Rutvik Damjibhai Roy – 400490159   SEP 775 Computational Natural Language Processing

Assignment - 2

Q 1. Analyze the generated text. Discuss how well your model captures the style and coherence of the chosen dataset. Reflect on the performance of the basic RNN model versus the LSTM-enhanced version. Consider the effects of different hyperparameters on the quality of the generated text.

Ans:

Text generated by basic RNN (100 words):

```
twinkle  twinkle  little star  star  i i wonder what you are  up above the
blazing sun is gone  when he nothing shines upon  then you show your shines
the world so high  like a diamond in the sky  when the blazing sun is gone
when he nothing shines upon  then you show your shines the world so high
like a diamond in the sky  when the blazing sun is gone  when he nothing
shines upon  then you show your shines the world so high  like a diamond in
the sky  when the blazing sun is gone  when he nothing shines upon
```

Text generated by LSTM (enhanced RNN - 100 words):

```
twinkle  twinkle  little star  star  star  sky  sky  sky  sky  sky  sky
sky  sky  sky  sky  sky  sky  sky  sky  sky  sky  sky  sky  sky  sky  sky
sky  sky  sky  sky  sky  sky  sky  sky  sky  sky  sky  sky  sky  the the
the the the the the the the the the the the the the the the the the the the
the the the the the the the the the the the the the the the the the the the
the the the the the the the the the the the the the the the the the the the
the the
```

- I've provided the final text generated by both basic RNN and LSTM after adjusting certain parameters.

- At first, I used small sizes for embedding and hidden layers (embedding_size=10 and hidden_size=10), which couldn't handle longer connections well. When predicting the fourth word "star", it lost track of the context. It only got the first four words right, then kept predicting "sky" repeatedly for the next 96 words.

- To make sure the neural network retained accurate information from each time step to the end, I increased the hidden_size to 150 and embedding_size to 100 to capture better dependencies.

- I also set the learning rate to 0.001 to prevent the gradient descent from making large updates to the neural network parameters. After training the network for another 1000 epochs, I saw a significant decrease in training loss for the basic RNN, as shown in fig 1. It correctly predicted 5 to 6 sentences out of 100 words and captured good dependencies. It only repeated two words in the first two sentences.

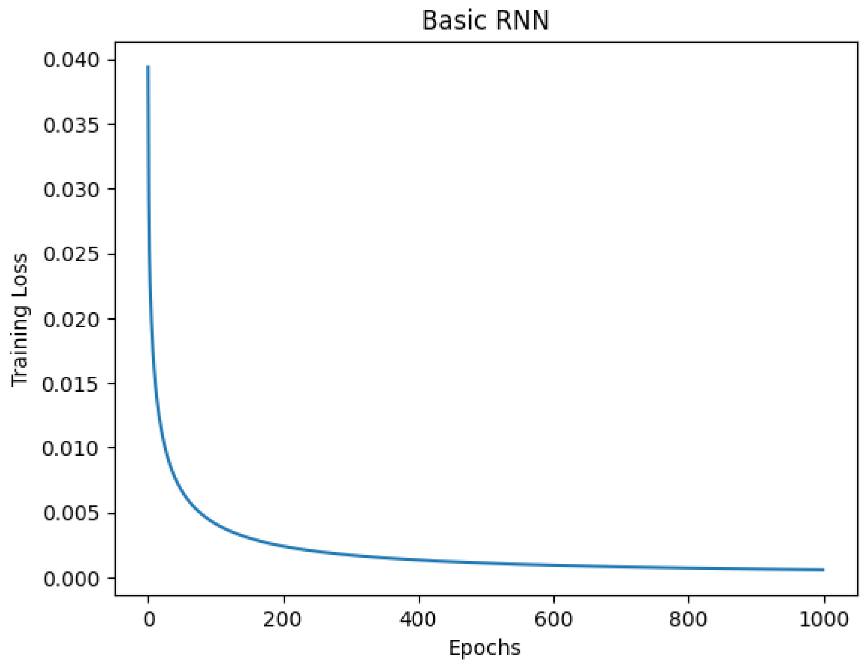


Fig 1 - Training Loss of Basic RNN

- I tried increasing the number of layers to 5 to see how it affected the outcome. However, I only noticed a slight improvement over the previous result. The outcome has been shown earlier.

