Lab7: RNNs for next-word prediction

In this lab you will implement a very simple model for text generation, using one of 3 types of RNNs: vanilla RNN, GRU, LSTM.

The model is trained to predict the next word in the training sentences.

When we use the model for inference, we can repeatedly ask the model to predict the next word, given the previous word that was predicted.

Before you begin

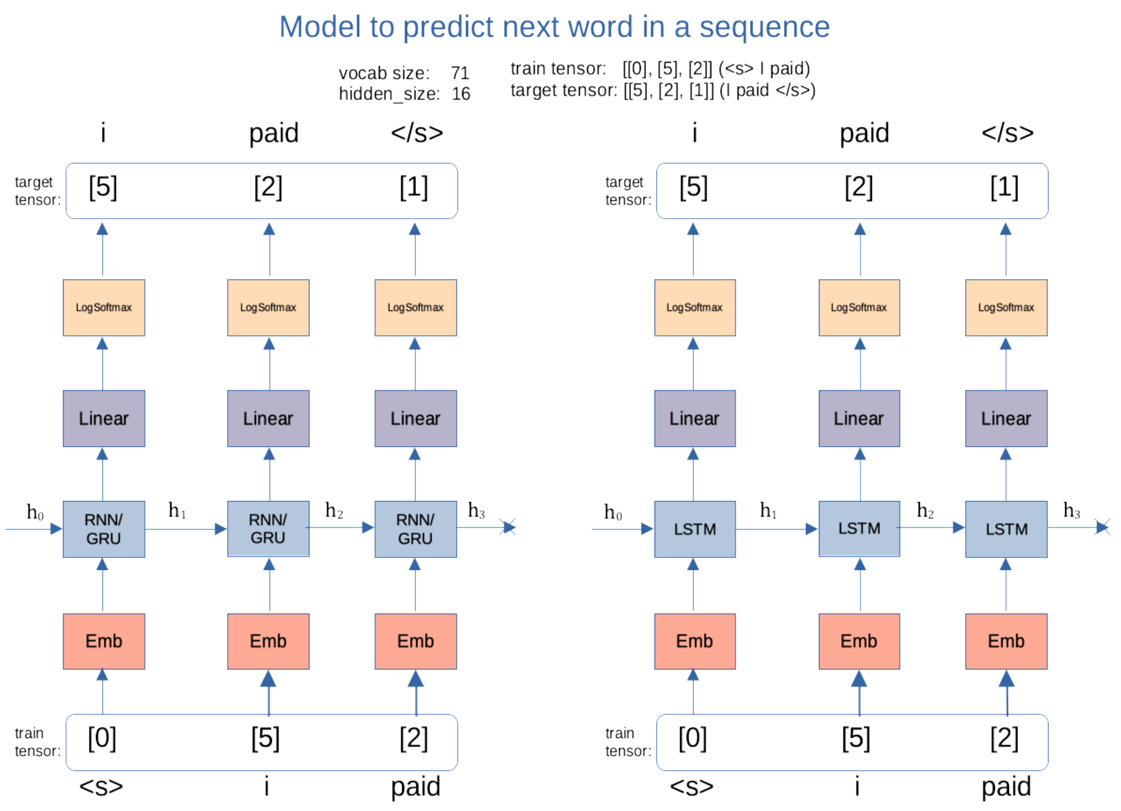
1. Create a new python project and a virtual environment for it in the usual way.
2. Add the code and data files in the starter code to your project.

Next-Word-Prediction: Basic Model Architecture

The model architecture is illustrated below. It's the same architecture that was used for the Decoder, but this time the model is always given the next token in target tensor (never its own output from the previous step).

Note that we don't need annotated training data for this task. The target output is simply the next word in the training sentence.

Note also that both the input and output have the same length and use the same vocabulary, so encoding is not necessary.



Task1: Text Generation

In main.py, 3 predict-next models are trained on the children's book *Green Eggs and Ham* by Dr. Seuss. The first model is trained with a vanilla RNN. The second model is trained with a GRU. The third model is trained with an LSTM.

After training, each model is used for generating sentences: The BOS token is used as the first input, and then the model is given its prediction from the previous time step as input to the next time step.

This continues until the EOS token is predicted or MAX_LENGTH is reached.

Implement *generate_text()*. For now, assume that the choose_random flag is False.

What do you notice about the generated texts?

Implement the choose_random flag in generate_text(). When the choose_random flag is True, instead of always choosing the model's top prediction (the highest one in the probability distribution), choose randomly from the model's top 3 predictions.

What do you notice about the texts generated this way?

Task2: LSTM Shapes

RNNs and GRUs can be used interchangeably, but LSTMs process more information and we need to adjust the inputs somewhat.

Fill in the missing shapes in the RNN/GRU and LSTM architectures in the picture above.

This is easily accomplished by setting appropriate breakpoints in the code and training a model of the type you're interested in.