Report
Seq2Seq Machine Translation with Attention

Lyle He

Contents

1 Seq2Seq Machine Translation with Attention 2
1.1 Introduction 2
1.2 Model Architecture 2
1.3 Dataset and Preprocessing 4
1.3.1 Data Preprocessing Results 4
1.4 Training 5
1.4.1 Hyperparameters Search 5
1.4.2 Final Trainning Results 7
1.5 Translation and Evaluation 8
1.5.1 Translation Example 8
1.6 Analysis 9
1.6.1 Beneficial of Attention Mechanism 9
1.6.2 Comparison with Seq2Seq without Attention 10
1.6.3 Final Comparison Result 12
2 References 12

1 Seq2Seq Machine Translation with Attention

1.1 Introduction

In this part, I have implemented a seq2seq model with attention for machine translation between English and French and analyzed the beneficial of the attention mechanism and compared the performance of the seq2seq model with attention and the seq2seq model without attention.

In addition, my project code is organized as follows, the first two python notebook files are used to run the model in kaggle to utilize the GPU resources and the last one is the main project code for the seq2seq model.

1. python notebook file: `neuralmachinetranslation_hyperparameter_grid_search.ipynb`
2. python notebook file: `neuralmachinetranslation_final_training.ipynb`
3. pure python project: (this is the main project code for the seq2seq model with attention for machine translation between English and French)
   - ./Seq2Seq-Attention-Model
     - data_preprocessing.py
     - eng_-french.csv (you can download from kaggle[3])
     - model.py
     - train_test_the_model.py
     - utils.py
     - run.py

The model architecture, dataset, preprocessing, training, translation, and evaluation are shown in the below.

1.2 Model Architecture

According to the assignment requirement, I have implemented a seq2seq model with attention for machine translation between English and French. The model architecture is shown in the below figure and text description.

```
NMTModel(
  (embeddings): WordEmbeddingForTranlationTask(
    (source): Embedding(14516, 512, padding_idx=0)
    (target): Embedding(28336, 512, padding_idx=0)
  )
  (encoder): LSTM(512, 512, bidirectional=True)
  (decoder): LSTMCell(1024, 512)
  (encoder_hidden_to_initial_decoder_hidden):
    Linear(in_features=1024, out_features=512, bias=False)
  (encoder_cell_to_initial_decoder_cell):
    Linear(in_features=1024, out_features=512, bias=False)
  (encoder_hidden_to_decoder_hidden_for_attention):
    Linear(in_features=1024, out_features=512, bias=False)
  (combined_output_and_hidden_to_hidden):
    Linear(in_features=1536, out_features=512, bias=False)
  (target_vocab_projection):
    Linear(in_features=512, out_features=28336, bias=False)
  (dropout): Dropout(p=0.5, inplace=False)
)
```

As we can see from the above code, the model consists of an embedding layer (for both source and target languages), an encoder, a decoder, and several linear layers for dimension prejection. The encoder

is a bidirectional LSTM layer, and the decoder is a LSTMCell layer. The attention mechanism is implemented according to the lectures as shown in the Figure 1.

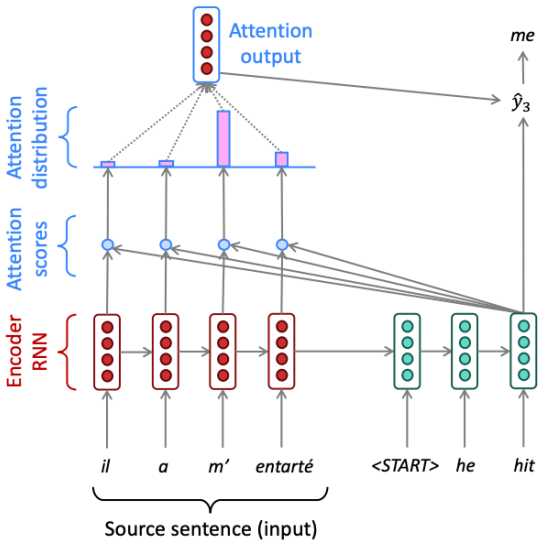

Figure 1: Seq2Seq Model with Attention [1]