- For LSTM, I used the same parameters as before, and the training loss decreased. However, after predicting 4 words, it started predicting similar words. I attempted to adjust some parameters, but the training loss followed the same pattern, as seen in fig 2. It appears that my model became overfitted according to the figure.

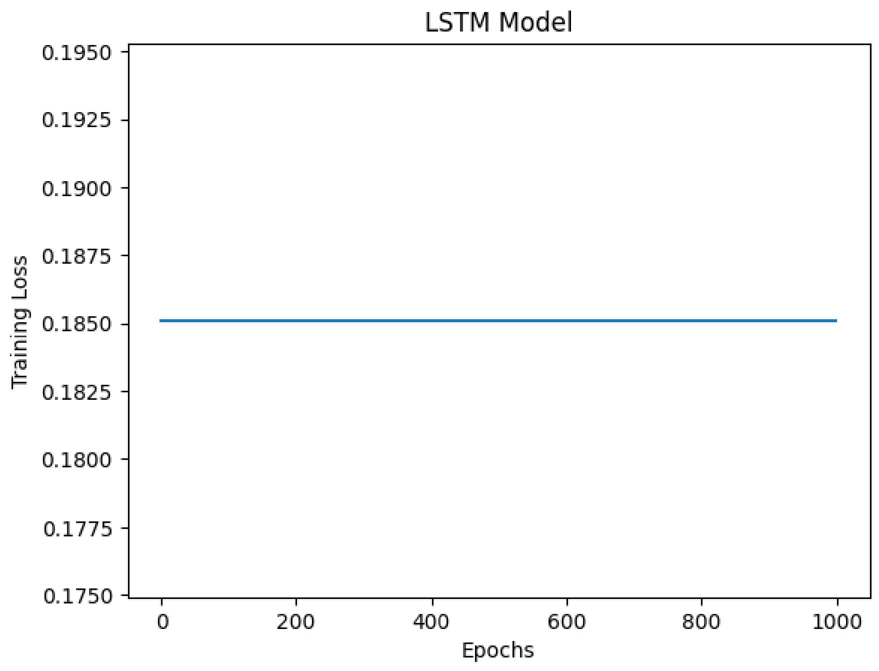


Fig 2 - Training Loss of LSTM

- Still, the architecture of LSTM model needs to be improved like adding dropout layer to prevent feature co-adoption and adding generalization term to prevent the overfitting.

Q 2. Discuss the impact of the attention mechanism on the translation quality. Compare the performance of your model with and without attention using both qualitative examples and quantitative metrics. Reflect on how different architectural choices and hyper-parameters affected the outcomes.

Ans:

Here, I built an encoder-decoder model (seq2seq model) using LSTM layers for translating English to Hindi. Using the PyTorch library, I created separate classes for encoder and decoder, implementing both with and without attention mechanisms. Although the original dataset comprised 23,000 sentences, I worked with only the first 500 sentences (pairs) to grasp the implementation nuances of encoder-decoder models for language translation. This decision also aimed to simplify the model due to limited computational resources.

I) Seq2Seq Model without Attention:

Initially, I employed a single layer of LSTM in both encoder and decoder, with an embedding size and hidden size of 90 and 256, respectively. Training for 250 epochs revealed a decrease in training loss. However, during prediction, the model struggled with even simple words to predict its meaning in Hindi as well as with words having multiple meanings in Hindi, like "Jump," which has three meanings: "**उछलो**", "**कूदो**", and "**छलांग**". The model consistently predicted only one meaning and failed to maintain word order in the translated sentences. To address these issues, I increased the number of LSTM layers and reduced the hidden size to 128. After another 250 epochs of training, the loss decreased significantly but still I experiences bottleneck issue, as depicted in Fig 3. The test results are outlined below.

