s skinny

5 Summary

In this assignment, we first compared the performances of single/bi-directional GRU/RNN/LSTM models and chose the best network to proceed. Hyperparameter tuning experiments were run on hidden size and layer amount to find the best learner model under our task and data. The best performance without attention enabled has achieved a BLEU score at 0.8728 .

Among the networks we tested, GRU always has the best learning quality, whilst the RNN architecture is the worst. Using bidirectional networks significantly improves the performance of unidirectional networks. Bidirectional enhances context understanding by processing data in both forward and backward directions. The hidden state is a combination of information from both directions, enriching the model's comprehension. In speech recognition, Bidirectional RNNs capture nuanced patterns, improving accuracy.

However, still, we could notice that the translation can miss semantic meanings. One of the major improvements could be made based on how we separate tokens from the source Chinese language. Currently, we extract Chinese tokens on a byte-base, yet using third-party segmentation modules could be expected to make significant progress, since the byte-based tokens are clearly missing grammar and semantic meanings, compared to subwords.

Another enhancement would be added in the attention mechanism. At the time we handed in this assignment, we were still debugging the attention part of codes. It was a pity we did not successfully implement such a mechanism, and we are definitely going to try it out in the future.

7 References

[1] A Guide on Word Embeddings in NLP https://www.turing.com/kb/guide-on-word-embeddings-in-nlp

[2] Jieba https://github.com/fxsjy/jieba

[3] THULAC: an efficient Chinese lexical analysis toolkit http://thulac.thunlp.org/

[4] Sequence to Sequence (seq2seq) and Attention
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

[5] BLEU https://en.wikipedia.org/wiki/BLEU

[6] Understanding BLEU and ROUGE score for NLP evaluation
https://medium.com/@sthanikamsanthosh1994/understanding-bleu-and-rouge-score-for-nlp-evaluation-1ab334ecadcb#:~:text=However%2C%20it%20has%20some%20limitations,be%20unfair%20in%20some%20cases.

[7] Recurrent Neural Network, Bidirectional RNN, Deep Recurrent Networks, Recursive Neural Network, Long Term Dependencies and More
https://www.linkedin.com/pulse/recurrent-neural-network-bidirectional-rnn-deep-networks-salunke-9f66f/

[8] To what extent does the dimension of LSTM hidden states affect performance?
https://www.quora.com/To-what-extent-does-the-dimension-of-LSTM-hidden-states-affect-performance

[9] pytorch-seq2seq
https://github.com/bentrevett/pytorch-seq2seq/blob/main/1%20-%20Sequence%20to%20Sequence%20Learning%20with%20Neural%20Networks.ipynb