The math equation for this model is from the lectures and the attention mechanism is according to Global Attention Model (Luong, et al. 2015)[2]. Below are the key python code implementation for this model.

```
# encoder computation
encoder_hiddens, (last_hidden, last_cell) = self.encode(input_sequences)

# last hidden and cell projection
initial_decoder_hidden = self.encoder_hidden_to_initial_decoder_hidden(last_hidden)
initial_decoder_cell = self.encoder_cell_to_initial_decoder_cell(last_cell)

# decoder computation
combined_outputs = self.decode(target_sequences, encoder_hiddens, initial_decoder_hidden, initial_decoder_cell)

# project the combined outputs to the target vocabulary
combined_outputs = self.dropout(combined_outputs)
combined_outputs = self.target_vocab_projection(combined_outputs)
P = F.log_softmax(combined_outputs, dim=-1)
```

Figure 2: Seq2Seq Model Forward Processes

```
for word_embedding_at_t in torch.split(embeddings, 1, dim=0):  for loop to predict one word for one time step
    # shape of word_at_t: (1, batch_size, embedding_dim)
    word_embedding_at_t = word_embedding_at_t.squeeze(0) # remove the dimension of 0
    # shape of word_at_t: (batch_size, embedding_dim)

    word_embedding_cat_prev_output_at_t = torch.cat((word_embedding_at_t, prev_output), dim=1)

    # lstm cell computation and attention computation
    combined_output_t,  dec_state, _ = self.step(word_embedding_cat_prev_output_at_t, encoder_hiddens, encoder_hiddens_for_attention, decoder_hidden_t, decoder_cell_t)
    decoder_hidden_t, decoder_cell_t = dec_state
    combined_outputs.append(combined_output_t)
    prev_output = combined_output_t
combined_outputs = torch.stack(combined_outputs)

return combined_outputs
```

Figure 3: Seq2Seq Model Forward Processes

```python
def step(self, features_t, encoder_hiddens, encoder_hiddens_for_attention, prev_hidden, prev_cell):
    3. compute the context vector
    4. compute the combined output

    @param features_t (torch.Tensor): input features, shape: (batch_size, feature_dim)
    @param encoder_hiddens (torch.Tensor): output of the encoder (the excat hidden states of each time step), shape: (batch_size, src_sequence_length, hidden_dim)
    @param encoder_hiddens_for_attention, shape: (batch_size, src_sequence_length, hidden_dim)
    @param prev_hidden (torch.Tensor): previous hidden state of the decoder
    @param prev_cell (torch.Tensor): previous cell state of the decoder
    """
    decoder_hidden_t, decoder_cell_t = self.decoder(features_t, (prev_hidden, prev_cell)) # shape of decoder_hidden_t: (batch_size, hidden_dim)

    if self.has_attention:
        # compute the attention score, use encoder_hidden_t to dot product with encoder_hiddens_for_attention  computing as shown in the above model graph
        attention_score = torch.bmm(decoder_hidden_t.unsqueeze(1), encoder_hiddens_for_attention.permute(0, 2, 1)).squeeze(1) # shape: (batch_size, src_sequence_length)
        attention_score_after_softmax = torch.softmax(attention_score, dim=1)

        # (batch_size, 1, src_sequence_length) * (batch_size, src_sequence_length, hidden_dim*2) -> (batch_size, 1, hidden_dim*2)
        src_context_vector = torch.bmm(attention_score_after_softmax.unsqueeze(1), encoder_hiddens).squeeze(1)


        # concatenate the output of the decoder and the context vector
        combined_output = torch.cat((decoder_hidden_t, src_context_vector), dim=1)  concatenent the attention context with the decoder hidden for final hidden

        # project the combined output to the hidden_dim
        final_features = self.combined_output_and_hidden_to_hidden(combined_output)

        return final_features, (decoder_hidden_t, decoder_cell_t), attention_score_after_softmax
```

Figure 4: Seq2Seq Model Forward Processes

In the above code, the key part is the `decode` function. We need to loop through the target sequence and compute the attention mechanism at each time step. Before we compute the attention mechanism, we need to concatenate the word embedding at time t and the previous output and input it into the LSTM cell. Then we can compute the attention mechanism and the combined output at time t. The combined output will be the input for the next time step. The `step` function is the implementation of the attention mechanism.

The `test` function is the function to generate the translated text which is similar to the `decode` function but with a different loop condition and we use the generated word at time t as the input for the next time step instead of the word embedding at time t.

1.3 Dataset and Preprocessing

I choose a medium-sized dataset for this task from kaggle[3]. There are 175621 pairs of English and French sentences. The vocabulary size for English is 14516 and for French is 28336.

