Seq2Seq Translation

Veerendarnath NALADALA B00760534 | Priyanka PIPPIRI B00746159 | Shamir KAZI shahmirkazi93 | Prannoy BHUMANA b00770106

Abstract

The objective of this report is to implement a seq2seq model and test it on a translation task. This implementation is divided into three individual parts and worked upon as per the exercise.

1 Part-1

Introduction

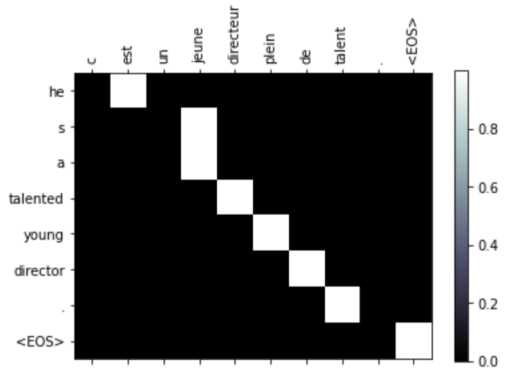
Seq2Seq (Sequence to Sequence) is a many to many network where two neural networks, one encoder and one decoder work together to transform one sequence to another. The core highlight of this method is having no restrictions on the length of the source and target sequence. At a high level, the way it works is: An encoder network condenses an input sequence into a vector, this vector is a smaller dimensional representation and is often referred to as the context/thought vector. This thought vector is served as an abstract representation for the entire input sequence. The decoder network takes in that thought vector and unfolds that vector into the output sequence.

1.1 *Analyze how attention is distributed along the tokens. Do you observe any special patterns for EOS/SOS tokens? What happens if you remove those tokens? Do you have any interpretation for such behavior?*

Analyzing attentions

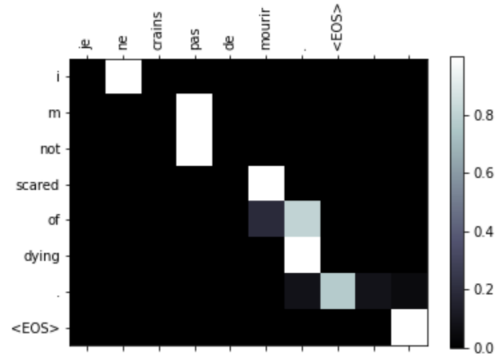
A useful property of the attention mechanism is its highly interpretable outputs. Because it is used to weight specific encoder outputs of the input sequence, we can imagine looking where the network is focused most at each time step. We passed on French sentence and it is translated to English through attention mechanism. The following is the visualization of attention distribution for some example sentences:

```
input = c est un jeune directeur plein de talent .
output = he s a talented young director . <EOS>
```

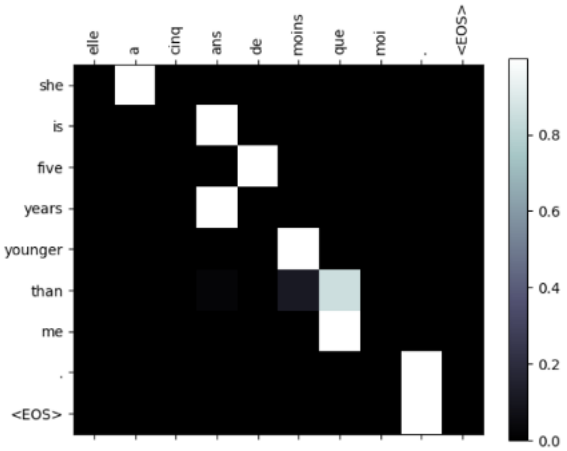


We can see that for each output word from the decoder, the weights assigned to the input words are different, except for once and the relationship between the inputs and outputs that the model is able to draw. This tells us how much of weight to a English word given by the French word. The EOS tokens should be matched with the predicted sentence. This is indirect way of seeing if the predicted sentence is correct or not. This only gives the highest weight for a word by the corresponding English word. By this the mechanism can chose only one word based on the highest probability(weights in this case). For the rest of the non-related words, the attention weight is 0. We can also observe that our model is not assigning attention weights to EOS/SOS tokens.

```
input = je ne crains pas de mourir .
output = i m not scared of dying . <EOS>
```



The EOS tags doesn't match here. This slightly shows that there is a slight error in the sentence predicted. The EOS tag got assigned weights, which clearly doesn't make much sense. But we can see that sentence predicted very close to the required output. Maybe we need to train on larger dataset for the mechanism to learn the context more accurately.

