Translation English to French

Mohamad Lakkis

November 10, 2024

LINK TO COLLAB (for the models (files .pth) please reach out to me, as they are too big to be uploaded to the colab)

Abstract

This document explains the model used to translate English text to French text, presenting the model's results and discussing its limitations. This study provides insights for understanding sequence-based NLP tasks, particularly in the context of translation using recurrent neural networks.

1 Introduction

This translation model was trained in two different settings: one using unidirectional and the other using bidirectional LSTM networks to convert English text to French. Although this basic LSTM architecture produces reasonably not bad results, we explore potential improvements using an attention mechanism, which we leave for future work. For now, we focus on LSTM networks with multiple layers, as will be discussed later.

The goal of these simulations is to enable the model to grasp the semantic meaning of phrases, rather than performing a simple word-by-word translation. To accomplish this, we selected an encoder-decoder architecture: the encoder captures the semantic structure and order of the words, while the decoder generates the French translation. In future work, we plan to enhance this model by introducing an attention mechanism and expanding the dataset.

2 Methodology

In this section, we describe the model's architecture and the training process. The model consists of an encoder and a decoder, both of which are LSTM networks. The encoder processes the input sequence, while the decoder generates the output sequence. The model is trained using the Adam optimizer and the categorical cross-entropy loss function.

2.1 Data Preprocessing

As a first step, we need an english and french vocabulary, so we tokenize the input sentences (in this case we used a tokenizer for each word, from NLTK library) and then formed the english vocabulary, from these disticnt tokens. After that we start indexing each token in the input sequences. At this step each token is represnted by an integer, so consequently each sentence is represented by a sequence of integers.
The same prcoess is done for getting the french vocabulary and indexing the output sequences.
Additionally, we will also get the inverse maping of the french and english vocabs, to be able to convert the output sequences from the model to the corresponding words.
Now in each of the two vocabs (english and french) we will add to them 4 speical tokens:

- <PAD> token: to pad the sequences to have the same length.
- <UNK> token: to represent the unknown words.
- <EOS> token: to represent the end of the sentence.
- <SOS> token: to represent the start of the sentence.

Now we for the output sequences we will add the <SOS> token at the start of each sequence and the <EOS> token at the end of each sequence.
Note: we will not add these tokens to the input sequence.
Furthermore, we will pad the input sequences to have the same length (as maximum lenth of the input sequences) so any input sequence that is shorter than the maximum length will be padded with the <PAD> token. And any input sequence that is longer than the maximum length will be truncated.
Similar steps will be done for the output sequences.
Not that for the input sequence we will not be adding <EOS> and <SOS> tokens. We will use another method to process only the unpadded part of the sequence, and for that we will need the src_lenth, which will denote the actual lenth of the each sequence before padding.
After this step our inputs and expected outputs are ready to be fed to the model.
So to wrap up: Our inputs for the models will be the padded input sequences and the src_length. As for its output (during training) will be probablity distribution over the french vocab, as for the output during inference it is just the class (the french token, that is most probable).

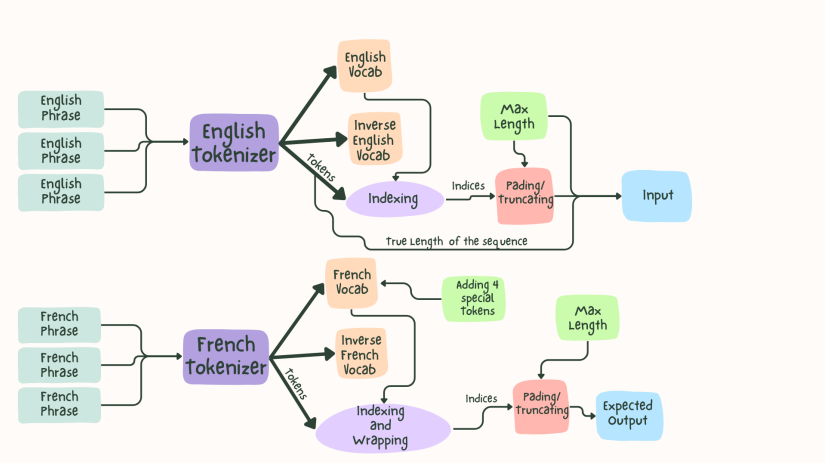

Figure 1: Data Preprocessing

Important Note: Both Inputs and Outputs are common for both models.

2.2 Model 1 Architecture: One-Directional LSTMs

Now that we understand the data preprocessing, we can move to the model architecture.
I think it is better now to look at the general hierachy of the model through the following figure.

