

Assignment 3

Use recurrent neural networks to build a transliteration system.

Pranav Agrawal

▼ Instructions

- The goal of this assignment is threefold: (i) learn how to model sequence-to-sequence learning problems using Recurrent Neural Networks (ii) compare different cells such as vanilla RNN, LSTM and GRU (iii) understand how attention networks overcome the limitations of vanilla seq2seq models
- Discussions with other students is encouraged.
- You must use `Python` for your implementation.
- You can use any and all packages from `PyTorch` or `PyTorch-Lightning`. NO OTHER DL library, such as `TensorFlow` or `Keras` is allowed.
- Please confirm with the TAs before using any new external library. BTW, you may want to explore [PyTorch-Lightning](#) as it includes `fp16` mixed-precision training, wandb integration, and many other black boxes eliminating the need for boiler-plate code. Also, do look out for [PyTorch2.0](#).
- You can run the code in a jupyter notebook on Colab/Kaggle by enabling GPUs.
- You have to generate the report in the same format as shown below using `wandb.ai`. You can start by cloning this report using the clone option above. Most of the plots we have asked for below can be (automatically) generated using the APIs provided by `wandb.ai`. You will upload a link to this report on Gradescope.
- You must also provide a link to your GitHub code, as shown below. Follow good software engineering practices and set up a GitHub repo

for the project on Day 1. Please do not write all code on your local machine and push everything to GitHub on the last day. The commits in GitHub should reflect how the code has evolved during the course of the assignment.

- You have to check Moodle regularly for updates regarding the assignment.

▼ Problem Statement

In this assignment, you will experiment with a sample of the [Aksharantar dataset](#) released by [AI4Bharat](#). This dataset contains pairs of the following form:

x, y

ajanabee, अजनबी

i.e., a word in the native script and its corresponding transliteration in the Latin script (how we type while chatting with our friends on WhatsApp etc). Given many such $(x_i, y_i)_{i=1}^n$ pairs your goal is to train a model $y = \hat{f}(x)$ which takes as input a romanized string (ghar) and produces the corresponding word in Devanagari (घर).

As you would realize, this is the problem of mapping a sequence of characters in one language to a sequence of characters in another. Notice that this is a scaled-down version of the problem of translation where the goal is to translate a sequence of **words** in one language to a sequence of words in another language (as opposed to a sequence of **characters** here).

Read this [blog](#) to understand how to build neural sequence-to-sequence models.

▼ 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

Generalized equation of RNN:

$$s_i = \text{sigmoid}(U_e * x + W_e s_{i-1} + b)$$

Equation of output

$$y = O(V_e * s_i + c)$$

$$U_e = (m \times k), W_e = (k \times k), V_e = (k \times v)$$

First character are mapped in the embedding layer. The order of computation in embedding layer is $O(V \times m)$

$$\text{Order of computation in } s_i \text{ equation} = O(m*k + k*k + k)$$

$$\text{Order of computation in sigmoid} = O(k)$$

$$\text{Order of computation in Encoder layer of a sequence of length } T = O(T*(V*m + m*k + k*k + 2k))$$

Decoder, in addition to encoder computation there is output computation and softmax computation.

$$\text{For linear transformation in output} = O(Vk + V)$$

Softmax computation = $O(V)$

Hence total computation in decoder layer = $O(T^*(V*m + m*k + k*k + 2k + Vk + 2V))$

(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

Number of parameters in the embedding layer: $V \times m$ (for each encoder and decoder)

Number of parameters in the encoder layer:

1. $U_e = m \times k$
2. $W_e = k \times k$

Number of parameters in the decoder layer:

1. $U_d = m \times k$
2. $W_d = k \times k$
3. $V_d = k \times V$

Total number of parameters = # parameters in the embedding layer * 2 + # parameters in the encoder layer + # parameters in the decoder layer

▸ 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)