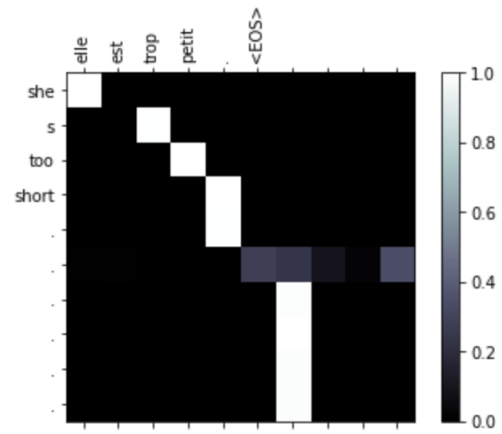


Removal of EOS/SOS tags

We append a start of sequence (<SOS>) and end of sequence (<EOS>) token to the start and end of sentence, respectively. The encoder reads the input sequence to construct an embedding representation of the sequence. Terminating the input in an EOS token signals to the encoder that when it receives that input, the output needs to be the finalized embedding. The EOS token is important for the decoder as well: the explicit "end" token allows the decoder to emit arbitrary-length sequences. The decoder will tell us when it's done emitting tokens: without an "end" token, we would have no idea when the decoder is done talking to us and continuing to emit tokens will produce gibberish. This is why the tokens are necessary for the translation.

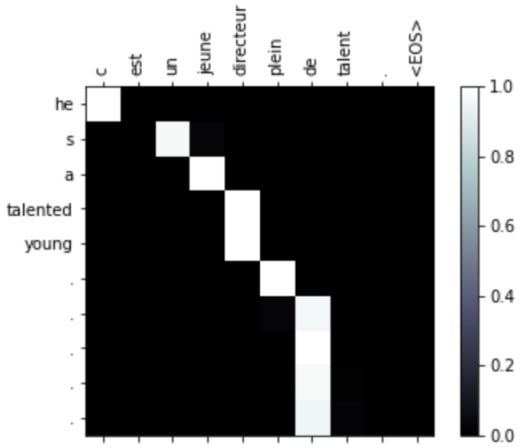
We performed tests by removing the tags using attention model. The results are displayed below.

```
input = elle est trop petit .
output = she s too short . . . . . .
```



Firstly, the SOS tokens in the figure is due to the tag we added in the plot function. This has nothing to do with the total model with no tags. We can onserve that model is assigning weights to words which were not even present in the sentence. This shows that the model doesn't know when to stop the decoding and encoding process. Although, it is able to predict sentences very close to the actual sentence, but this way it makes model less efficient.

```
input = c est un jeune directeur plein de talent .
output = he s a talented young . . . . .
```



1.2 *why is attention so important in this model? What would happen if it gets removed?*

Attention is simply a vector, often the outputs of dense layer using soft-max function. In simple seq2seq, translation relies on reading a complete sentence and compress all information into a fixed-length vector, a sentence with hundreds of words represented by several words will surely lead to information loss which leads to inadequate translation. However, attention partially fixes this problem. It allows machine translator to look over all the information the original sentence holds, then generate the proper word according to current word it works on and the context. It can even allow translator to zoom in or out (focus on local or global features).

Similar to the basic encoder-decoder architecture, this mechanism plugs a context vector into the gap between encoder and decoder. Here context vector takes all cells' outputs as input to compute the probability distribution of source language words for each single word decoder wants to generate. By utilizing this mechanism, it is possible for decoder to capture somewhat global information rather than solely to infer based on one hidden state. This way this model gives better accuracy and more efficient.

