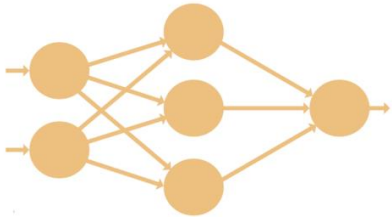
CS5590 APS - Deep Learning Programming

ASSIGNMENT 3

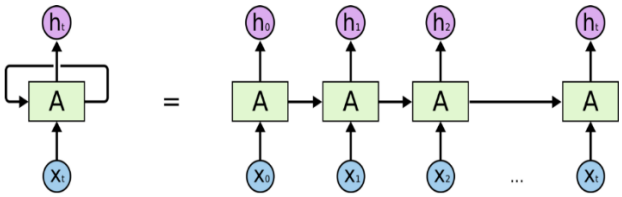
Abdoh Jabbari

Introduction

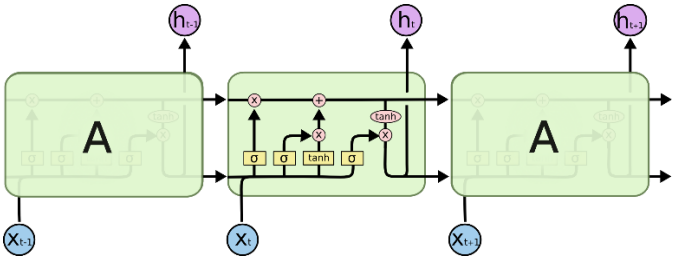
In deep learning, CNN is models has inputs of layers to start the processing to outputs layers. In between those two steps, there are also hidden layers that are convolutional layers such as pooling and fully connected layers.



RNN is the well-known model for text classification in which it uses its internal memory to do the output process in which allow it to remember impotent details about the inputs to use in the loop for the output process.



LSTM, which it was designed to solve the problem of long-term dependency which makes it remember the input for longer to predict the output.



Objectives

In this lab, we are going to use TensorFlow with CNN, RNN and LSTM models to analyze text file. Our objective in this lab to use CNN, RNN, and LSTM for text file classification to compare the results we get from them to check which one better for our datasets with higher accuracy and less loss.

Approaches/Methods

In this lab, we use TensorFlow to run the three models CNN, RNN, and LSTM. The method is using text classification with word embedding.

- loading the datasets
- Read the datasets and analyze it.
- Determining the number of features, and classes on the datasets.
- Defining placeholders and variables.
- Setting the parameters of the model.
- Training and testing the model.
- Optimizing to minimize the loss or error
- Optimizing to increase the accuracy

RNN approach

Data loading and building.

```
training_data = read_data(training_file)
print("Loaded training data...")


def build_dataset(words):
    count = collections.Counter(words).most_common()
    dictionary = dict()
    for word, _ in count:
        dictionary[word] = len(dictionary)
    reverse_dictionary = dict(zip(dictionary.values(), dictionary.keys()))
    return dictionary, reverse_dictionary
```

The parameters the number of Hadden layers

```
# Parameters
learning_rate = 0.001
training_iters = 10000
display_step = 1000
n_input = 3

# number of units in RNN cell
n_hidden = 512
```

Creating placeholders and the variables

```
# tf Graph input
x = tf.placeholder("float", [None, n_input, 1])
y = tf.placeholder("float", [None, vocab_size])

# RNN output node weights and biases
weights = {
    'out': tf.Variable(tf.random_normal([n_hidden, vocab_size]))
}
biases = {
    'out': tf.Variable(tf.random_normal([vocab_size]))
}
```

Creating the RNN function

```
def RNN(x, weights, biases):
    # reshape to [1, n_input]
    x = tf.reshape(x, [-1, n_input])

    # Generate a n_input-element sequence of inputs
    # (eg. [had] [a] [general] -> [20] [6] [33])
    x = tf.split(x, n_input, 1)
```

Running Basic RNN

```
    # RNN with no LSTM
    rnn_cell = rnn.BasicRNNCell(n_hidden)
    # generate prediction
    outputs, states = rnn.static_rnn(rnn_cell, x, dtype=tf.float32)

    # there are n_input outputs but
    # we only want the last output
    return tf.matmul(outputs[-1], weights['out']) + biases['out']


pred = RNN(x, weights, biases)
```