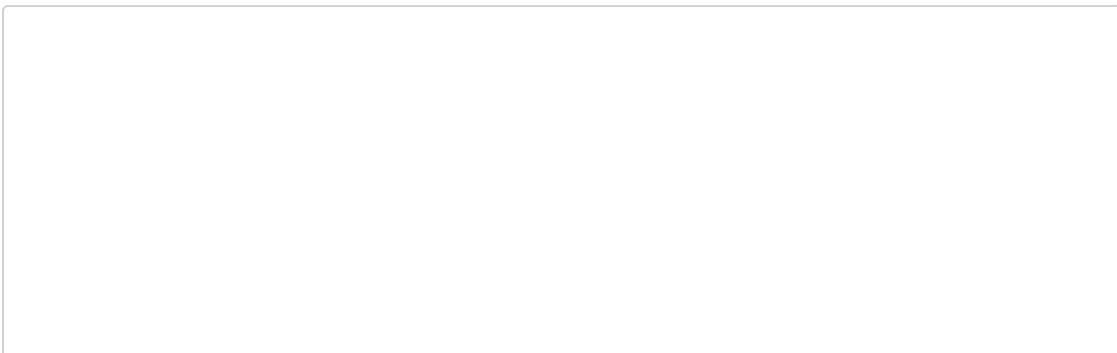
BTW, you should read up on how NLG models operate in inference mode, and how it is different from training. This [blog](#) might help you with it.

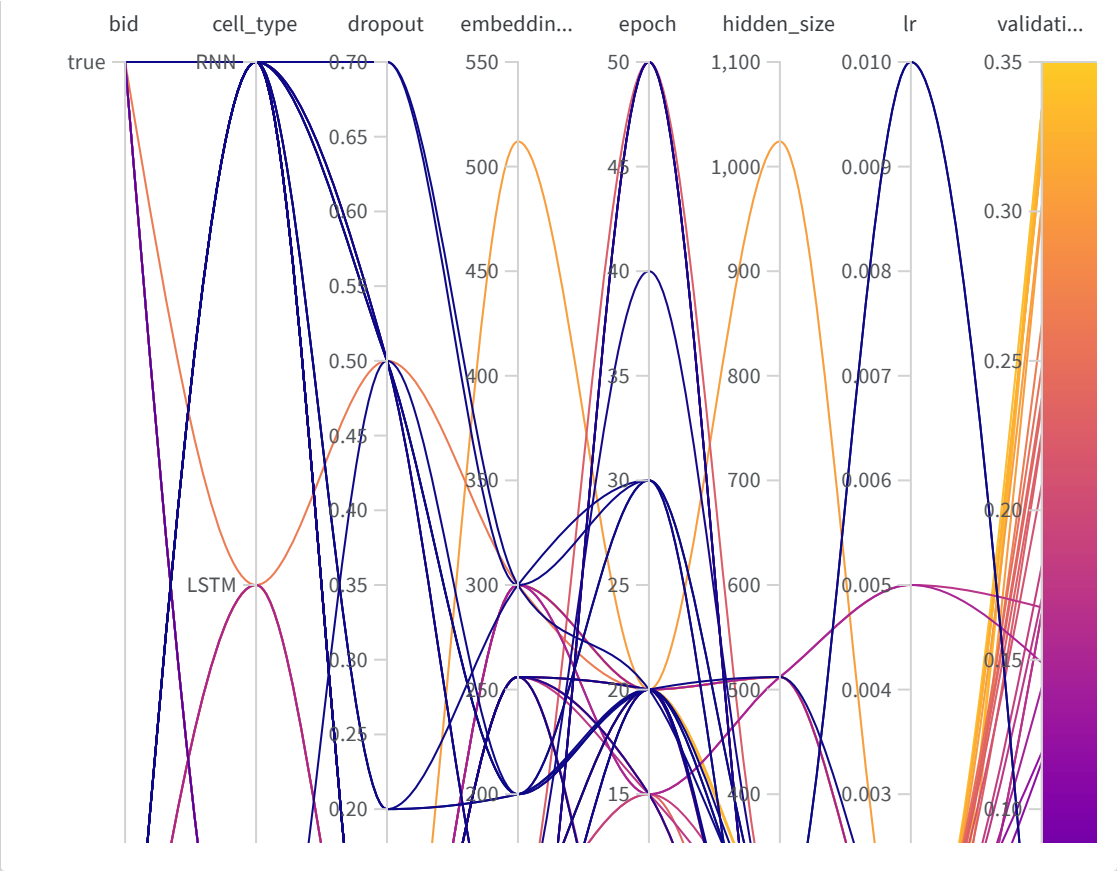
Using the sweep feature in wandb find the best hyperparameter configuration. Here are some suggestions, but you are free to decide which hyperparameters you want to explore

