COMP9444 assignment 2 report

student name: Hang Zhang

student id: z5153042

1. ## preprocessing

   For the **preprocessing** part, firstly changing all the characters to lower case and replacing "<br />" with a single space " ". Then removing all punctuations with a single space " ". Next removing all stop words in previous set and replacing more than one spaces to only one single space " ". Finally changing the whole review to list form using split(' ') function.

2. ## network design

   For the **defining graph** step, the $labels$ and $input\_data$ defined as a $placeholder$ with the size $[batch\_size, 2]$ (because there are only 2 classes negative and positive) and $[batch\_size, max\_words\_in\_review, embedding\_size]$ separately are going to be fed while being trained.

   The $dropout\_keep\_prob$ is defined as $placeholder$ as well but with default value.

   And this model is choosing a RNN(**recurrent neural networks**) using $LSTM$ (**long and short term memory**) units inside it. Because the **input data** is a sequence of words, each word in input data sequence will be associated with a specific time step. The number of time steps will be equal to the $max\_words\_in\_review$. The RNN model treat words in the sequence in order and give it a summary all of the information seen before. And the addition of $LSTM$ units make RNN model possible to determine the useful and correct information that needs to be stored in the model.

   Then warpping the $LSTM$ cell in a dropout layer to help prevent the network from overfitting.

   Next, feeding the $LSTM$ cell into a function called $tf.nn.dynamic\_rnn$. This function takes charge of unrolling the whole network and developing a pathway for the data to flow through the RNN graph. The first output of the $dynamic\ RNN$ function is the last hidden vector needed. And this vector is reshaped and then multiplied by a final weight matrix and a bias term to obtain the final output values ($logits$).

   Next, the $predict\_labels$ and the $correct\_predict$ are defined to track how the networks is doing. They works by looking at the index of the maximum value of the 2 output values, and then seeing whether it matches with the training labels.

   Finally, putting the **standard cross entropy loss with a softmax** layer on top of the final prediction values. For the optimizer, choosing the **Adam** optimizer and using 0.005 as the $learning\ rate$ because the speed of converge is neither fast nor slow.

Reference:

https://www.oreilly.com/learning/perform-sentiment-analysis-with-lstms-using-tensorflow