Results without Attention

We carried out tests on random sentences with the labels included. The results were skewed. The model is able to perform well in some occasions and not so much in other. Here we can see that the predicted sentence is not very close as the actual sentence. The model is able to capture the words, but it is not able to put them in a proper context.

```
> je suis sideree .
= i m stunned .
< i m stunned . <EOS>

> nous pigeons .
= we re getting it .
< we re getting it . <EOS>

> il est recherche par la police .
= he is hunted by the police .
< he is interested in the police . <EOS>

> je suis convaincu de son innocence .
= i am convinced that he is innocent .
< i am convinced of his innocence . <EOS>

> je vais y repenser encore une fois .
= i m going to reconsider it .
< i m going to it it . <EOS>
```

Results without Attention

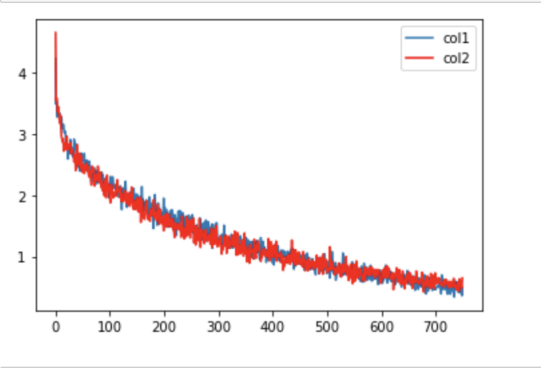
We tested the attention model on 10 random sentences and the results were intriguing. All the sentences were predicted were very close to actual sentence. The results can be seen in the image below. The model was able to predict almost every word correctly and was able to capture the context used in the sentences. As we know, context vector is the main difference between simple and attention decoders. From the results we can say that it is achieved, and the model performed absolutely well.

```
> il est occupe a faire quelque chose .
= he is busy doing something .
< he is busy doing something . <EOS>

> j y vais .
= i m going .
< i m going . <EOS>

> il est plutot optimiste .
= he is rather optimistic .
< he is rather optimistic . <EOS>

> vous etes mon ennemi .
= you re my enemy .
< you re my enemy . <EOS>

> tu es riche n est ce pas ?
= you re wealthy aren t you ?
< you re german aren t you ? <EOS>

> elles ont des ennuis .
= they re in trouble .
< they re in trouble . <EOS>
```

In addition, We selected 4 sentences to analyze the performance and difference between the 2 models. The difference in results can be seen and model with attention mechanism predicted better results. We

	Input	Simple Seq2Seq	Seq2Seq+Attention
1	elle a cinq ans de moins que moi	she s five years younger than me .	she s two years older than me .
2	elle est trop petit	she is too trusting .	she s too short .
3	je ne crains pas de mourir	i m not scared to die .	i m not scared of dying .
4	c est un jeune directeur plein de talent	he is a talented young to .	he s a talented young director .

observed that trend in losses didn't change much. In fact, the model with attention ended up with slightly more loss. But the predictions showed us the better model.



1.3 *What is teacher forcing? why would you want to use teacher forcing?*

Teacher forcing(TFor.) is the concept of using the real target outputs as each next input, instead of using the decoder's guess as the next input. Models that have recurrent connections from their outputs leading back into the model may be trained with teacher forcing. Training with Teacher Forcing converges faster. At the early stages of training, the predictions of the model are very bad. If we do not use Teacher Forcing, the hidden states of the model will be updated by a sequence of wrong predictions, errors will accumulate, and it is difficult for the model to learn from that.

	Input	Without Tfor.	With TFor.
1	elle a cinq ans de moins que moi	she is five years younger than i am .	she s two years older than me .
2	elle est trop petit	she s too short .	she s too short .
3	je ne crains pas de mourir	i m not dying of dying .	i m not scared of dying .
4	c est un jeune directeur plein de talent	he s a talented young young .	he s a talented young director .

The model with teacher forcing preformed better. In the 3rd and 4th example, the model without TFor. was not able to predict correctly. This shows that with TFor., the grammar and context are correct. We also observed another behavior here. In the first sentence, the model without TFor. predicted correctly rather than model with TFor. The correct translation for 'cinq' should be 'five'. But, with TFor, predicted as 'two'. Finally we came to a conclusion that although TFor. can make 2 or 3 mistakes, we decided to use it as it generates more context and predicts better.