- 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)

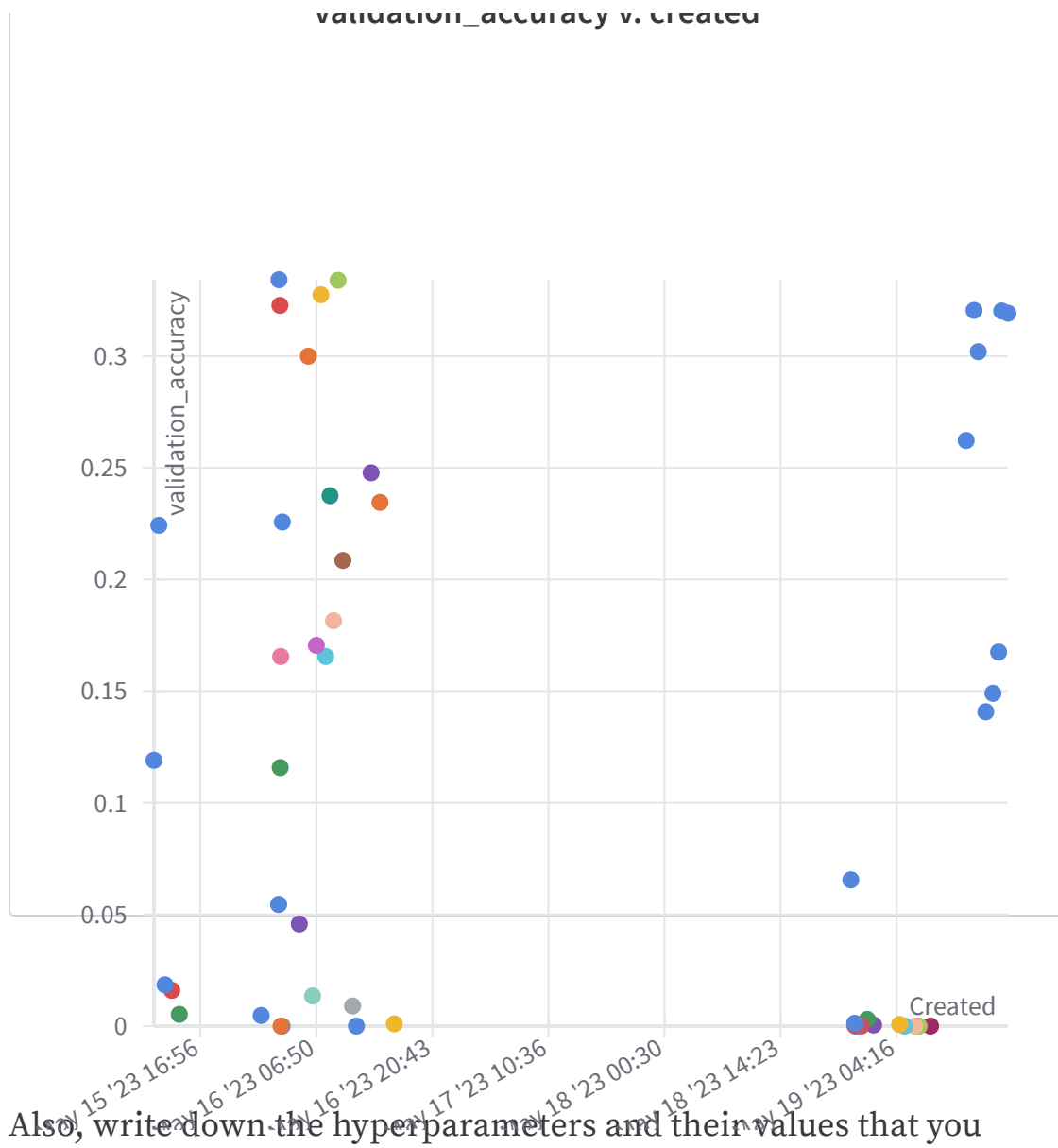




Parameter importance with respect to validation_acc...