1. I remove the non-meaningful characters (e.g. non-breaking space, etc.)
2. remove accents of the French sentences, and convert all the characters to lowercase.
3. Then I tokenize the sentences, get the vocabularies of the source and target languages respectively, add ‘start’, ‘end’, ‘unknown’ , and ‘padding’ tokens to the vocabularies
4. convert the sentences to sequences of integers.
5. I pad the sequences to the same length and split the dataset into training and validation sets.
5. The result is shown in the python notebook file.

1.3.1 Data Preprocessing Results

```
number of samples 175621
characters in English sentences
 {'N', '"', 'e', '0', 'p', 'b', '%', 'P', 'é', '—', 'a', '-', 'r', 'T', '7', 'M', '2',
'y', 'W', 'Z', 'c', 'ç', 's', 'h', '8', 'O', 'd', 'Q', 'k', 't', 'ö', ',', 'L', 'C',
'4', '&', '\xa0', 'a', '€', 'D', 'I', 'u', 'Y', 'U', 'l', 'z', 'x', 'v', '’', 'n', '.',
'K', 'E', 'ú', 'R', 'F', '1', '/', ' ', 'j', 'i', '!', '6', '?', 'G', 'g', "'", 'B',
'5', 'f', '–', 'w', '+', 'V', 'X', 'A', '3', 'º', 'o', 'm', ':', 'q', 'S', '²', '$',
';', 'H', '‘', '\xad', 'J', '9'}
characters in French sentences
 {'è', 'N', '"', 'e', 'À', '0', 'p', 'b', '%', 'P', 'é', '-', 'r', 'T', '7', 'M', '2',
'y', 'W', 'Z', 'c', 'ç', 'ê', 's', 'h', '8', 'd', 'O', 'Q', 'k', 't', 'ö', '\u202f',
',', 'L', 'C', 'É', '4', '&', '\xa0', 'a', 'D', 'ô', 'I', 'î', 'u', 'U', 'Y', 'l', 'z',
'«', 'x', 'v', '’', 'n', 'C', '…', '.', 'ù', 'Ê', ')', 'K', 'E', '\u2009', 'R', '»',
```

```
'1', 'F', '/', '(', ' ', 'j', 'i', 'Ô', '!', '\u200b', '6', '?', 'Ç', 'G', 'g', "'",
'â', 'B', '5', 'œ', 'ë', 'f', '—', 'w', '+', 'V', 'X', 'A', '3', 'û', 'o', 'á', 'm',
':', 'à', '?', 'q', 'S', '₂', 'ï', '$', 'Â', ';', 'H', '‘', 'J', '9'}
length of Engish characters in English sentences 91
length of French characters in French sentences 113
Remove the unmeaningful character and convert to uppercase, and keep any meaningful
characters
length of en character set after cleaning 51
length of fr character set after cleaning 55
Tokenize the sentences
English-French pairs Examples
Src: hi.  Tgt: salut!
Src: run! Tgt: cours !
Src: run! Tgt: courez !
Src: who? Tgt: qui ?
Src: wow! Tgt: ca alors !
Src: fire!  Tgt: au feu !
Build the vocabulary
English vocabulary Examples
<SOS> : 2
hi  : 3
. : 4
...
length of fr_vocab, 28336
number of samples - en_sequence:  175621
number of samples - fr_sequence:  175621
data preprocessing done
```

