

# Homework 4 Part 2 (Summer Version)

## MACHINE TRANSLATION WITH A SEQUENCE TO SEQUENCE NETWORK

11-785: INTRODUCTION TO DEEP LEARNING (FALL 2019)

OUT: **June, 2019**

DUE: **September, 2019**

### Start Here

- **Collaboration policy:**

- You are expected to comply with the University Policy on Academic Integrity and Plagiarism.
- You are allowed to talk with / work with other students on homework assignments
- You can share ideas but not code, you must submit your own code. All submitted code will be compared against all code submitted this semester and in previous semesters using MOSS.

- **Overview:**

- **Part 2:** All of the problems in Part 2 will be tested by yourself. You can download the starter code from Course Page.

## 1 Introduction

In part 2 of homework 4, we will be teaching a neural network to translate from French to English. You will practice using a sequence to sequence model. In 11785, Part 2 of homework 4 is the most difficult assignment among all the homeworks. In the coming semester, you not only need to design a seq2seq model yourself, but also implement the attention modules successfully. We strongly recommend that you should finish this part by yourself. **Please Do Not Copy the Codes From the Original Website!**

Here are the papers and blogs we recommend you to read: (By the way, if you find yourself hard to comprehend these readings, please do not worry. You will master these ideas at the end of this semester.)

1. Translation with a Sequence to Sequence Network and Attention This is the original website about this assignment. We have given some hints for the codes you need to write.
2. Neural MT 1: Neural Encoder-Decoder Models for the basic ideas of Seq2Seq model.
3. Here is a blog about how to use teacher forcing in RNNs.

This assignment would help you be familiar with the idea of sequence to sequence model, which is extremely important.

### 1.1 Files

The template provided to you is a Jupyter notebook. There are TODO sections in the notebook that you need to complete. The notebook is based on the original Pytorch tutorial, which forces you to think about the codes behind. **DO NOT COPY THE CODES** from the original website.

Your solutions will be tested locally by your self.

## 2 Dataset

This dataset contains many thousands of English to French translation pairs.

- `eng-fra.txt`: a txt file containing your training data
- `names`: a directory containing lots of names in different languages. Refer to CLASSIFYING NAMES WITH A CHARACTER-LEVEL RNN.

### 2.1 Function `prepareData`

We do not have a traditional `DataLoader` Class here in this tutorial. Here we use the `prepareData` function to deal with the data. As you have seen how to overwrite some methods in Pytorch's `DataLoader`, we could also use function to feed the training data to our models instead.

Here, the tutorial has finished most of the parts for us. However, you need to (in the Class `Lang`):

1. Figure out how to add the SOS and EOS indexes successfully.
2. And how to specify the order of SOS and EOS in your language model.

EOS and SOS are extremely important in Seq2Seq models. EOS will be appended to the end of the input sequence, which will provide a consistent signal for the model to learn when to stop the generations for arbitrary-length sequences. SOS will be fed into decoder as the beginning signal.

## 3 Model

Traditionally speaking, there are always two parts in Seq2Seq model: an Encoder and a Decoder. The encoder would help you encode the inputs to the context information. The decoder will use this context information to generate the output sequence. Sequence-to-Sequence model, i.e., Seq2Seq was first introduced in the paper Sequence to Sequence Learning with Neural Networks.

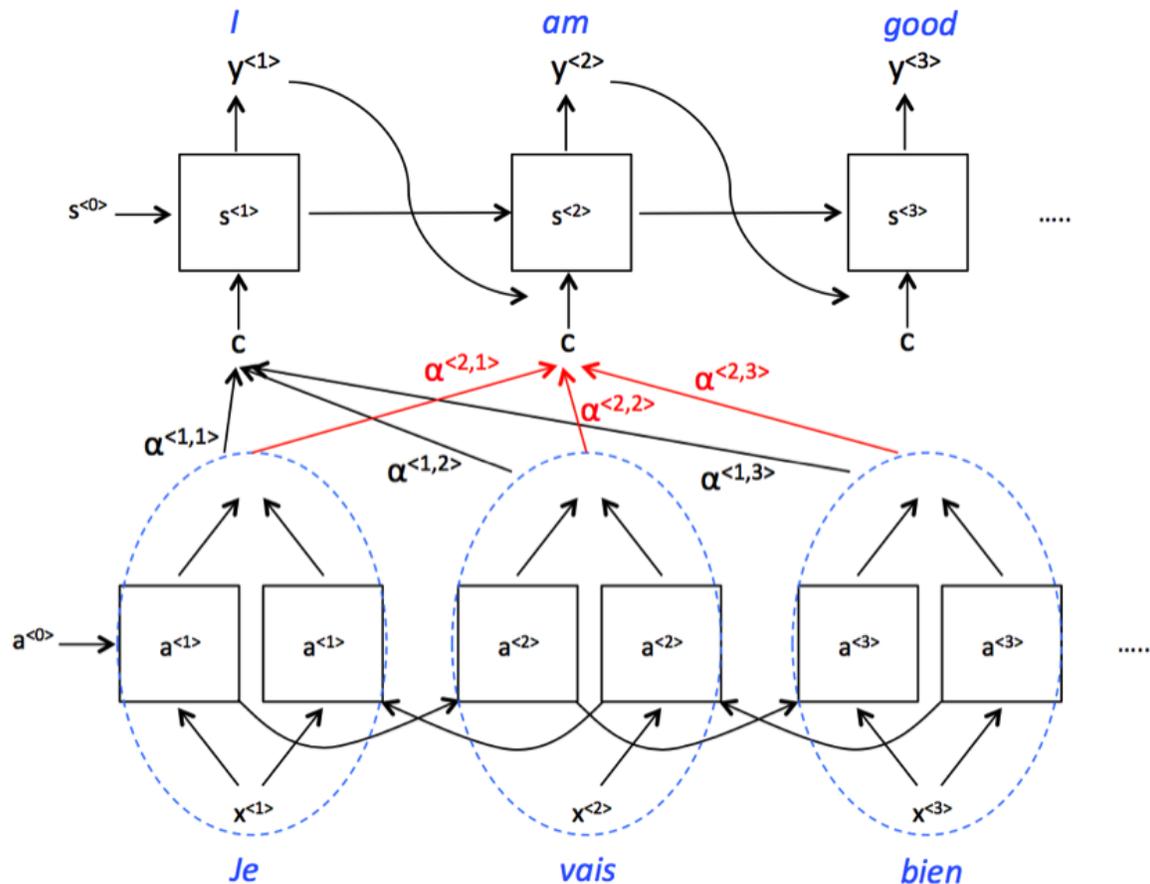
However, the context information could be calculated dynamically. In the paper NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE, the attention mechanism was first introduced. The context information is not constant any more and it is based on the similarity scores between the hidden states of the encoder and hidden state of the decoder.