Search Parameters 1-10 of 16

Config parameter	Importance ⓘ ↓	Correlation
cell_type.value_RNN		
lr		
Runtime		
num_layer		
epoch		
bid		
hidden_size		
cell_type.value_LSTM		



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

The hyperparameters and their values which for which sweeping is performed are as follows:

1. Hidden size: The hidden size ranges [32, 64, 256, 512, 1024].
2. Embedding size: [64, 256, 512, 300]
3. Number of layer: [1,2,3,4]
4. Cell types = ['RNN', 'LSTM', 'GRU']

5. Bidirectionality = [True, False]
6. Dropout = [0, 0.2, 0.5, 0.7]
7. Teacher forcing = [0.2, 0.5, 0.9]
8. Epochs = [30, 50]

Combination of grid and random sweeps were used. Extremes combination were tried initially like max hidden size and embedding layer and compared with lesser hidden size. It was observed that more complex model performs better. Similarly for the larger values all the cell type was run to certain that which cell gives higher accuracy. Followed by sweep over bidirectionality and dropout.

▼ Question 3 (15 Marks)

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

1. Larger **hidden size** gives better performance, however too large hidden size overfits the model. There is positive correlation in the correlation summary plot.
2. Larger **embedding size** gives better accuracy.
3. Compared to **RNN, GRU, LSTM**, it was observed that RNN performed worst among the lot and its correlation is negative in the correlation summary plot.

4. **Bidirectionality** has negative effect on the accuracy and can be confirmed from the negative correlation.
5. Presence of **dropout** doesn't increase the accuracy.
6. Using less **number of layer** doesn't give good results.
7. **LSTM** are slower however gives best accuracy among the three cell types.
8. Loss saturated after 20 **epochs** and model overfits the data.
9. **Teacher forcing** was not included as sweep parameter, however few values were tried. It was observed that teacher forcing of 0.2 and 0.5 gives similar outputs, however high teacher forcing of 0.9 gives lesser accuracy.

▼ Question 4 (10 Marks)

You will now apply your best model on the test data (You shouldn't have used test data so far. All the above experiments should have been done using train and val data only).

(a) 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).

