CS6910 - Assignment 3

Use recurrent neural networks and transformers to build a transliteration system.

Gowthamaan

Submitted by: Gowthamaan, ED23S037

Instructions

Problem Statement

Question 1 (15 Marks)

Build a RNN based seq2seq model which contains the following layers: (i) input layer for character embeddings (ii) one encoder RNN which sequentially encodes the input character sequence (Latin) (iii) one decoder RNN which takes the last state of the encoder as input and produces one output character at a time (Devanagari).

The code should be flexible such that the dimension of the input character embeddings, the hidden states of the encoders and decoders, the cell (RNN, LSTM, GRU), and the number of layers in the encoder and decoder can be changed.

(a) What is the total number of computations done by your network? (assume that the input embedding size is m, the encoder and decoder have 1 layer each, the hidden cell state is k for both the encoder and decoder, and the length of the input and output sequence is the same, i.e., T, the size of the vocabulary is the same for the source and target language, i.e., V)

Solution

- RNN : Total Computations = Computations by encoder + Computations by decoder

$$(\mathrm{km} + \mathrm{k2})\mathrm{T} + (\mathrm{km} + \mathrm{k2} + \mathrm{km})\mathrm{T}$$

- LSTM : Total Computations = Computations by encoder + Computations by decoder

$$(4(\mathrm{km} + \mathrm{k}^2) + 3\mathrm{k})\mathrm{T} + (4(\mathrm{km} + \mathrm{k}^2) + 3\mathrm{k} + \mathrm{mk})\mathrm{T}$$

- GRU: Total Computations = Computations by encoder + Computations by decoder

$$(3(\mathrm{k}^2 + \mathrm{km}) + 3\mathrm{k})\mathrm{T} + (3(\mathrm{k}^2 + \mathrm{km}) + 3\mathrm{k} + \mathrm{mk})\mathrm{T}$$

- Computations for Embedding layer:

$$2\mathrm{mV}$$

(b) What is the total number of parameters in your network? (assume that the input embedding size is m, the encoder and decoder have 1 layer each, the hidden cell state is k for both the encoder and decoder, and the length of the input and output sequence is the same, i.e., T, the size of the vocabulary is the same for the source and target language, i.e., V)

Solution

- RNN : Total parameters = Parameters for encoder + Parameters for decoder

(km+k^2+k)+(2km+k^2+k+m)

- LSTM : Total parameters = Parameters for encoder + Parameters for decoder

4(k^2+km+k))+(4(k^2+km+k)+km+m)

- GRU : Total parameters = Parameters for encoder + Parameters for decoder

  (3(k^2+km+k))+((3(k^2+km+k))+km+m)

- Parameters for Embedding Layer:

  2mV

Question 2 (10 Marks)

You will now train your model using any one language from the Aksharantar dataset (I would suggest picking a language that you can read so that it is easy to analyze the errors). Use the standard `train`, `dev`, `test` set from the folder `aksharantar_sampled/hin` (replace `hin` with the language of your choice)

BTW, you should read up on how NLG models operate in inference mode, and how it is different from training. This blog might help you with it.

Using the sweep feature in wandb find the best hyperparameter configuration. Here are some suggestions, but you are free to decide which hyperparameters you want to explore

- input embedding size: 16, 32, 64, 256, ...
- number of encoder layers: 1, 2, 3
- number of decoder layers: 1, 2, 3
- hidden layer size: 16, 32, 64, 256, ...
- cell type: RNN, GRU, LSTM
- bidirectional: Yes, No
- dropout: 0.2, 0.3 (btw, where will you add dropout? You should read up a bit on this)
- beam search in decoder with different beam sizes:

Based on your sweep, please paste the following plots, which are automatically generated by wandb:

- accuracy v/s created plot (I would like to see the number of experiments you ran to get the best configuration).
- parallel co-ordinates plot
- correlation summary table (to see the correlation of each hyperparameter with the loss/accuracy)