### 3.1 The Encoder

You could learn more ideas about the encoder from this note: Neural Machine Translation, Seq2seq and Attention.

The Encoder is just a stack of LSTM or GRU cells. In translation problem, the encoder would take a sequence as the input. In conversation problem, the encoder would take the statement or question as the input. In the Encoder part, you are required to finish these parts:

1. Feel free to try different Recurrent layers, such as RNN, LSTM and GRU.
2. Figure out how to use the embedding layer.



### 3.2 The Decoder

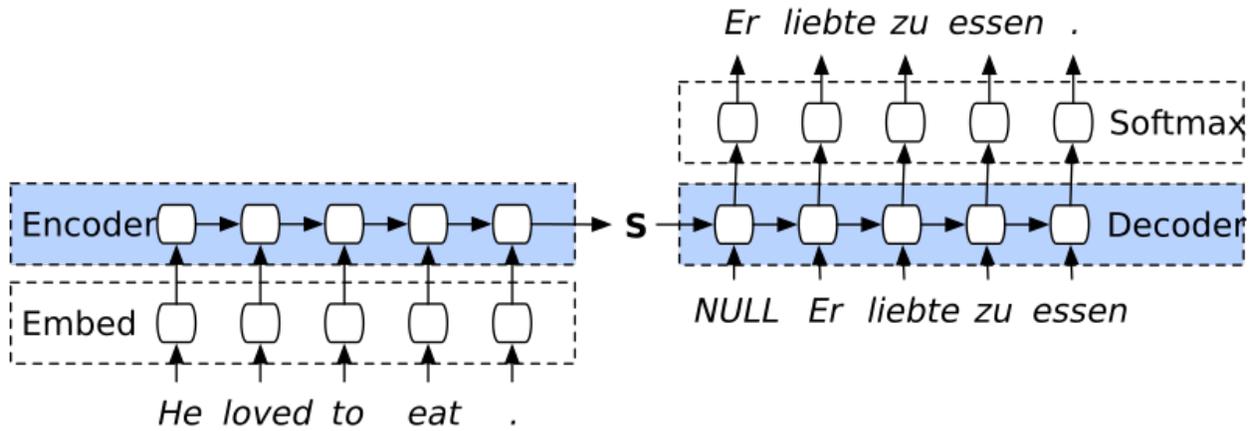
The decoder is a stack of RNN units. The decoder would use three parts: the output from the last step, the hidden states from last step and the outputs from the encoder. You could learn more from these two graphs. If we choose to use teacher forcing, the output from the last step might be replaced by the ground truth label.

Here are the differences between traditional Decoder and Attention based Decoder:

1. Rather than using the final RNN hidden state as the single "context vector", we could specifically provide the decoder with the entire input sequence information for every decoding step. The context vector would be calculated based on different similarity scores between the inputs and the outputs.  
Note: You could learn more about the Attention part: [The Illustrated Transformer](#)
2. In pytorch 1.1.0, there is a class called: `torch.nn.MultiheadAttention`. You could use it directly. We strongly recommend that way.

In the Decoder part, you need to finish the following parts:

1. Feel free to try different modules, such as RNN, LSTM and GRU, just like what have been described in the Encoder part.
2. Figure out how to use the embedding layer.
3. Figure out how to specify the softmax layer in the attention model.



### 3.3 The Loss

The loss is the traditional cross entropy loss. In pytorch, the difference between `nn.CrossEntropyLoss` and `nn.NLLLoss` is that `nn.CrossEntropyLoss` combines `nn.LogSoftmax()` and `nn.NLLLoss()`. The way that this tutorial does is to calculate the log softmax at the end of output, which is not recommended.

## 4 Questions

Here are the questions you need to answer successfully after you finish this summer version homework 4 part 2.

1. Is there any difference between LSTMcell and LSTM with the time step equal to 1?
2. Which is better: choose to use the same optimizer for the encoder and the decoder or just use different optimizers just like what have been mentioned in this tutorial?
3. Why is the attention map diagonal?
4. When it comes to the teacher forcing, which is better: mix the ground truth labels with the predictions or just use all the ground truth labels or predictions separately just like what was used in this tutorial?
5. How to combine the output from the last step and the outputs from the encoder?

Good luck! May the global optimum be with you!