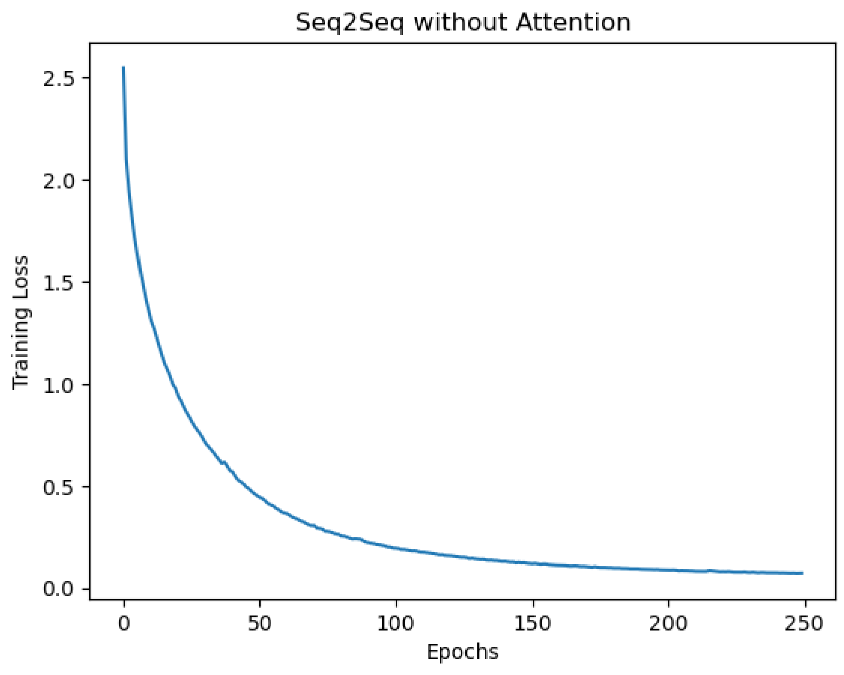


Fig 3 - Training Loss of Seq2Seq Model without Attention

Test Results (Qualitative examples):

```
Encoder Input: ['help', '<pad>', '<pad>', '<pad>', '<pad>']
Decoder output: बचाओ <pad> <pad> <pad> <pad> <pad> <pad> <pad>


Encoder Input: ['jump', '<pad>', '<pad>', '<pad>', '<pad>']
Decoder output: <pad> <pad> <pad> <pad> <pad> <pad> <pad> <EOS>


Encoder Input: ['jump', '<pad>', '<pad>', '<pad>', '<pad>']
Decoder output: <pad> <pad> <pad> <pad> <pad> <pad> <pad> <EOS>


Encoder Input: ['jump', '<pad>', '<pad>', '<pad>', '<pad>']
Decoder output: <pad> <pad> <pad> <pad> <pad> <pad> <pad> <EOS>


Encoder Input: ['hello', '<pad>', '<pad>', '<pad>', '<pad>']
Decoder output: <pad> <pad> <pad> <pad> <pad> <pad> <pad> <EOS>



Encoder Input: ['hello', '<pad>', '<pad>', '<pad>', '<pad>']
Decoder output: <pad> <pad> <pad> <pad> <pad> <pad> <pad> <EOS>


Encoder Input: ['cheers', '<pad>', '<pad>', '<pad>', '<pad>']
Decoder output: <pad> <pad> <pad> <pad> <pad> <pad> <pad> <EOS>


Encoder Input: ['cheers', '<pad>', '<pad>', '<pad>', '<pad>']
Decoder output: <pad> <pad> <pad> <pad> <pad> <pad> <pad> <EOS>


Encoder Input: ['got', 'it', '<pad>', '<pad>', '<pad>']
Decoder output: समझे कि <pad> <pad> <pad> <pad> <pad> <EOS>


Encoder Input: ['im', 'ok', '<pad>', '<pad>', '<pad>']
Decoder output: मैं ठीक हूँ। <pad> <pad> <pad> <pad> <pad>
```

Evaluation of seq2seq model without attention (Quantitative metrics):

```
1. Blue Score of Seq2Seq Model without attention: 0.8270509611962559
2. METEOR Score of Seq2Seq Model without attention: 0.0
```

II) Seq2Seq Model with Attention:

For the seq2seq model with attention, I maintained consistency in parameters, employing two layers of LSTM within both the encoder and decoder classes. The embedding_size and hidden_size were set to 90 and 128, respectively. In the decoder class definition, I incorporated an attention mechanism to decrease the bottleneck issue. Training the model for 250 epochs showed a downward trend in training loss, as illustrated in Fig 4, aligning with the loss trend observed in the model without attention.