Figure 5: Data Preprocessing Results

1.4 Training

1.4.1 Hyperparameters Search

- Different Batch size: 32, 64, 128, 256

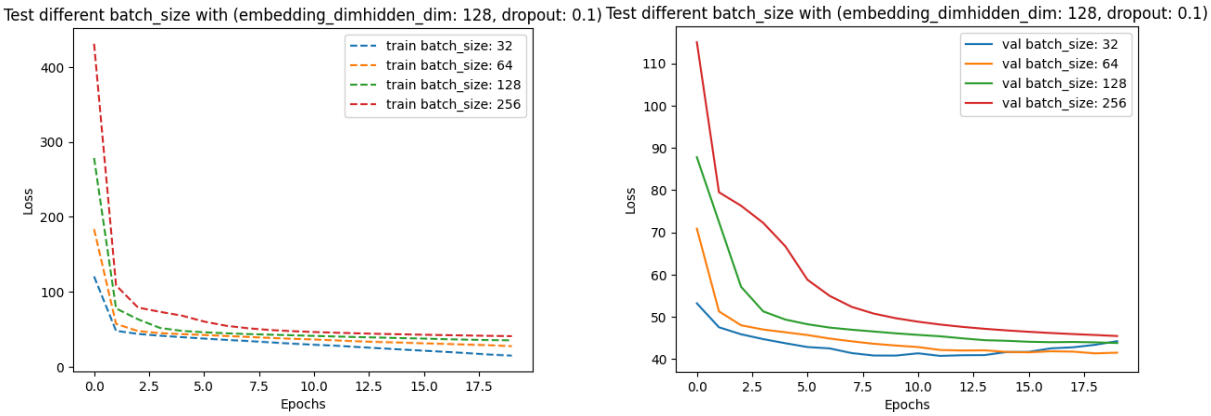

Figure 6: Different Batch size

As we can see the difference between different batch sizes is not significant. The batch size of 128 is chosen for the final model.

- embedding_dim: 128, 256, 512

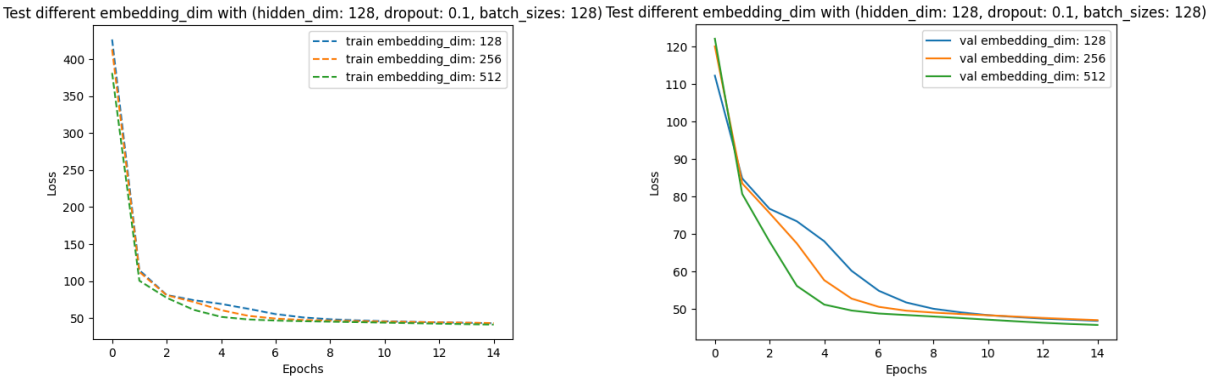

Figure 7: Different Batch size

As we also could see the difference between different embedding dimensions is not significant. The embedding dimension of 512 is chosen for the final model.

- hidden_dim: 128, 256, 512

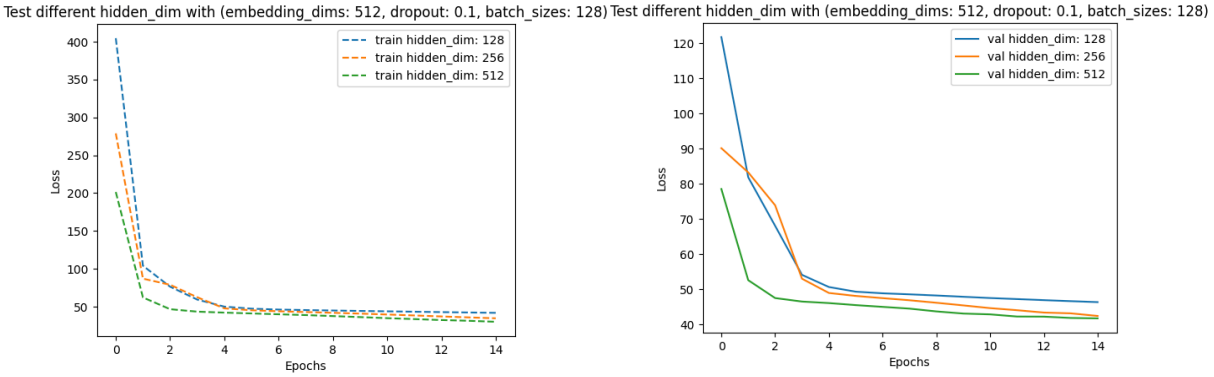

Figure 8: Different Batch size

As we can see the 512 seems to have a better performance than the other two. So I choose the hidden dimension of 512 for the final model.

