华南理工大学

South China University of Technology

# The Experiment Report of *Machine Learning*

**School:** School of Software Engineering

**Subject:** Electronic and Information


*Author:*
Weiwen Hu

*Supervisor:*
Mingkui Tan

*Student ID:*
202021045611

*Grade:*
Graduate


January 3, 2021

# Chinese-English Translation Machine Based on Sequence to Sequence Network

***Abstract*—In this experiment we build a sequence to sequence model to do the Chinese-English translation task, with a small-scale open dataset from Tatoeba Project.**

## I. Introduction

WE are conducting this experiment in hope of: 1. Understand natural language processing; 2. Understand the classic Sequence-to-Sequence machine translation model; 3. Master the application of attention mechanism in machine tranlstion model; 4. Build machine translation models, verify model performance on simple and small-scale datasets, and cultivate engineering capabilities; 5. Understand the application of Transformer in machine translation tasks. After the experiment, we get a basic sequence-to-sequence Chinese-English machine translation model. It can successfully translate simple Chinese sentence into English.

## II. Methods and Theory

### A. RNN

A Recurrent Neural Network (RNN) is a network that operates on a sequence and uses its own output as input for subsequent steps. There are many variants of RNN. In this experiment, we use Gated Recurrent Units (GRUs).

### B. The Sequence-to-Sequence Model

A Sequence to Sequence network, or seq2seq network, or Encoder Decoder network, is a model consisting of two RNNs called the encoder and decoder. The encoder reads an input sequence and outputs a single vector, and the decoder reads that vector to produce an output sequence. As illustrated in figure 1

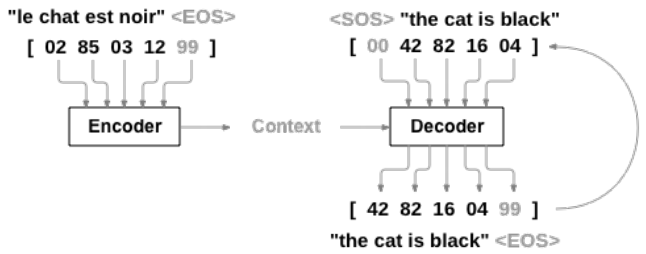


Fig. 1. Example of sequence-to-sequence model

Unlike sequence prediction with a single RNN, where every input corresponds to an output, the seq2seq model frees us from sequence length and order, which makes it ideal for translation between two languages.

With a seq2seq model the encoder creates a single vector $\mathbf{c} \in \mathbb{R}^N$ which, in the ideal case, encodes the "meaning" of the input sequence into a single vector.

### C. Attention in Decoder

If only the context vector $\mathbf{c}$ is passed betweeen the encoder and decoder, that single vector carries the burden of encoding the entire sentence.

Attention allows the decoder network to "focus" on a different part of the encoder' s outputs for every step of the decoder' s own outputs. First we calculate a set of attention weights. These will be multiplied by the encoder output vectors to create a weighted combination. The result should contain information about that specific part of the input sequence, and thus help the decoder choose the right output words.

Calculating the attention weights is done with another feed-forward layer, using the decoder' s input and hidden state as inputs. Because there are sentences of all sizes in the training data, to actually create and train this layer we have to choose a maximum sentence length (input length, for encoder outputs) that it can apply to. Sentences of the maximum length will use all the attention weights, while shorter sentences will only use the first few.

The *dot-product* attention is also very famous. It don' t have restriction on input length, and is used by the *Transformer* models. We don' t use this type of attention is this experiment though.

## III. Experiments

### A. Dataset

We use the Chinese-Engilish bilingual sentence pairs from *Tatoeba Project*[1]. It consists of 24,026 pairs. Since the dataset is small, the model is not likely to learn very complex translation. So I just used the pairs with Chinese sentence shorter than 16 characters. I randomly choose 70% (15,802 samples) for training and use the rest (6,773 samples) for testing.

### B. Implementation

I use PyTorch to implement the model and the whole pipeline. Although I referred to the official tutorial[2], there are still some major differences: First, My model can

[1]https://www.manythings.org/anki/cmn-eng.zip

[2]https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

process a batch of sentence pairs, which greatly improves the training efficency. And this should also improve the training result stability, since we can use mini-batch SGD to work with more accurate gradient. Second, I implemented a slightly more complicated tokenizer, which can handle punctuations, and some more special cases. Last, My pipeline is fully distributed. It can utilize multiple GPUs to accelerate the training.