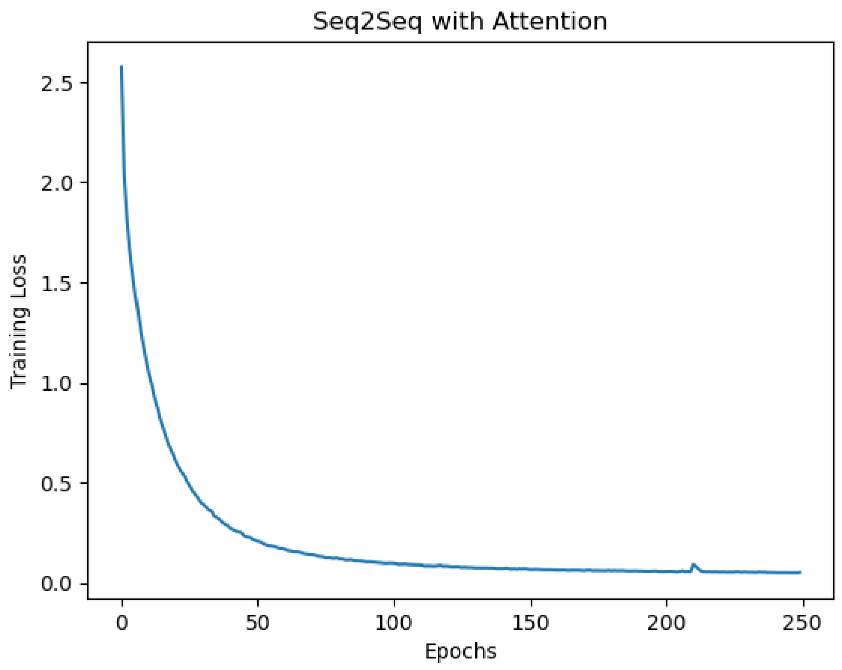


Fig 4 Training Loss of Seq2Seq Model with Attention

Test Results (Qualitative examples):

```
Encoder Input: ['help', '<pad>']
Decoder output: बचाओ <pad> <pad> <pad> <pad>


Encoder Input: ['jump', '<pad>']
Decoder output: छलांग <pad> <pad> <pad> <pad>


Encoder Input: ['jump', '<pad>']
Decoder output: छलांग <pad> <pad> <pad> <pad>


Encoder Input: ['jump', '<pad>']
Decoder output: छलांग <pad> <pad> <pad> <pad>


Encoder Input: ['hello', '<pad>']
Decoder output: नमस्कार। <pad> <pad> <pad> <pad>


Encoder Input: ['hello', '<pad>']
Decoder output: नमस्कार। <pad> <pad> <pad> <pad>
```

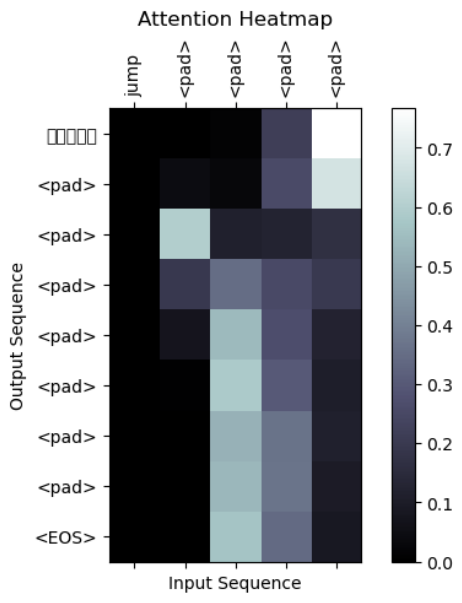
```
Encoder Input: ['cheers', '<pad>']
Decoder output: चियर्स <pad> <pad> <pad> <pad>


Encoder Input: ['cheers', '<pad>']
Decoder output: चियर्स <pad> <pad> <pad> <pad>


Encoder Input: ['got', 'it']
Decoder output: समझे कि नहीं <pad> <pad>


Encoder Input: ['im', 'ok']
Decoder output: मैं ठीक हूँ। <pad> <pad>
```