- dropout_rate: 0.1, 0.2, 0.5

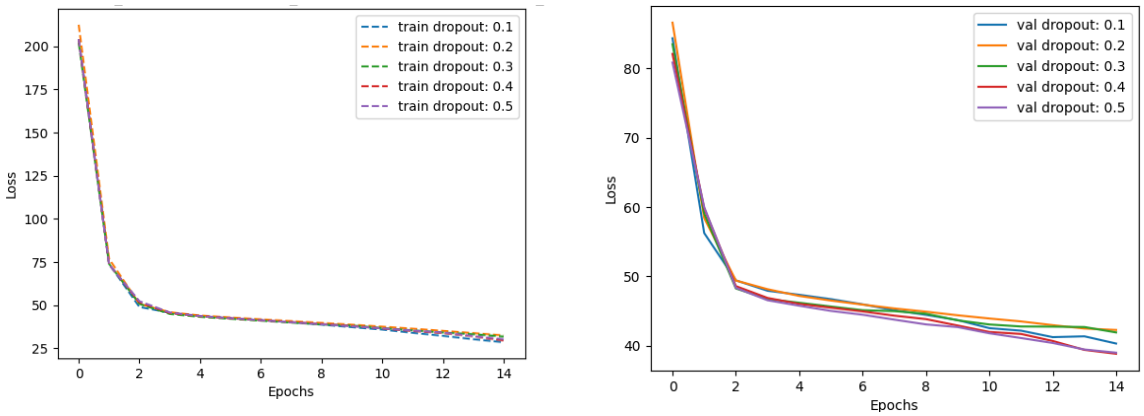

Figure 9: Different Batch size

As we can see the dropout rate of 0.5 seems to have a better performance than the other two. So I choose the dropout rate of 0.5 for the final model.

1.4.2 Final Trainning Results

After tested different hyperparameters and the best hyperparameters are shown in the below table. The training process is shown in the python notebook file.

Hyperparameters	Value
Batch size	128
embedding_dim	512
hidden_dim	512
Learning rate	0.001
dropout	0.5
Epochs	10
Optimizer	Adam
Loss function	negative log probability loss

Here is the loss curve of the training process.

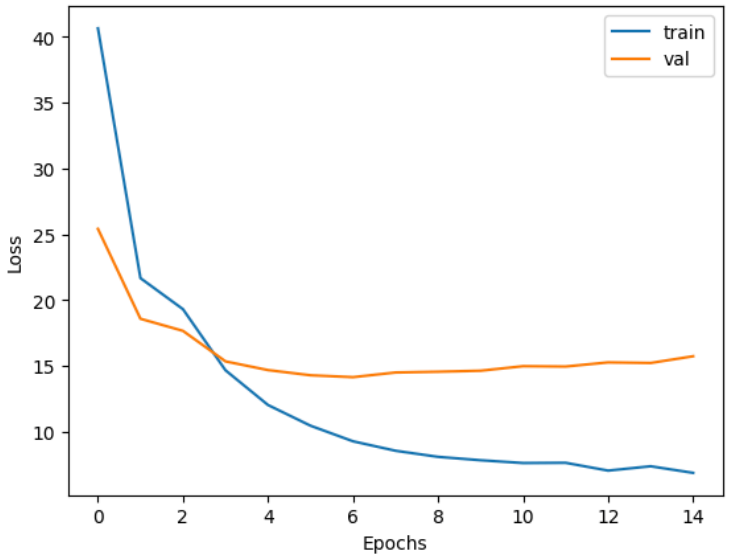

Figure 10: Loss Curve

1.5 Translation and Evaluation

According to the assignment requirement, I tested the model in my test set and the result is shown in the below table.

Model	BLEU Score
Seq2Seq with Attention	0.29

1.5.1 Translation Example

I also show an example of the translation result in the below.

Src: What is your name ?
Tgt: comment vous nom ?

Src: I am doing great
Tgt: je suis bon en train de faire .

Src: What are you doing today ?
Tgt: que faites-vous aujourd'hui ?

Src: Can you help me with my homework ?
Tgt: pouvez-vous m'aider a mes devoirs ?

Src: I am a student at the university and I am studying computer science
Tgt: je suis etudiant a l'universite et j'ai etudier le ordinateur .

Src: In my opinion , the best way to learn a new language is to practice speaking with native speakers
Tgt: selon mon avis , le meilleur moyen est de pratiquer avec ce qui est a parler de parler un truc de monde a apprendre le pays de parler avec ce qui est a parler de parler .

