CS349 - ARTIFICIAL INTELLIGENCE II

ASSIGNMENT-4: ENCODER-DECODER

(Read all the instructions carefully & adhere to them.)

Date: 11 November 2023 **Deadline: 19 November 2023**

Total Credit: 20

Instructions:

1. The assignment should be completed and uploaded by 17 November 2023, 11:59 PM IST.
2. Markings will be based on the correctness and soundness of the outputs. Marks will be deducted in case of plagiarism.
3. Proper indentation and appropriate comments are mandatory.
4. Make proper documentation of all results and observations with their analysis.
5. You are supposed to make a group of at most three members.
6. You should zip all the required files and name the zip file as roll_no_of_all_group_members .zip, e.g., 2101ai01_2102ai02_2101ai03.zip.
7. Upload your assignment (the zip file) in the google classroom with the class code: jtuplux
8. Course Webpage: https://ai-nlp-ml-iitp.github.io/ai-cs349/
9. For any queries regarding this assignment, you can contact:
   Priyanshu Priya (priyanshu528priya@gmail.com) or
   Manisha Boorja (manishaboorja@gmail.com)

Problem Statement: The objective is to convert an English sentence to its Hindi counterpart using a Neural Machine Translation (NMT) system.

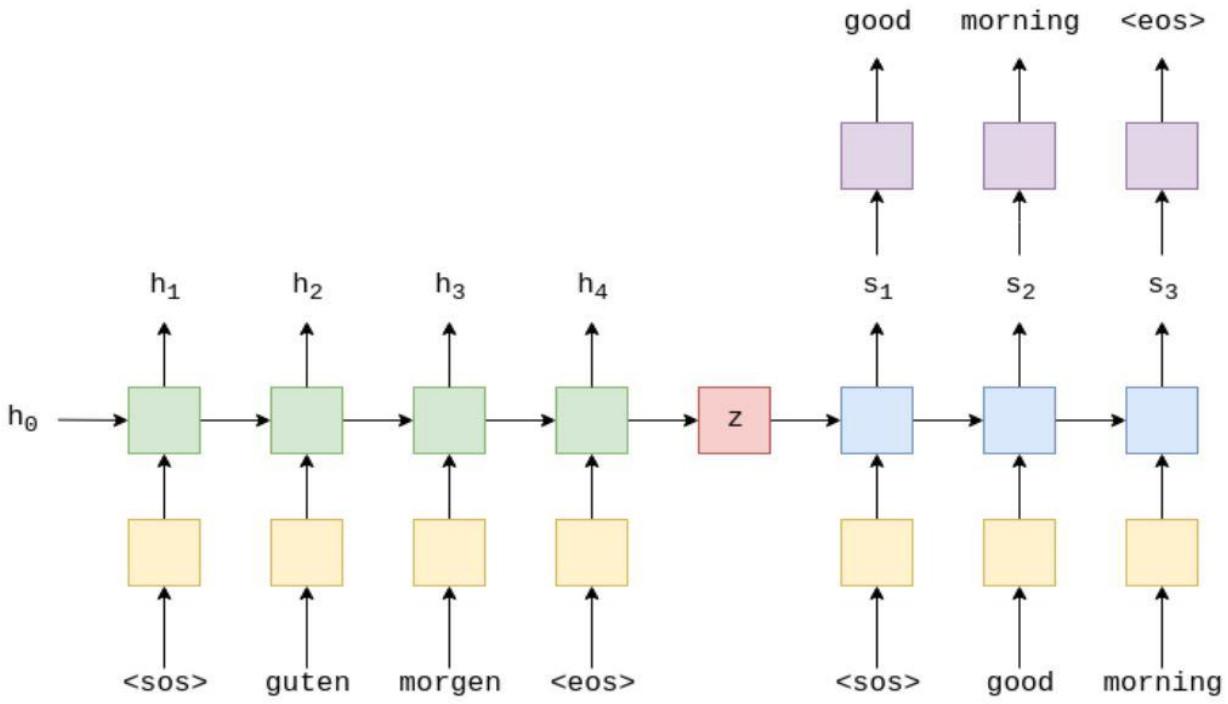
- **Input**: Given Sentence in English. A start of the sentence (<sos>) and end of the sentence (<eos>) token needs to be appended.
  - '<sos>', 'it', 'is',' raining', 'outside', '<eos>'
- Output: Corresponding translated sentences in Hindi. A start of the sentence (<sos>) and end of the sentence (<eos>) token needs to be appended.
  - '<sos>', 'बाहर', 'वर्षा','हो', 'रही', 'है' '<eos>'

Instructions:

- You may consider the following details for the implementation.
- Input Vec(Wi input at the encoder): The word embeddings of the words from the input sentences will be the input to the model. You can use the Word2Vec or GLOVE embedding.

- Output Vec(Wo Input at the decoder): The word embeddings of the words from the input sentences will be the input to the model. You can use the Word2Vec or GLOVE embedding.
  - Link → Word2vec: http://vectors.nlpl.eu/repository/20/5.zip or https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharig
  - Link→ Glove: http://nlp.stanford.edu/data/glove.840B.300d.zip
- Steps to use pre-trained word embeddings:
  - Prepare a dictionary of all the unique words in the dataset.
  - Load the word2vec or glove embeddings.
  - Get embeddings for each word and save them in a numpy or torch matrix.
- You may use any deep learning libraries such as TensorFlow, PyTorch, Keras etc. for the implementation. Use 300 dimensions for word embeddings.

- **Neural Model**
  - A Vanilla RNN based Encoder
  - A Vanilla RNN based Decoder



- **Dataset**: Download the dataset for Machine translation from here :
  - https://drive.google.com/file/d/1jvZxoMsfVDvupZMqTMx11aHmMQFPyWG4/view?usp=sharing
  - There are 3 files consisting of English and Hindi data

    - Use 30% of the data in the files ‘english.train.txt’ and ‘hindi.train.txt’ for training. The sentences in the two files are aligned.
    - Test your model using the 30% of the sentences of the file ‘english.test.txt’. The target sentences in Hindi are in `hindi.test.txt’
- **Evaluation Metrics:** Evaluate your model based on the following metrics:
    - BLEU score: BLEU looks at the overlap in the predicted and actual target sequences in terms of their n-grams. (Use the torchtext.data.metrics https://pytorch.org/text/stable/data_metrics.html for computing bleu)
    - Using the gold samples from `hindi.test.txt’ compute the BLEU score.
- **Loss Function**: Use the CrossEntropyLoss function since it calculates both the log softmax as well as the negative log-likelihood for the predicted tokens.