<u>Evaluation of seq2seq model with attention (Quantitative metrics):</u>

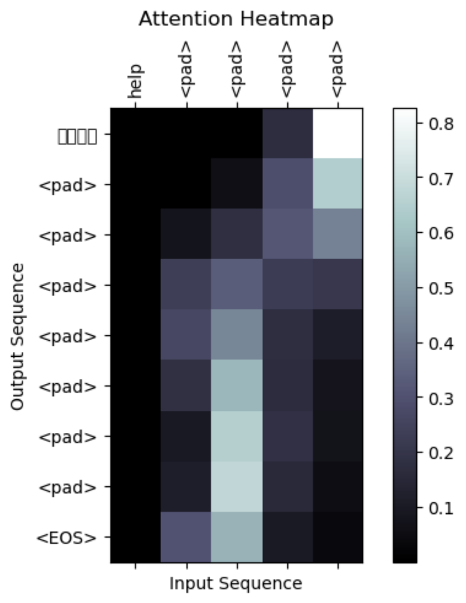
```
   1. Blue Score of Seq2Seq Model with attention: 0.9364800453186035
   2. METEOR Score of Seq2Seq Model with attention: 0.2631578947368421
```

Attention Heatmap of seq2seq model with attention:

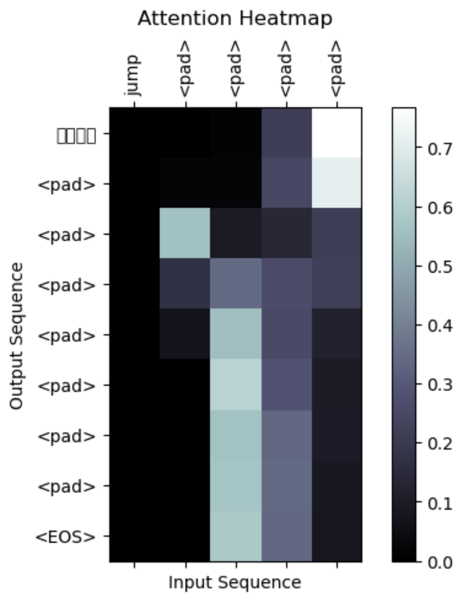
छलांग



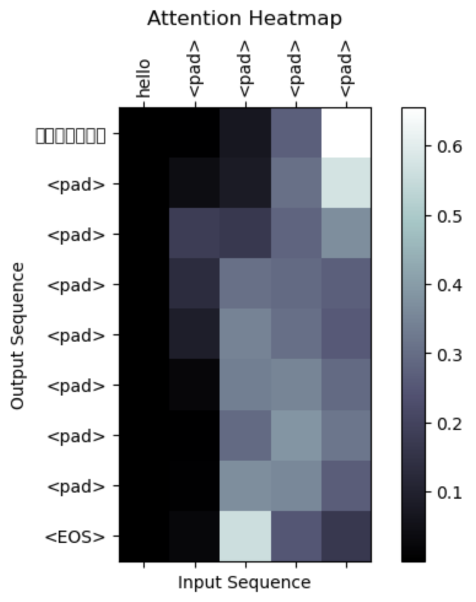
बचाओ



कूदो



नमस्कार।



(Note: Matplotlib does not support Hindi text. So, for understanding, I have written Hindi text for respective English text on lop left corner of each image of heatmaps. Those should be on the Y limb of the heatmaps in actual images, but it shows some symbols for that Hindi text)

Conclusion:

Seq2Seq Model without Attention:

- The model initially struggled with capturing meanings and preserving word order, especially for words with multiple meanings.
- After adjustments, such as increasing LSTM layers and reducing hidden size, there was a significant improvement in training loss.
- However, the model's performance on test data remained suboptimal, with a relatively lower BLEU score of 0.827 and a METEOR score of 0.0.
- Translated only 3 out of 10 sentences correctly during the prediction task due to bottleneck problem.

Seq2Seq Model with Attention:

- Integrating attention mechanism into the model significantly improved translation accuracy and quality.
- The attention mechanism helped to address the bottleneck problem and allowed the model to focus on relevant parts of the input sequence during decoding.
- Test results showed improved translation quality with better handling of words and preservation of word order.
- The model achieved a higher BLEU score of 0.936 and a METEOR score of 0.263, indicating superior performance compared to the model without attention.
- Attention heatmap visualization shows the model's ability to align input and output sequences effectively, leading to more accurate translations.

In conclusion, the Seq2Seq model with attention outperforms the model without attention in terms of translation quality and accuracy. The attention mechanism proves to be crucial in capturing context and improving the model's ability to handle complex language structures, resulting in more fluent and contextually relevant translations.