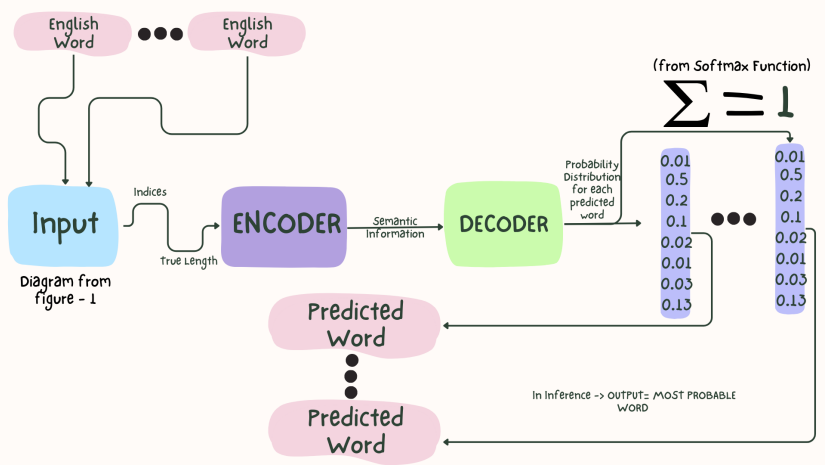

Figure 2: General Overview of the Model

2.2.1 Encoder

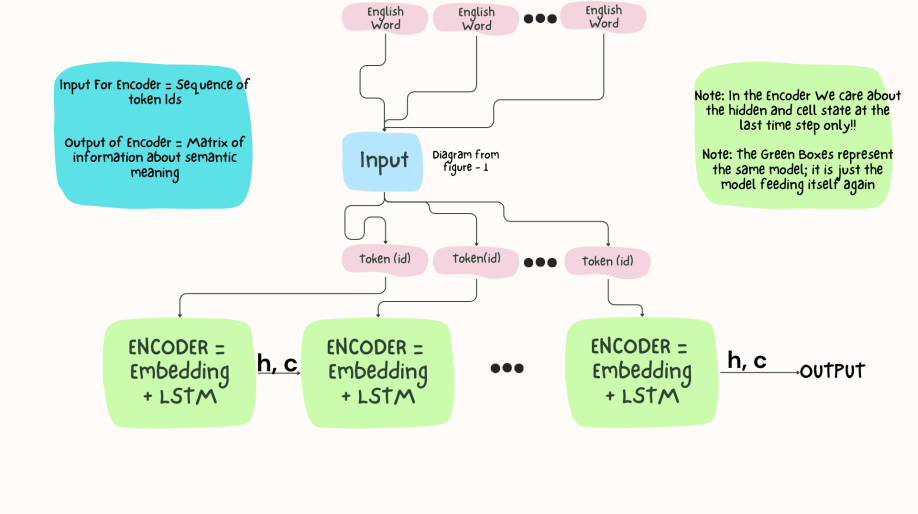

Figure 3: Encoder

The architecture of the encoder is summarize in Figure 3. The encoder is a unidirectional LSTM, its goal is encode the inputs, and make semantic meaning of the phrase, then it will pass this information to the decoder to generate the output.
So in general sense, the encoder will take each input token id on its own (and in order), apply to it an embedding layer and then pass it to LSTM cell.
Then this LSTM cell will output the hidden state and the cell state, which will be passed to the same model but for the next token id.
So in other words it is the same model that we are working with but we are passing each input token one by one, and after each pass we get another hidden state and cell state, we pass them to the next "Encoder" to process the new token id with the last h, c.
And after we finish all the input tokens, we will get the last hidden state and cell state, which will be passed to the decoder.
Importance of the Embedding Layer: The embedding layer(for the english vocab) is important because it will convert the input token id to a dense vector, which will be passed to the LSTM cell. This dense vector will have a semantic meaning, and it will be learned during the training process.
A Tricky Question: How big should the embedding space be?
Well the general sense answer should is, it should be big enough to capture the semantic meaning of the words, but not too big (like equal to the size of the vocab) because in this case it will lose its meaning(it will just be like a lookup table, not capturing anything about sematics and relations between words).
Setup:

- Embedding Layer: input_dim = len(english_vocab) = 203(in our case), output_dim = 256 (The input dimension is the length of the vocabulary because each unique token in the input vocabulary needs its own representation in the embedding layer.)
- LSTM Layer: Has itself 2 layers with 512 units(neurons) each.