2 Part-2

Beam search is an improved algorithm based on greedy search. It has a hyper-parameter named beam size, k . At timestep 1, we select k words with the highest conditional probability to be the first word of the k candidate output sequences. For each subsequent timestep, we are going to select the k output sequences with the highest conditional probability from the total of $k*y$, where y is size of output text dictionary size, possible output sequences based on the k candidate output sequences from the previous timestep. These will be the candidate output sequences for that timestep. Finally, we will filter out the sequences containing EOS tag from the candidate output sequences of each timestep and discard all the sub-sequences after it to obtain a set of final candidate output sequences.

Lenght Penalty is a small change to the beam search algorithm that can help get much better results.

2.1 *How do different beam sizes affect the result?*

We observed different behaviors of model. This is due to various hyper parameters like vocab size, Byte Pair encoding are involved. But we tried on changing maximum length of sentences which is related to vocab size. By increasing the size of beam, the BLEU scores started to increase as we increased the vocab size. However it didn't change much when the beam widths are increased until 50. Then we tried by increasing the beam width to 60. This led to increasing BLEU scores. The results are displayed below.

	Max Length	Beam Width	BLEU scores	Large Beam(60) BLEU
1	10	3; 10; 30	72.5; 72.5; 72.5	NaN
2	30	3; 10; 30	84.42; 84.42; 84.42	84.42
3	50	3; 10; 30	86.96; 86.96; 86.96	86.99

2.1.1 max_length=10

When the input dataset is processed to get words and trimmed sentences, we obtained 48 words. This will be shape of input tensor and the data to train on. The data is so small here. So, the model was not able to learn more. This can also lead to overfitting if trained for more iterations. But we kept the iterations value as same through out the project. We ended up getting BLEU score of 72.5. It didn't change if the beam size is changed until a size of 30.

2.1.2 max_length=30

As the length is more, the length of the tensors also will increase. We had a length of 265 as our input tensor. The model executed without any error. We observed a BLEU score of 84.42. The remained same as the other beam widths.

2.1.3 max_length=50

The width is then increased to 50. The data we were able to train has increased broadly. We were looking at a number 3389. Due to the huge data and more training, the BLEU scores increased to 86.96 accordingly.

We tried 2 types of length penalties. In the first approach, we kept penalty to zero. In the second approach, we tested when beam width is 50. Surprisingly, BLEU score value didn't change even after applying a penalty factor as 10. It remained same. This might be due to wrong implementation.

2.2 *What happens with a very large beam size (> 50)?*

2.2.1 max_length=10

We observed a different pattern of behavior here. It changed from one dataset to another and also by maximum length of the sentences. As the length is 10, when we read the data from the dataset, the available words for training were 48. This would be the length of the input tensor. This goes into encoder and decoder models. After this the output shape of decoder will also be the same. Then when this tensor is given as input to 'topk' function, which Returns the k largest elements of the given input tensor along a given dimension, shows an error because the maximum possible is 48 and our dimension which is beam width is 60. As the dimension is more than the tensor size, this action can't be performed.

2.2.2 max_length=30

This didn't change with change in beam size to 60.

2.2.3 max_length=50

The model gave us better result here extending the score to 86.99

3 Part-3

The trained data for above part 1 and part2 are different. We tried to see how the model performed on different datasets. So, we trained the model based on part-1 dataset(<TRAIN>) and predicted on part-2 dataset(<TEST>).

3.1 Impact on the vocabulary choice

As we need huge dataset, we kept the max_length to 50. This resulted in more words to train on. After this, we executed the model with differnet beam sizes and predicted on <TEST> dataset. The BLEU scores are also calculated to analyze the performance.

	Beam Width	BLEU scores
1	3	43.002932151162845
2	10	43.32002272996931
3	30	43.38203836374787
4	60	43.82248447094242

The scores kept on increasing as we increased the size of the beam. This shows the significance of the beam width. The model was able to capture the context. But it didn't perform very well in instances where the sentences were very large. Overall, the model performance is very good which resulted in good BLEU scores.