Assignment 3: Recurrent Neural Networks

Homework assignments will be done individually: each student must hand in their own answers. Use of partial or entire solutions obtained from others or online is strictly prohibited. Electronic submission on Canvas is mandatory.

1. **Language Modeling** (100 points) In this homework, you need to implement a Recurrent Neural Network for language modeling with word embeddings on the Penn Treebank corpus. Please follow the steps as below:
   (a) (10pts) Preprocess the train and validation data, build the vocabulary, tokenize, etc.
   (b) (10pts) Initialize parameters for the model.
   (c) (20pts) Implement the forward pass for the model. Use an embeddings layer as the first layer of your network (i.e. `tf.nn.embedding_lookup`). Use a recurrent neural network cell (GRU or LSTM) as the next layer.
   (d) (20pts) Calculate the loss of the model (sequence cross-entropy loss is suggested) i.e. `tf.contrib.seq2seq.sequence_loss`.
   (e) (10pts) Calcuate the model's perplexity (perplexity is $\frac{\exp(\text{total loss})}{\text{number of predictions}}$).
   (f) (10pts) Set up the training step: use a learning rate of $1e-3$ and an adam optimizer.
   (g) (20pts) Train you model and report the perplexity (set window size to be 20, batch size to be greater than 1, e.g. 50).

   Checking on your model: To check how your model is doing (and to actually see the cool thing your model is doing), we recommend, but are not requiring, looking at the sentences your model is generating. To do so, you can append together the `argmax` of the logit predictions after each `sess.run()` to create a generated sentence representation. You can then convert each word index to its corresponding English word from your vocab to create readable sentences.