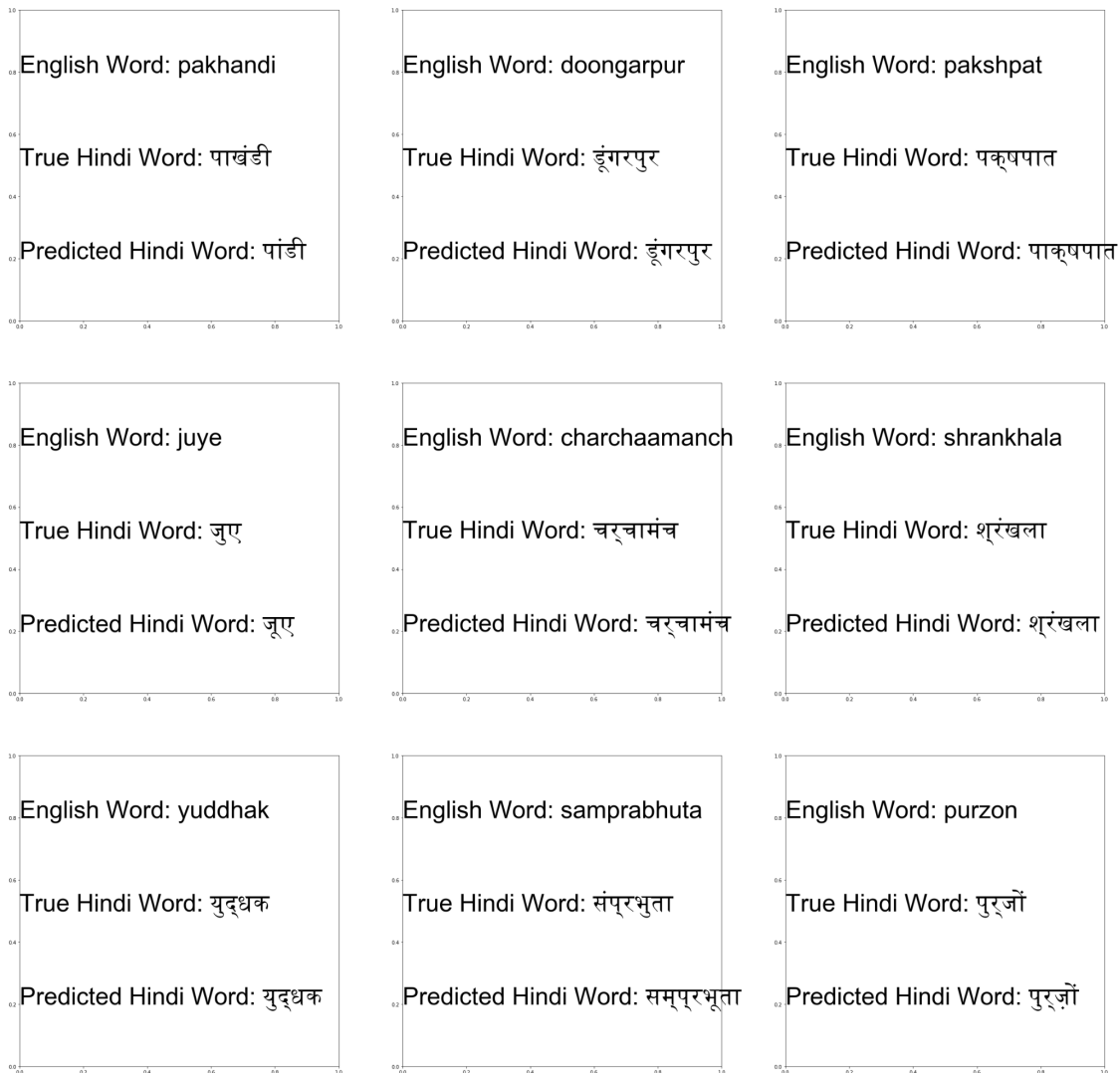
▼ Solution

The best model is with following parameters:

1. Hidden size = 512
2. Embedding size = 300
3. num_layers = 3 (both decoder and encoder)
4. bidirectionality = False
5. dropout = 0
6. cell type = 'LSTM'

The accuracy on the test data is **32 percent**.

(b) Provide sample inputs from the test data and predictions made by your best model (more marks for presenting this grid creatively). Also, upload all the predictions on the test set in a folder `predictions_vanilla` on your GitHub project.



(c) Comment on the errors made by your model (simple insightful bullet points)

- The model makes more errors on consonants than vowels
- The model makes more errors on longer sequences
- I am thinking confusion matrix, but maybe it's just me!

▾ Solution

1. The model makes mistake more frequently while translating the vowels like a, e for example in the above grid translating the english

words like pakhandi and juye, have made errors in 'a' and 'e'.

2. No such pattern was found where the model has made error in selectively the longer sequences. However this is a qualitative statement and needs to be established quantitatively.

▼ 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

The model was run with very few configuration and results were recorded locally. It was observed that model followed same behaviour as the vanilla encoder-decoder and hence no hyperparameter tuning was done.

(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

The test accuracy with the attention layer is 30 percent.

(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)