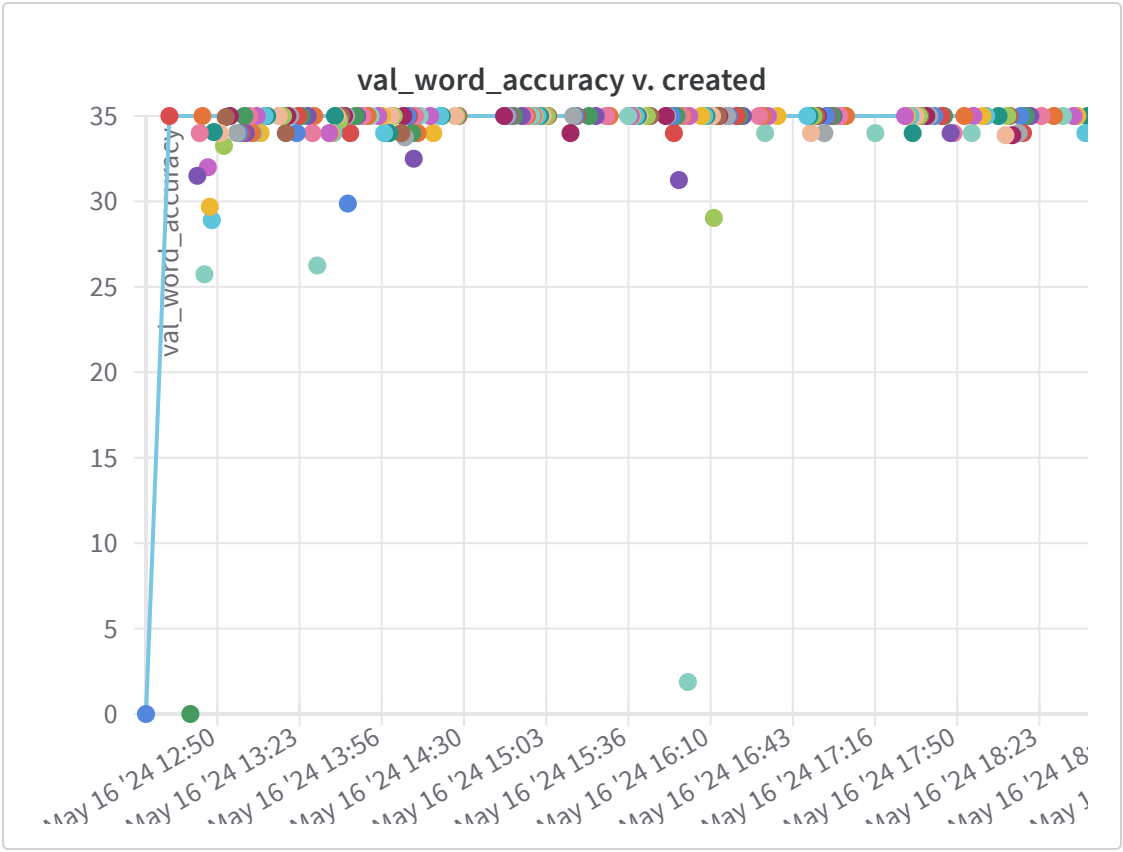
Also, write down the hyperparameters and their values that you swept over. Smart strategies to reduce the number of runs while still achieving a high accuracy would be appreciated. Write down any unique strategy that you tried for efficiently searching the hyperparameters.

Solution

I have run 300 runs in total using Bayesian search option in the wandb sweep. In this method, the next set of parameters for the network is selected based on the previously evaluated configurations.

```
sweep_config = {
        'method': 'bayes',
        'name': 'Q2_SWEEP_1',
        'metric': {
            'name': "val_word_accuracy",
            'goal': 'maximize',
        },
        'parameters': {
            'embedding_size': {'values': [16, 32, 64, 128, 256]}
            'num_layers': {'values': [1,2,3, 4]},
            'hidden_size': {'values': [16, 32, 64, 128, 256, 512
            'cell': {'values': ['lstm', 'gru', 'rnn']},
            'bidirectional': {'values': [False, True]},
            'beam_width': {'values': [1,2,3,4]},
            'dropout': {'values': [0.2, 0.3, 0.4, 0.5]},
            'learning_rate': {'values': [1e-2, 1e-3, 1e-4, 1e-5]
            'batch_size': {'values': [16, 32, 64, 128, 256, 512]
            "epochs": {'values': [5, 10, 15, 20]},
            'lang': {'value': 'tam'},
            'use_attention': {'value': False},
```

```
        },
    }
```



