In this assignment you will train and test sequence to sequence networks. We will work with *transliteration* as an example of a sequence to sequence task.

The goal of transliteration is to write a word in one language using the closest corresponding letters of a different alphabet or language (Oxford Dictionary). More formally, the goal of transliteration is to transform a name (a string of characters) in one language (and corresponding script) to the target language (and corresponding script) while ensuring phonemic equivalence preserving and conforming to the phonology and conventions of the target language. For instance, the transliteration of the English word `NEURAL` in Hindi is न्यूरल. The table below shows some examples of transliterations from English to some Indian languages.

`KERALA`	केरला	Hindi
`KERALA`	केरळा	Marathi
`PALANI`	பழநி	Tamil

If you want to know more, you can read a good survey by [1], though it is now slightly outdated. You can think of this as converting a sequence of characters (k, e, r, a, l, a) from one language to a sequence of characters in another language (, , ,)

You can read more about seq2seq models here : (`https://www.tensorflow.org/tutorials/seq2seq`)

In this assignment, you have to build a transliteration system from English to Hindi using the NEWS 2012 (Named Entities Workshop) shared task dataset [2].

Instructions

- Download the NEWS 2012 English-Hindi dataset from the Assignments section in moodle. The following is the data split (in number of words):- train: 14122, validation: 997, test: 1000.
- Using tensorflow, train a sequence-to-sequence model using the encoder-decoder architecture. The overall structure of the network is as follows:
  (a) **INEMBED**: A feedforward layer of size $|V_s| \times inembsize$, where V_s is the source language vocabulary (*i.e.* set of characters) and *inembsize* is the output size of the layer. Use $inembsize = 256$
  (b) **ENCODER**: bidirectional LSTM layer with *encsize* outputs in either direction. $encsize = 512$
  (c) **DECODER**: LSTM layer with *decsize* outputs. $decsize = 512$
  (d) **SOFTMAX**: softmax layer for classification: *decsize* inputs and V_t outputs, where V_t is the target language vocabulary (*i.e.* set of characters).
  (e) **OUTEMBED**: A feedforward layer of size $|V_t| \times outembsize$, where *embsize* is the output size of the layer. Use $outembsize = 256$. This layer is used for

obtaining the embedding of the output character and feeding it to the input of the decoder for the next time step.

(f) **ATTENTION**: Incorporate an attention mechanism in your network which will consist of: (i) a single feedforward network whose inputs are the previous decoder state, previous decoder output's embedding and the annotation vector (encoder output) under consideration. It outputs a single attention score. (ii) a softmax layer over the attention scores from each annotation vector to convert it to a probability value. The attention weights from the softmax layer are used to linearly interpolate the annotation vectors. The resulting *context vector* will be an input to the decoder along with the decoder output in the previous timestep.

The objective function is to minimize the negative log-likelihood. For inference, you should implement a greedy decoding. Additionally, you can also try to implement beam search. Implementation of beam search is optional (for extra credits).

- Use tanh-nonlinearities for the feedforward as well as LSTM layers.
- Train the network using Adam using the entire training dataset. Use the `valid` set for validation.
- Use dropout on the output of the encoder and the decoder when training the network. Do not use dropout while decoding.
- Use early stopping using the validation set with a patience of 5 epochs.
- Your code should support the following options:
  - --lr (initial learning rate η for gradient descent based algorithms)
  - --batch_size (the batch size to be used - valid values are 1 and multiples of 5)
  - --init (the initialization method to be used - 1 for Xavier, 2 for uniform random)
  - --dropout_prob (the probability of dropping a neuron)
  - --decode_method (0: greedy, 1: beam [default: 0])
  - --beam_width (the beam width, in case beam search is being used)
  - --save_dir (the directory in which the checkpoints will be saved)

  You should use the `argparse` module in python for parsing these parameters.
- The primary evaluation metric for the task is Accuracy (exact match). This means that the output is correct only if it matches the reference transliteration exactly. The task is to get an accuracy of 65% on the test data.

Submission Instructions:

- You need to submit the source code for the assignment. Your code should include one file called train.py which should be runnable using the following command:

  python train.py --lr 0.01 --batch_size 20 --init 1 --save_dir <some_dir> --dropout_prob <prob_value> --decode_method <value> --beam_width <value>

  All other supporting files used for generating plots, etc. should also be placed in the zip file.

- Prepare a report containing the following:
  - A plot of the learning curve showing iterations on the x-axis and negative log likelihood over labels on the y-axis. Make a single plot showing both the training loss and the validation loss.
  - The performance on the test data of the model that performs best on the validation data.
  - The parameter setting which gave you the best results.
  - Write down the dimensions of the input and output at each layer (for example, the input to INEMBED layer is $20 \times 50 \times 256$)
  - Did you try using only a unidirectional LSTM for the encoder? What was the effect?
  - What was the effect of using attention mechanism?
  - Plot a visualization of the attention layer weights for a sequence pair. Do you see meaningful character alignments? Do you see one-one, one-many, many-one alignments? Do you see non-contiguous alignments?
  - What was the effect of using dropout?
  - Is the early stopping criterion optimal? Try training for a few epochs beyond the early stopping epoch. Does the validation loss reduce?
  - Instead of using validation loss as early stopping criterion, you can also try using validation accuracy as stopping criterion. How do the two approaches compare?
  - How does beam search compare with greedy decoding?
  - What is the effect of beam width on beam search?

GPU Usage:

- If you want to use GPUs then you can register on AWS Educate [1] as a student using your iitm email id. Mention your University Name as "Indian Institute of Technology Madras" (no abbreviation). On registering you should get a credit of 100$. So each team will have 200 $. You can use these credits to rent EC2 GPU instances on the AWS cloud.

[1] `https://aws.amazon.com/education/awseducate/`

References:

1. Karimi, Sarvnaz, Falk Scholer, and Andrew Turpin. "Machine transliteration survey." *ACM Computing Surveys*, 2011.

2. Zhang, Min, Haizhou Li, Ming Liu, and A. Kumaran. "Whitepaper of news 2012 shared task on machine transliteration." *In Proceedings of the 4th Named Entity Workshop*, 2012.