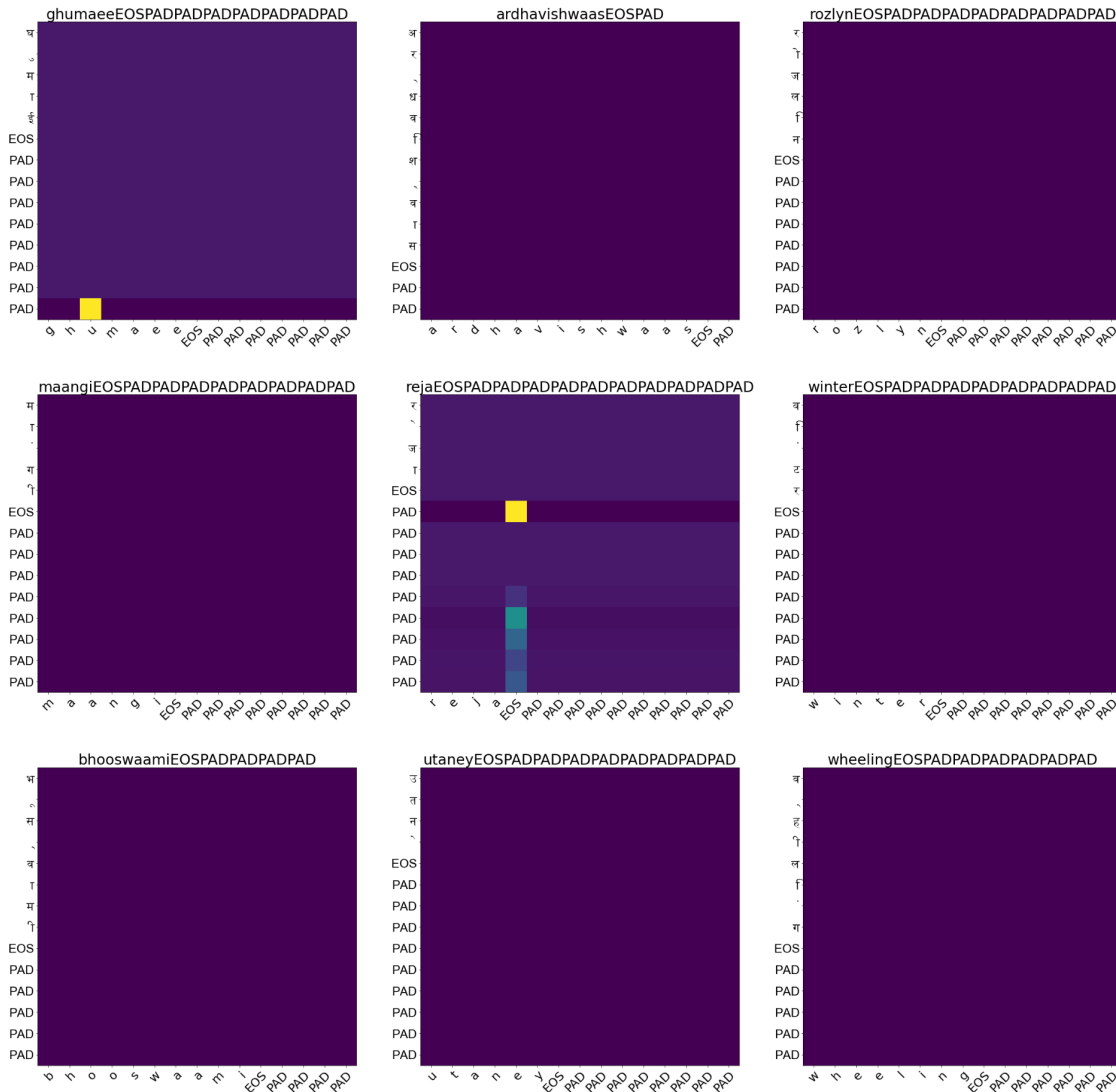
▼ Solution

The attention model didnt improve the accuracy of the model. This might be due to shorter sequences or due to large number of padding.

(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

Attention Heatmap



▼ (UNGRADED, OPTIONAL) Question 6 (0 Marks)

This is a challenging question, and most of you will find it hard. Note that this question has no marks and will not be graded. This is only for students looking for a challenge and wanting to get something more out of the course.

I like the visualization in the figure captioned "Connectivity" in this [article](#). Make a similar visualization for your model. For some starter code, please look at this [blog](#). The goal is to figure out the following:

When the model is decoding the i^{th} character in the output which is the input character that it is looking at?

▼ Question 7 (10 Marks)

Paste a link to your GitHub Link

- 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).

https://github.com/Pranav335/Seq2SeqModel_CH22D031

- 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 8 (0 Marks)

Note that this question has no marks and will not be graded. This is only for students looking for a challenge and wanting to get something more out of the course.

Your task is to finetune the GPT2 model to generate lyrics for English songs. You can refer to this [blog](#) and follow the steps there. This blog shows how to finetune the GPT2 model to generate headlines for financial articles. Instead of headlines, you will use lyrics so you may find the following datasets useful for training: [dataset1](#), [dataset2](#)

At test time, you will give it a prompt: `I love Deep Learning` and it should complete the song based on this prompt :-) Paste the generated song in a block below!

▼ Self-Declaration

I, Pranav (Roll no: CH22D031), swear on my honor that I have written the code and the report by myself and have not copied it from the internet or other students.

Created with  on Weights & Biases.

https://wandb.ai/pranav_/Assignment-3/reports/Assignment-3--Vmldzo0MTEzMzU2