Parameter importance with respect to val_word_acc…

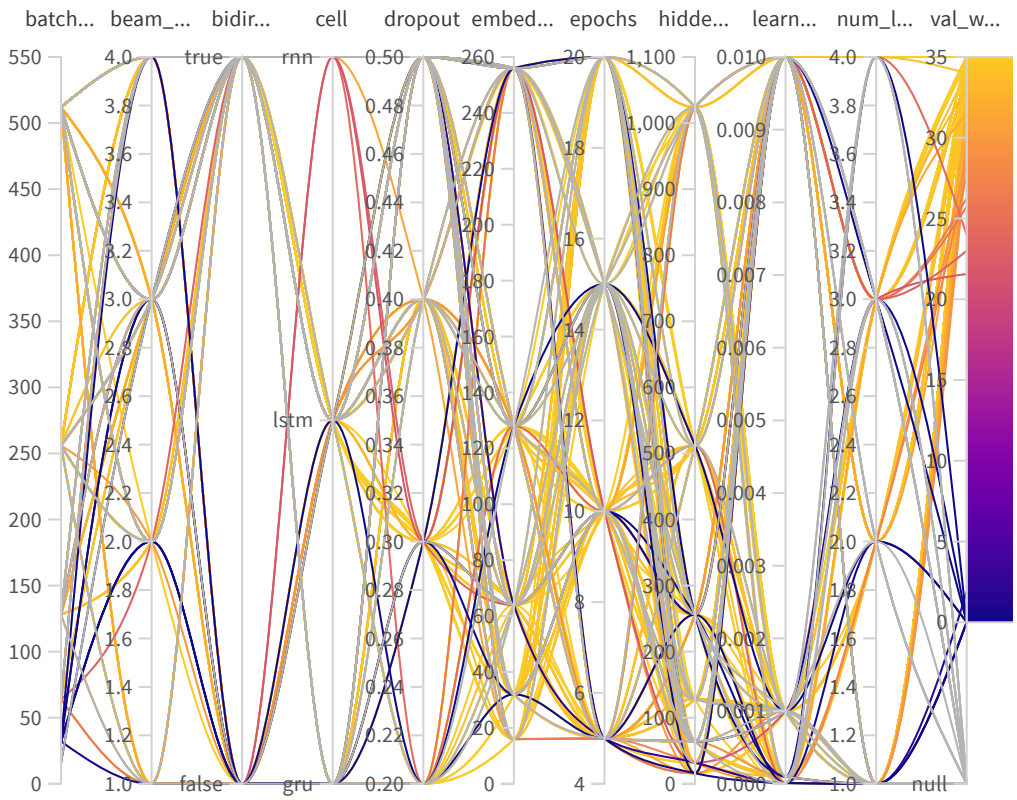
Search | Parameters | 1-10 of 15

Config parameter	Importance	Correlation
dropout		
cell.value_gru		
learning_rate		
batch_size		

Runtime

beam_width

cell.value rnn



Question 3 (15 Marks)

Use the best model from your sweep and report the accuracy on the test set (the output is correct only if it exactly matches the reference output). Based on the above plots write down some insightful observations. For example,

- RNN based model takes longer time to converge than GRU or LSTM

- using smaller sizes for the hidden layer does not give good results
- dropout leads to better performance

(Note: I don't know if any of the above statements are true. I just wrote some random comments that came to my mind)

Of course, each inference should be backed by appropriate evidence.

Solution

The best accuracy, I got is **35%**. These predictions are save in the file: https://github.com/Gowthamaan-P/cs6910_assignment3/blob/main/predictions_seq/predictions_seq.csv

- **Low learning rates** yielded better accuracy.Most models with high val_word_accuracy have learning rate of 0.0001(evident from the parallel co-ordinate plot). When the learning rate is set too high, it can lead to the phenomenon known as "overshooting" or "diverging" during the training process. The model's parameters are being updated in large steps, causing the optimization process to become unstable and unable to converge to a good solution.
- **GRU** performed better than LSTM and RNN. With a importance of 0.147 and positive correlation factor of 0.393, models with GRU cell type performed better than others.
- **Dropout** leads to better performance. High positive co-relation and importance (0.179) is seen in the correlation summary table. Dropout is a useful regularization technique that can improve the generalization ability of seq2seq models and help prevent overfitting. By introducing noise and encouraging independence among neurons, dropout promotes more robust learning and can lead to better performance on unseen data.
- **Bidirectional** doesn't really affect the validation accuracy. Though it has a high positive correlation (0.231) in the correlation summary, the importance (0.008) is very low.

