REPORT PROJECT 2

Name: Tsiatsios George

The project is separate in 3 parts. The three parts are as follows, basic statistics about the dataset, pre-processing stage and Train/Test phase.

Firstly, loading the dataset, which is included in the folder, and providing an overview. All answers and question sentences must be pre-proceeded to remove weird symbols and have a clear overview about sentences.

Secondly, Dataset statistics:

* Removing Symbols to load dataset
* Instead of having words like “I’m”, replacing them with “I am”, with an aim to remove unnecessary words and improve the format of the sentences.
* Provide useful statistics in order to show the length of each answer and question with a result to provide any pruning to the sentences with larger length.
* Provide useful statistics about word frequencies. Words with less frequency appear on top.

It is important before third step to create:

* **Lookup tables**
  + create two mapping tables
    - (key, value) == (unique word string, its unique index).
  + Since every model is mathematically represented, the input and the output (prediction) should also be represented as numbers. That is why this step is necessary for NLP problem because human readable text is not machine readable. This function takes a list of sentences and returns two mapping tables (dictionary data type). Along with the list of sentences, there are special tokens, <PAD>, <EOS>, <UNK>, and <GO> to be added in the mapping tables.
* **Text to word ids**
* convert each string word in the list of sentences to the index.
* From the Lookup Table we create texts with IDs of words:
  + [HI my name is] => [55 80 6 7], each word has unique representation.

Third, Apply pre-processing on the dataset:

* Create Vocabulary of words to keep unique words and times occurrence.
* Creation of the Lookup tables will help us to remove words with low frequency, we keep words with frequency up to 10, in order to create our Vocabulary.
* Cut sentences with length 2<=length<=20
* Add EOS to end of target words instead of “. “.
* “UNK” to the words that are not in the vocabulary of words.

The last Phase is, Create the Encoder-Decoder Network.

Placeholders creates and returns parameters (TF placeholders) related to the building model. Inputs placeholder will be fed with Questions, and its shape is [batch size, None]. The first variable means the batch size, and the batch size is unknown since the user can set it. The second None means the lengths of sentences. The maximum length of a sentence is different from batch to batch, so it cannot be set with the exact number. Inference placeholders with shape [1,None].And mask placeholder.

* Encoder\_seqs = the output encoder sentence
* Target\_seqs = the predicted decoder sentence
* Target\_mask= this is an array the same size as target\_seqs , but has 0 at the places where the padding is applied and 1 everywhere else.

The model:

* Purpose: Find the network output using the input encoder and decoder sentences
* The loss function is a cross entropy with applied mask to make sure each input sequence has the same length. The predicted output are taken from the preceding model output and are accessed using net\_out.outputs. The target\_seqs are the expected result for every input.The model’s optimizer AdamOptimizer and is defined using the built-in function from the Tensorflow tf.train.AdamOptimizer.
* Seq2seq model contains:
  + Encoder network
  + Decoder network
  + Seq2seq model combining above
  + The final model, fully connected dense layer producting the end result to calculate the final output of the network, the vocabulary size as the number of units.

Train Phase:

* Shuffle data , minibatches to split the data into sub-arrays each with the batch\_size.
* The first RNN is the encoder which accepts the encoder\_seqs as an input. We add padding to this sentence in order for it to to match the fixed length , max\_length\_sentence (remember 20)
* The second reccurent accepts the output of the first the encoded vector and returns the predicted result. The predicted output is compared with the target\_sequence.

Finally, initialize conversation!