2.2.2 Decoder

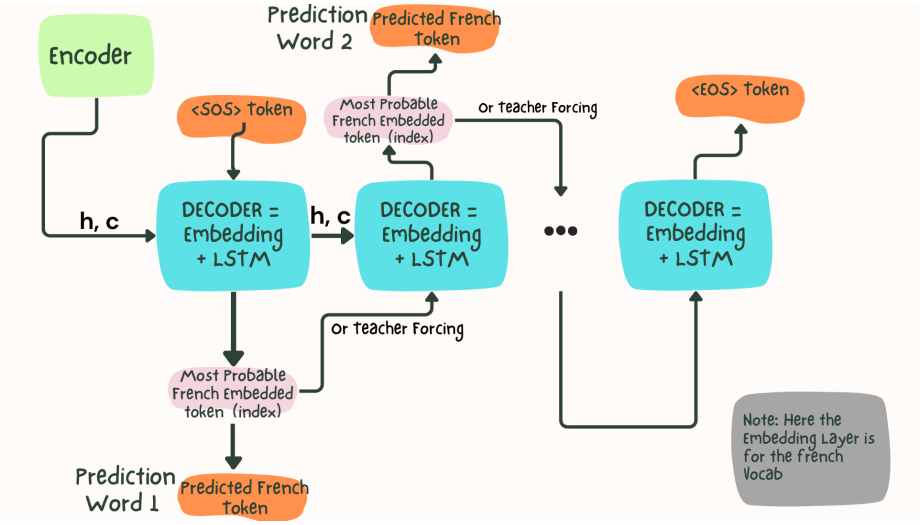

Figure 4: Decoder

The architecture of the decoder is summarize in Figure 4. The decoder is also a unidirectional LSTM, its goal is to generate the output sequence.
So similarly as before but here the decoder at each time step will take the last hidden state and cell state from the previous time step (initially they are the last hidden state and cell state from the encoder) and the output token id of the last time step (initially <SOS> token), and then it will pass them to the LSTM cell at this time step.
At each time step the LSTM cell will output the hidden state and cell state, and the output token id(which will the predicted word in french).
Notice how we output the most probable word (id) in the french vocab, but in the training process we output the probablity over the whole french vocab.
Importance of the Embedding Layer: Now the embedding layer in this case is for the french vocab, and it will convert the output token id to a dense vector, which will be passed to the LSTM cell. This dense vector will have a semantic meaning, and it will be learned during the training process(so in other word it will take the ouput from the last time step(which is in a form of an id)

and convert it to a dense vector, which will be passed to the LSTM cell).
Setup:

- Embedding Layer: input_dim = len(french_vocab) = 348, output_dim = 256 (The input dimension is the length of the vocabulary because each unique token in the output vocabulary needs its own representation in the embedding layer.)
- LSTM Layer: Has itself 2 layers with 512 units(neurons) each.
- Dense Layer: output_dim = len(french_vocab), activation = softmax (The dense layer is used to output the probability distribution over the french vocab, so the output will be a vector of size len(french_vocab) with values between 0 and 1, and the sum of them will be 1).
- Note: For inference we use the class with the highest probability, and for the loss computation we use the probablity distribution.

3 Training

The model training was conducted on an HPC cluster with the following resources:

- Compute Nodes: 1 node
- CPU Cores: 16 cores per node
- Memory: 16 GB per node

4 Importance of training and ways to prevent overfitting

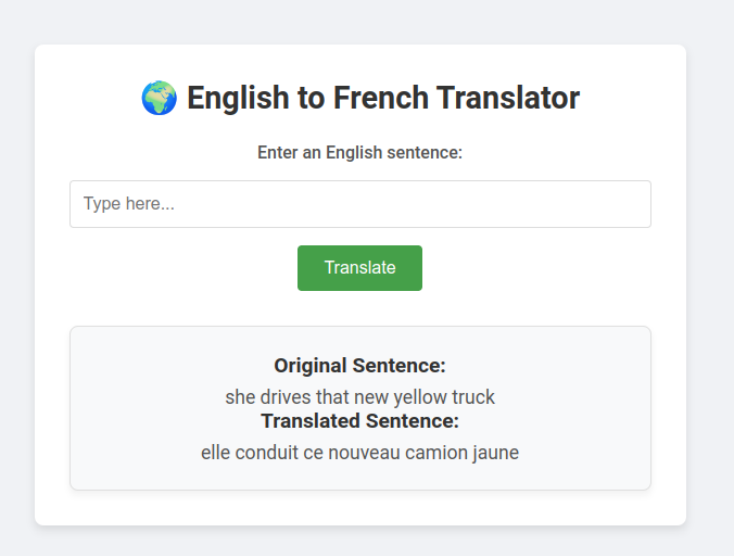

Figure 5: Correct Output

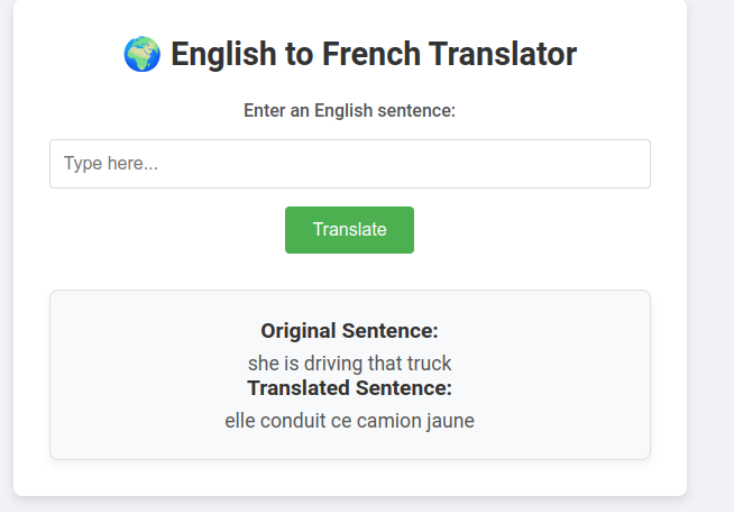

Figure 6: Overfitting and Association

From these two figures we can see how the model learned to associate the word truck always with color jaune, so even when the input is just truck the model will still output the "yellow truck" in french.
This could be solved by adding more data, of by using additional layers with dropout, or also by using the attention mechanism which we will explore next.

5 Importance of Attention Mechanism

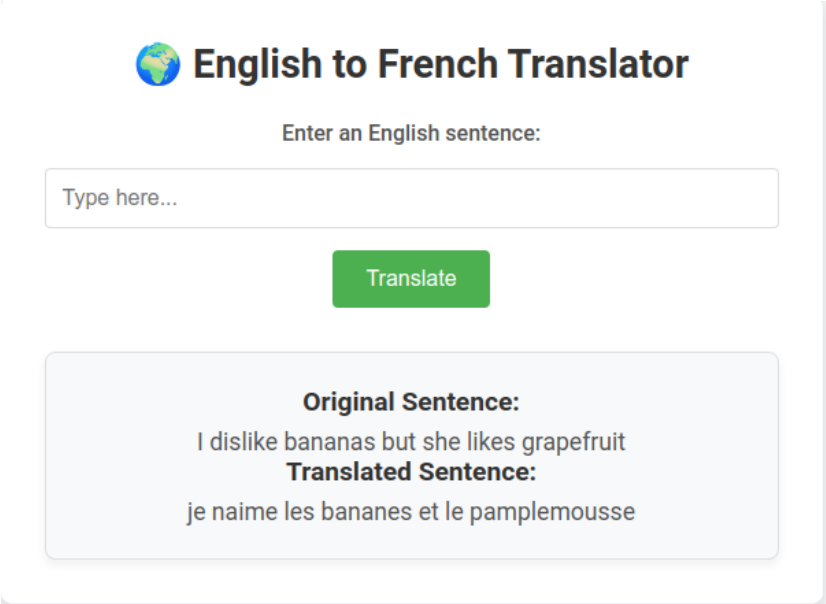

Figure 7: Attention Mechanism

this example clearly illustrates the importance of incorporating an attention mechanism in a Seq2Seq model. Without attention, the model processes the sentence sequentially, which can lead it to focus only on certain parts, especially the beginning or end, while losing context in the middle or when shifts in meaning occur (e.g., handling both "I dislike bananas" and "she likes grapefruit").
In this example, the model incorrectly translates "dislike" to "like" (in French, "je n'aime pas" would be used for "dislike" instead of "j'aime") and ignores the contrast between "I" and "she," leading to an incorrect translation.
The attention mechanism allows the model to dynamically focus on relevant parts of the input sequence as it generates each word in the output sequence. This enables the model to capture the context and meaning of the entire sentence, leading to more accurate translations.

The topic of attention will be explored in future work, as it is a crucial component for improving the model's performance.

6 Conclusion

In this study, we presented a model for translating English text to French using LSTM networks. We explored the model's architecture, training process, and results, highlighting the importance of attention mechanisms for improving translation accuracy. We also discussed the limitations of the model and potential areas for future work, such as incorporating attention mechanisms and expanding the dataset. This study provides insights for understanding sequence-based NLP tasks, particularly in the context of translation using recurrent neural networks.