Most of high accuracy models have bidirectional hyperparameters as either Yes or No (parallel plot).

- **High Batch Size** performed better. High batch size during training improved the performance. With high positive correlation of 0.203 and importance of 0.117, models trained with batch size above 128 have performed well (Parallel plot).
- **Moderate Embedding Size** provides more accuracy. Very high and very low embedding size leads to poor accuracy. There is a negative co-relation(-0.056) between accuracy and embedding size, seen in the co-relation summary table. With importance of 0.033, it is evident that embedding size, doesn't influence validation accuracy much.
- **High hidden size** yielded better accuracy. All the models with hidden size >= 512 yielded better validation accuracy. The hidden size determines the number of parameters and the representational capacity of the model. By increasing the hidden size, we allow the model to learn more complex and nuanced patterns in the data. It seems to be the case for Hindi dataset that this increased capacity is beneficial for capturing fine-grained details and complex structure between the characters
- **Low Beam width** provides better accuracy. Beam width of 2 or 3 have yielded better. Very low or higher beam width didn't perform well. This is indicated with a negative correlation in the table.
- **Less layers** in decoder and encoder improved validation accuracy for the models. We can see the negative correlation between the validation accuracy and number of layers in encoder and decoder from the correlation summary table. Adding more layers increases the number of parameters in the model and increases complexity. This leads to overfitting

(UNGRADED, OPTIONAL) Question 4 (0 Marks)

Question 5 (20 Marks)

Add an attention network to your base sequence-to-sequence model and train the model again. For the sake of simplicity, you can use a single-layered encoder and a single-layered decoder (if you want, you can use multiple layers also). Please answer the following questions:

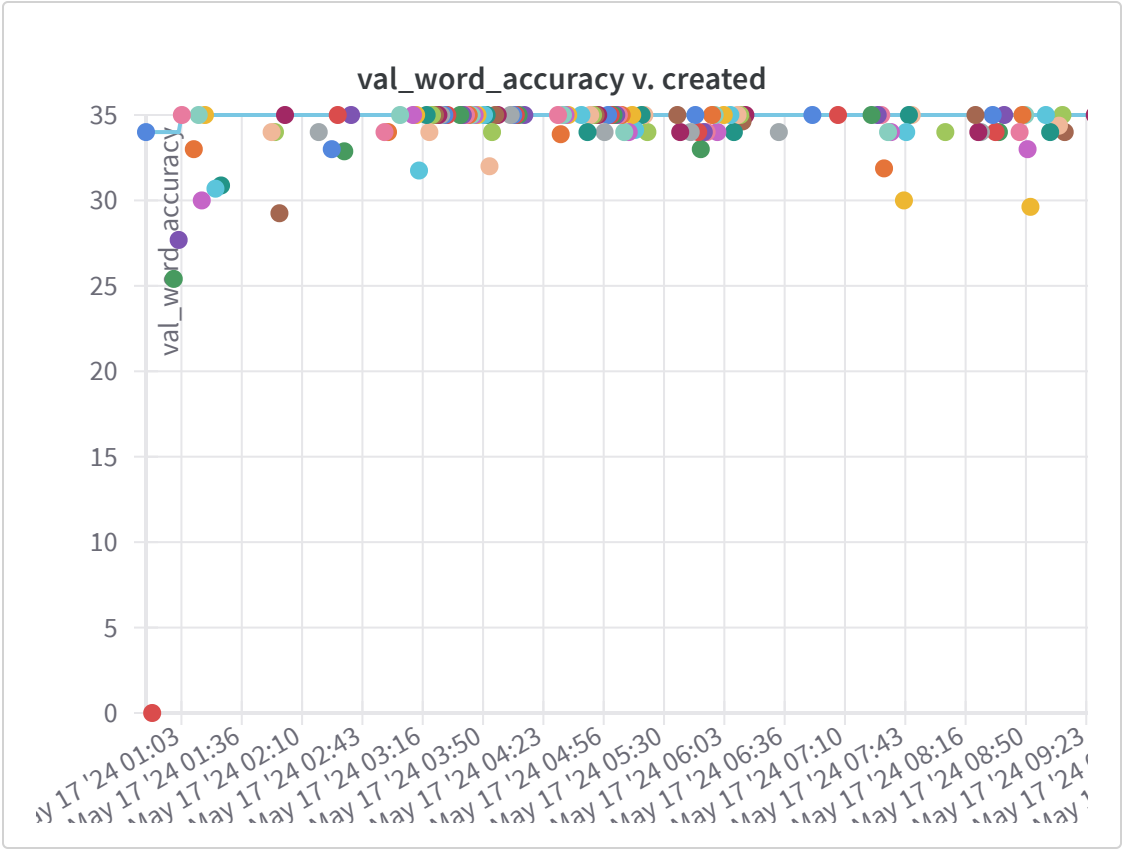
(a) Did you tune the hyperparameters again? If yes, please paste the appropriate plots below.