Loss optimizing and the model evaluation

```
# Loss and optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=pred, labels=y))
optimizer = tf.train.RMSPropOptimizer(learning_rate=learning_rate).minimize(cost)

# Model evaluation
correct_pred = tf.equal(tf.argmax(pred, 1), tf.argmax(y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# Initializing the variables
init = tf.global_variables_initializer()
```

LSTM approach

Using LSTM model (basic with a single layer)

```
  #RNN with BASIC LSTM
  rnn_cell = rnn.BasicLSTMCell(n_hidden)
```

Using LSTM model (basic with 2-layers)

```
#the 2-layer rnn.MultiRNNCell
rnn_cell = rnn.MultiRNNCell([rnn.BasicLSTMCell(n_hidden),rnn.BasicLSTMCell(n_hidden)])
```

CNN approach
For CNN, please refer the code files for details.

Datasets
In this lab, I used my own datasets, because I find it default to have one text datasets that can work for all the three models. I tried to keep the text datasets short so that the code can execute faster.

Evaluation & Discussion
Starting with RNN, I execute the basic rnn with no LSTM, the result showing in figure1 achieved 44%. That is because with basic rnn the model has no memory to use that allows it to save some useful details about the inputs to get better outputs. However, running the same data through the RNN with LSTM it has changed the accuracy to 89% as shown in figure 2. With the same dataset we use it to test it on CNN, as the results shown in figure 3, the accuracy is 100%, and loss is 2. That is due the datasets types and style.

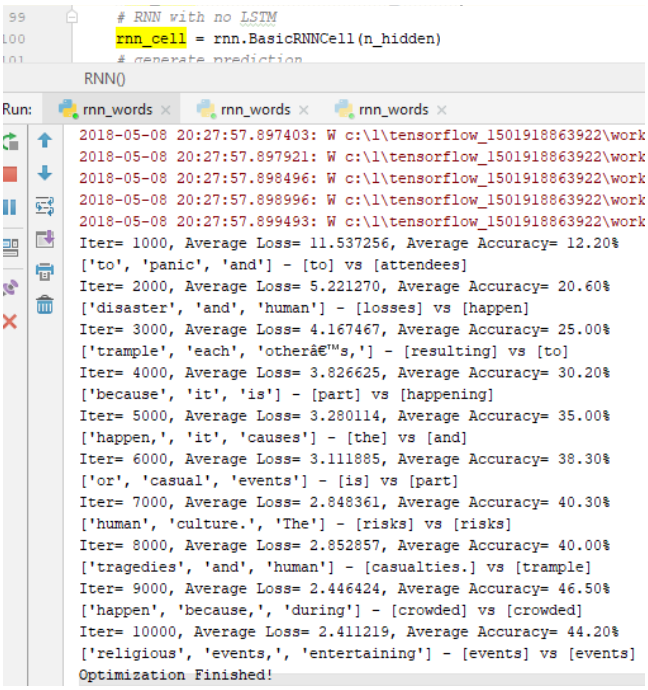


Figure 1, RNN without using LSTM

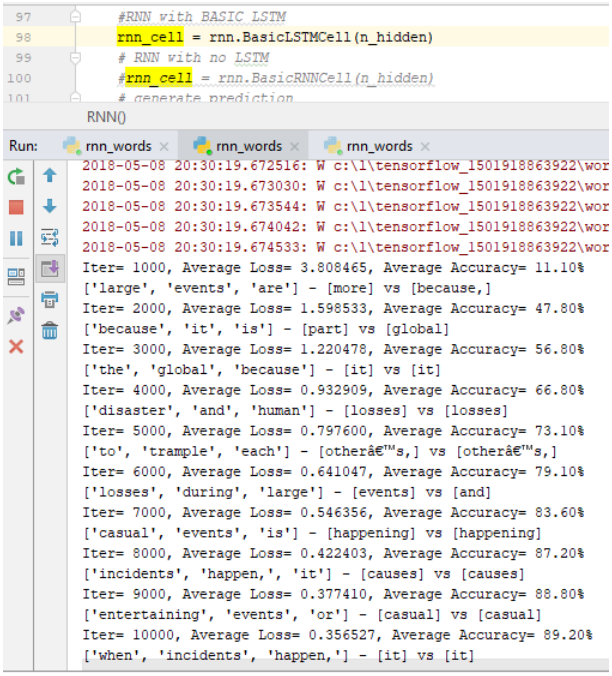


Figure 2, RNN with LSTM.