Src: I usually wake up at 6 am and then I go for a run in the park before I start working
Tgt: je me leve habituellement au restaurant et puis je commence a travailler dans le parc avant .

- In order to let you know the translation result, I also show the inverted translation result in the below.

Src: What is your name?
Tgt: what is your name?

Src: I am doing great
Tgt: I'm good at doing.

Src: What are you doing today?
Tgt: what are you doing today?

Src: Can you help me with my homework?
Tgt: can you help me with my homework?

Src: I am a student at the university and I am studying computer science
Tgt: I am a university student and I studied computer science.

Src: In my opinion, the best way to learn a new language is to practice speaking with native speakers
Tgt: in my opinion, the best way is to practice with what is to speak of speaking a world thing to learn the country of speaking with what is to speak of speaking.

Src: I usually wake up at 6 am and then I go for a run in the park before I start working
Tgt: I usually get up at the restaurant and then I start working in the front park.

As we can see from the translation result, the translated French sentences are almost the same as the target sentences. So, basically, the model can translate the simple English sentences to French sentences correctly.

1.6 Analysis

In this section, I will analyze the beneficial of the attention mechanism and compare the performance of the seq2seq model with attention and the seq2seq model without attention. The result is shown in the below.

1.6.1 Beneficial of Attention Mechanism

I recorded the attention result of the some of the translated sentences (shown above) and the result is shown in the below.