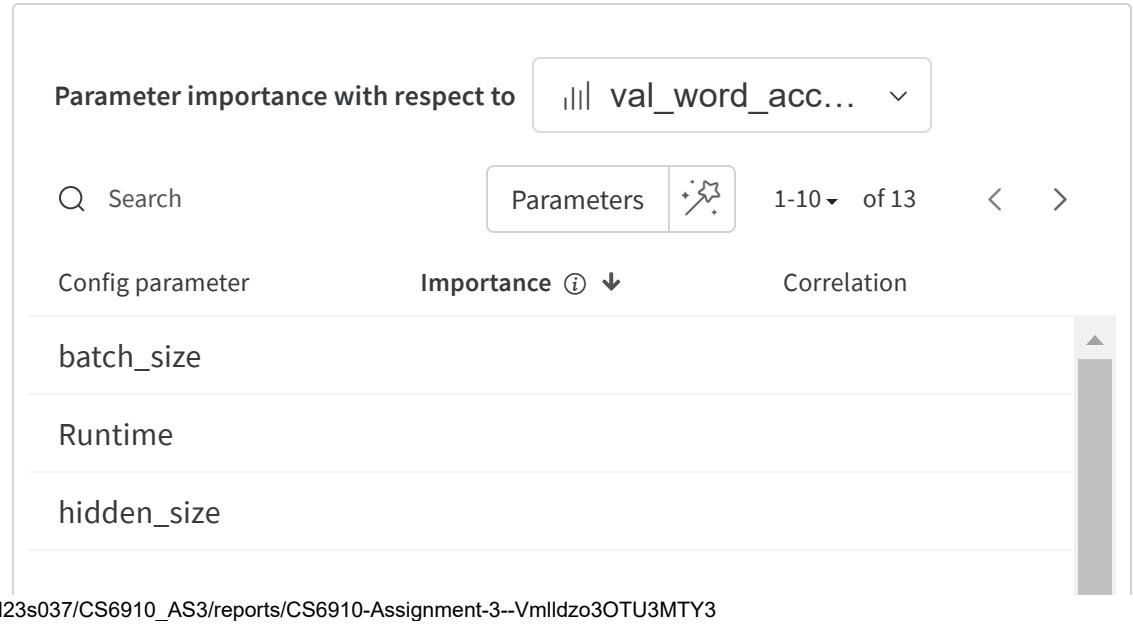
Solution

From the previous sweep, a few parameters are fixed. Cell type is fixed as GRU. All the runs are made with attention turned on. The following is the sweep config.

```
    sweep_config = {
        'method': 'bayes',
        'name': 'Q2_SWEEP_1',
        'metric': {
            'name': "val_word_accuracy",
            'goal': 'maximize',
        },
        'parameters': {
            'embedding_size': {'values': [16, 32, 64, 128, 256]}
            'num_layers': {'values': [1,2,3, 4]},
            'hidden_size': {'values': [16, 32, 64, 128, 256, 512
            'cell': {'value': 'gru'},
            'bidirectional': {'values': [False, True]},
            'beam_width': {'values': [1,2,3,4]},
            'dropout': {'values': [0.2, 0.3, 0.4, 0.5]},
            'learning_rate': {'values': [1e-2, 1e-3, 1e-4, 1e-5]
            'batch_size': {'values': [16, 32, 64, 128, 256, 512]
            "epochs": {'values': [5, 10, 15, 20]},
```

```
            'lang': {'value': 'tam'},
            'use_attention': {'value': True},
        },
    }
```