For Chinese sentence tokenization, I just split each character as a token. And I also did some normalization on punctuations.

The BLEU Implementation is from torchtext package.

All parameters are initialized using default method in PyTorch. I use Adam optimizer with default parameters (learning rate 1e-3, $\beta$ 0.9, 0.999) other hyper-parameters are summarized in table I The whole training process only takes 3 minutes on 2 GeForce GTX TITAN X GPU.

TABLE I
HYPER PARAMETERS

| | |
|---|---|
| Epoch | 50 |
| Batch size | 512 |
| Dropout | 0.1 |
| Encoder hidden size | 512 |
| Decoder hidden size | 512 |

### C. Results

The experiment result is summarizes in table II

TABLE II
RESULTS

| | Train | Test |
|---|---|---|
| loss | 0.0433 | 6.8399 |
| token accuracy (%) | 98.79 | 34.54 |
| BLEU | - | 0.2011 |

Very different from the previous linear classification experiment, this experiment shows severe overfitting. As illustrated in figure 2 and figure 3, The training loss is 2 orders of magnitude smaller than the test loss, and the token accuracy on training samples is nearly 100%. This could be because the model is more complicated, and is capable to "remember" all training samples; and because the translation task is more complicated while the amount of training data is too few.

Here I show some of the translation results in test set:

Success samples:

- 湯姆拍了手。
  tom clapped .
- 你喜歡這些照片里的任何一張嗎?
  do you like any of these pictures ?
- 請進來。
  please come in .

Not accurate but make some sense:

- 他偷了她的手表。
  he had her arm .
- 這不便宜，是嗎?
  it 's not warm , isn't it ?

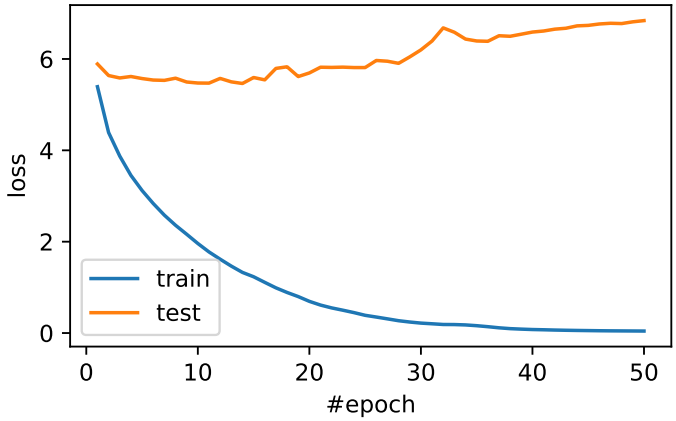


Fig. 2. Loss change with number of epoch

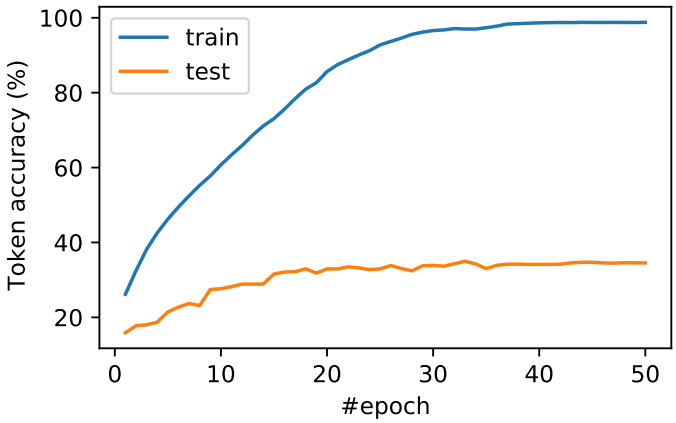


Fig. 3. Token accuracy change with number of epoch

- 我们现在就去医院吧。
  let 's go out the right away .

Failure samples:

- 参见上文。
  the man to the tree .
- 你的包开着。
  your team is right .
- 雪停了。
  it 's 3:30 .

Some failure is caused by perticular words not recognized by the model, the overall sentence structure is fine. But there are also some failures not making any sense to me.

## IV. CONCLUSION

This experiment gives satisfying results. The 3-minute training and 1.6k training bilingual sentence pairs give a not-so-bad machine translation model. It can handle simple sentences. In this experiment, I'm familiarised with the sequence-to-sequence model as well as other parts of the machine learning pipeline. And my engineering skills are improved.