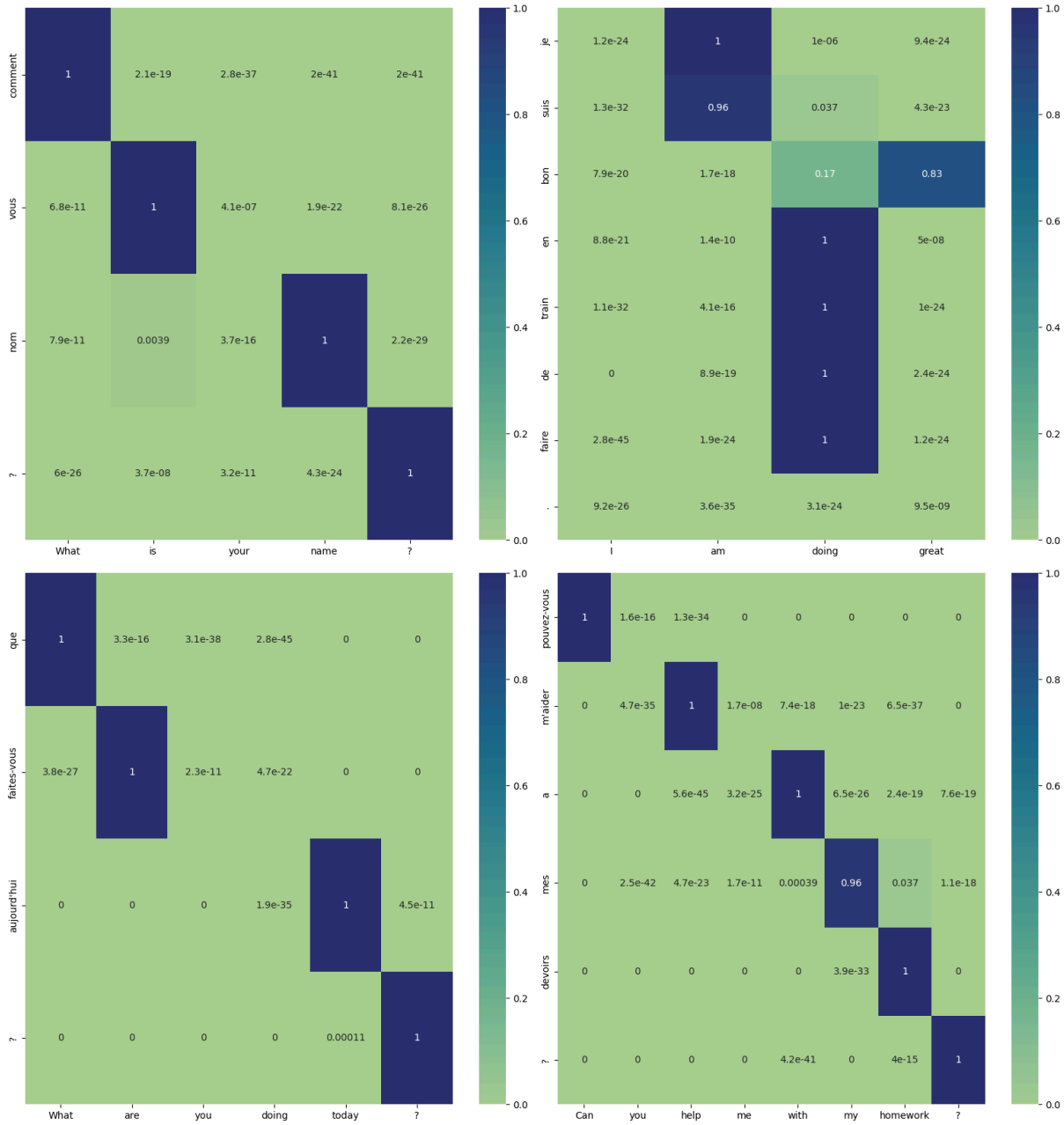

Figure 11: Different Batch size

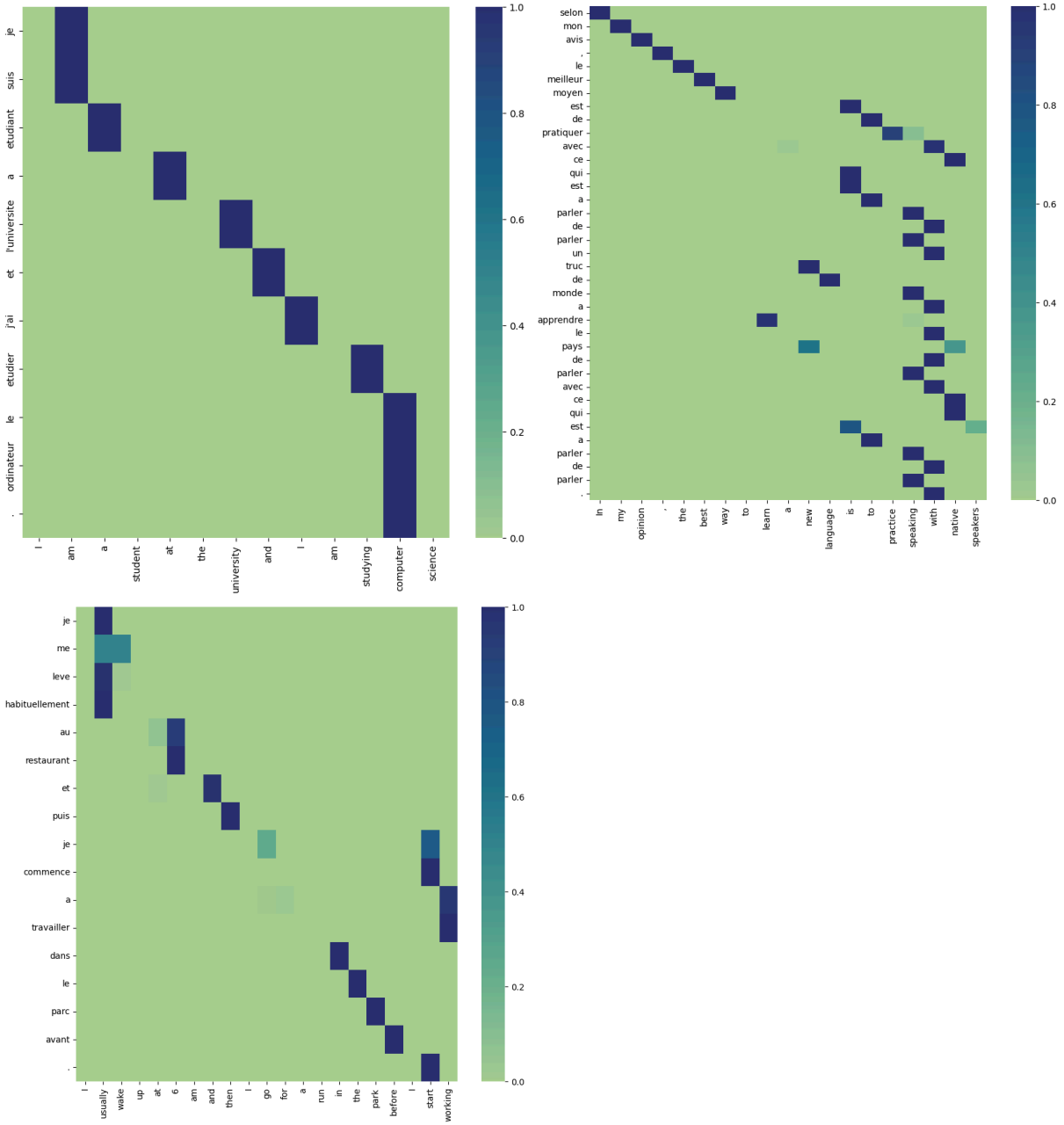

Figure 12: Different Batch size

Obviously, as we can see the attention mechanism can help the model to focus on the related parts of the input sequence and generate the translated text.