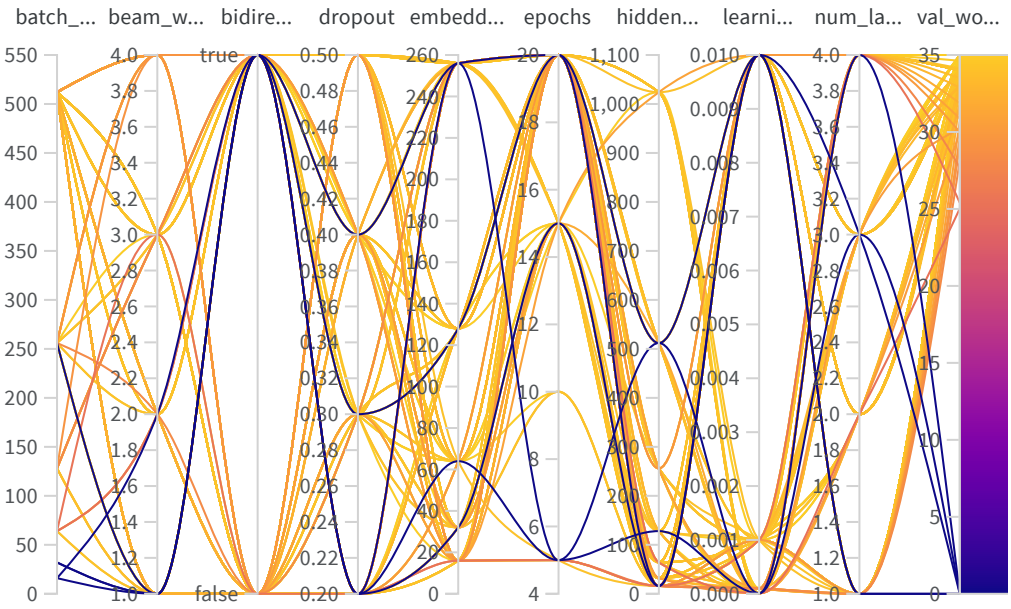




epochs

embedding_size

learning_rate



(b) Evaluate your best model on the test set and report the accuracy. Also, upload all the predictions on the test set in a folder `predictions_attention` on your GitHub project.

Solution

Accuracy on the test set is **35%**

Predictions for both the models are saved in the repository

- Attention based model: https://github.com/Gowthamaan-

P/cs6910_assignment3/blob/main/predictions_attention/predictions_attentions.csv

- Sequence-Sequence model: https://github.com/Gowthamaan-P/cs6910_assignment3/blob/main/predictions_seq/predictions_seq.csv

(c) Does the attention-based model perform better than the vanilla model? If so, can you check some of the errors that this model corrected and note down your inferences (i.e., outputs that were predicted incorrectly by your best seq2seq model are predicted correctly by this model)

Solution

All the following answers follow this pattern: English Word | Seq2Seq Model Output | Attention Model Output

1. Seq2Seq Model confuses with the thunaikal used in Tamil to indicate the longer sound. Attention outputs consider this and adds the thunaikal at the end.