For example, in the first sentence, the attention mechanism can help the model to extract the corresponding French words for the English words.

1.6.2 Comparison with Seq2Seq without Attention

I implemented another seq2seq model without attention and compared the performance of the two models. Here is the result.

Src: What is your name?
Tgt: si nom ?

Src: I am doing great
Tgt: je ne suis pas un comme un .

Src: What are you doing today?
Tgt: ce que est-ce que tu manges ?

Src: Can you help me with my homework?
Tgt: si quelqu'un avec moi d'acquerir .

Src: I am a student at the university and I am studying computer science
Tgt: un jour , je ne suis pas sur de chouette et de meme .

Src: In my opinion, the best way to learn a new language is to practice speaking with native speakers
Tgt: si que ce qui sache , c'est quelqu'un de tres causer des conseils , que l'un de mes parents reviennent .

Src: I usually wake up at 6 am and then I go for a run in the park before I start working
Tgt: je me fiche que je vais m'allonger au tennis avant qu'ils ne soient pas bien sous une heure .

And the in order to let you know the translation result, I also show the inverted translation result in the below.

Src: What is your name?
Tgt: if name?

Src: I am doing great
Tgt: I'm not one like one.

Src: What are you doing today?
Tgt: what are you eating?

Src: Can you help me with my homework?
Tgt: if anyone with me to acquire .

Src: I am a student at the university and I am studying computer science
Tgt: one day, I'm not sure if it's cool or the same.

Src: In my opinion, the best way to learn a new language is to practice speaking with native speakers
Tgt: if anyone knows, it's someone who can give advice, that one of my parents comes back.

Src: I usually wake up at 6 am and then I go for a run in the park before I start working
Tgt: I don't care if I'm going to lie down at tennis before they're not well under an hour.

And the translating loss curve is shown in the below.

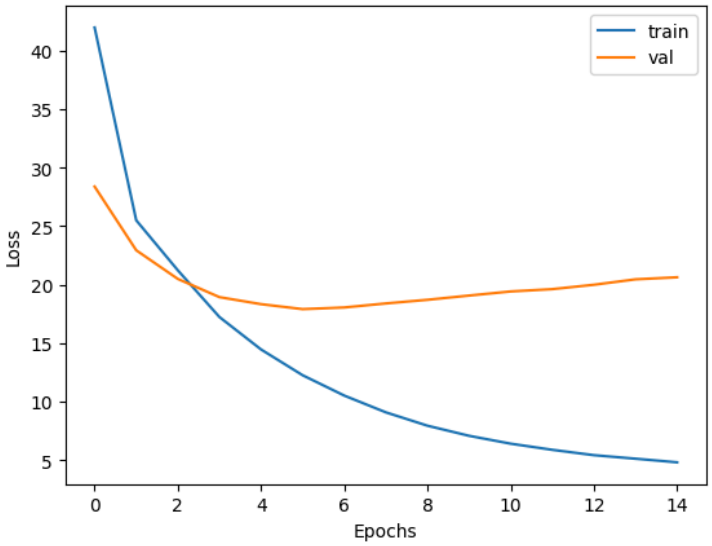

Figure 13: Loss Curve

The result of model without attention is abovioulsy worse than the model with attention. The translated French sentences are not that coherent.

1.6.3 Final Comparison Result

Here is the final comparison result of the two models.

Model	BLEU Score
Seq2Seq with Attention	0.29
Seq2Seq without Attention	0.21

As we can see from the above table, the seq2seq model with attention has a much better performance than the model without attention. The attention mechanism can help the model to focus on the important parts of the input sequence and improve the translation performance.

2 References

[1] https://arxiv.org/pdf/1409.0473.pdf [2] https://arxiv.org/pdf/1508.04025.pdf [3] https://www.kaggle.com/tilii7/englishfrench [4] https://www.gutenberg.org/cache/epub/84/pg84-images.html