   nalla | நல்ல | நல்லா

2. It makes mistakes for characters that have almost same pronunciation particularly vowel sounds.

   radaar | ரதார் | ராடார்

   thatavi | தத்வரி | தடவி

3. Most of the time Seq2Seq leaves out the final ending sound of the word.

   tough | டோக் | டோப் blue | பில் | பிலூ

4. Seq2Seq2 misses intermediate sounds in Tamil, whereas Attention based model captures this intermediate sounds.

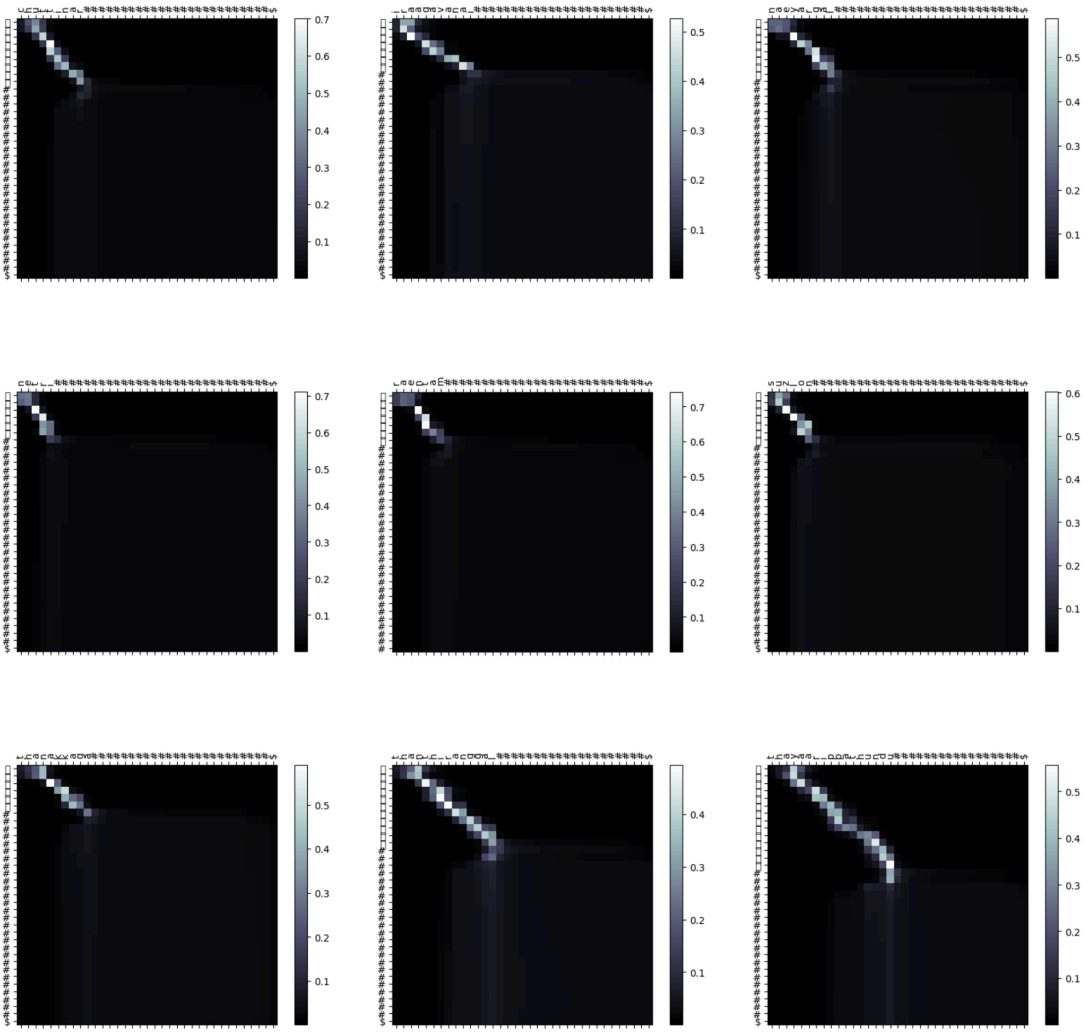
   araku | அரகு | அரக்கு

5. Seq2Seq predictions make error with words with continuously same sounds.

   russo | ரூஸ் | ரூச்சோ

(d) In a 3×3 grid, paste the attention heatmaps for 10 inputs from your test data (read up on what attention heatmaps are).

Solution



(UNGRADED, OPTIONAL) Question 6 (0 Marks)

(UNGRADED, OPTIONAL) Question 7 (0 Marks)

Question 8 (10 Marks)

Paste a link to your GitHub Link

Example: https://github.com/Gowthamaan-P/cs6910_assignment3.git

- We will check for coding style, clarity in using functions, and a `README` file with clear instructions on training and evaluating the model (the 10 marks will be based on this).
- We will also run a plagiarism check to ensure that the code is not copied (0 marks in the assignment if we find that the code is plagiarised).
- We will also check if the training and test splits have been used properly. You will get 0 marks on the assignment if we find any cheating (e.g., adding test data to training data) to get higher accuracy.

(UNGRADED, OPTIONAL) Question 9 (0 Marks)

Self-Declaration

I, **Gowthamaan** (**ED23S037**), swear on my honour that I have written the code and the report by myself and have not copied it from the internet or other students.

https://wandb.ai/ed23s037/CS6910_AS3/reports/CS6910-Assignment-3--Vmlldzo3OTU3MTY3