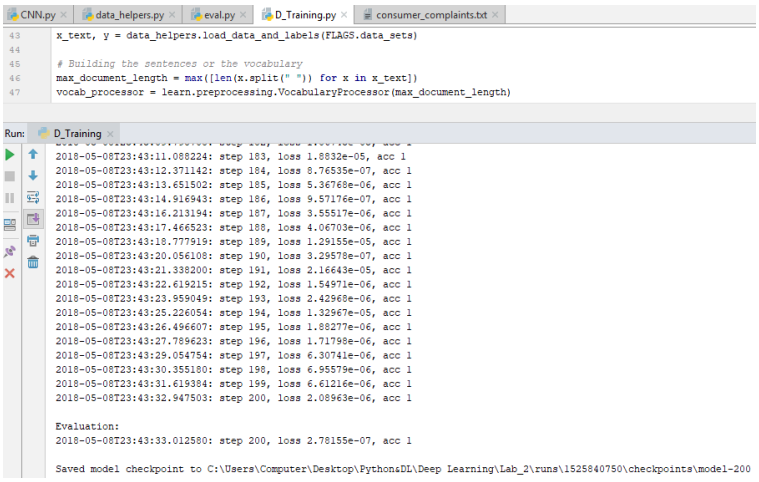
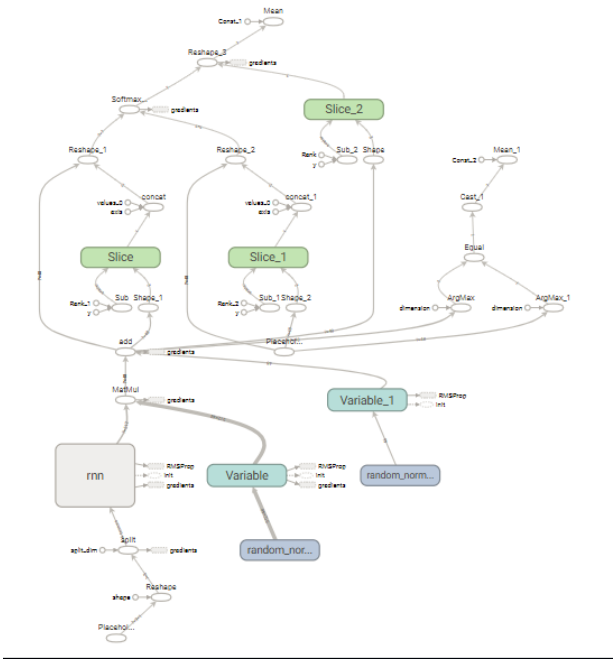


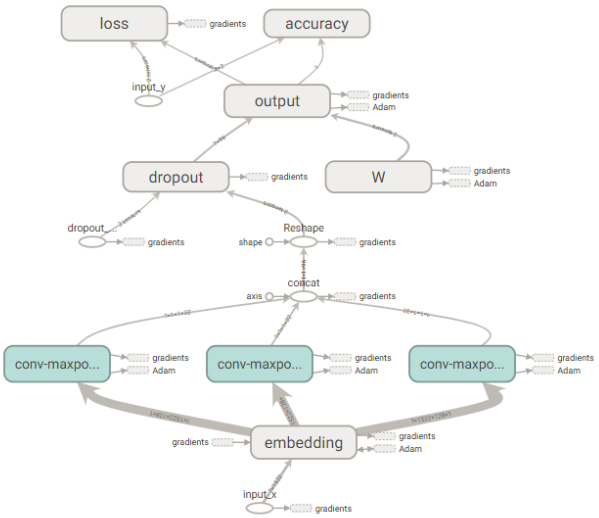
Figure 3, CNN

Workflow

The RNN and LSTM workflow



CNN Flow Work



Parameters

RNN and LSTM Parameters are:

```
# Parameters
learning_rate = 0.001
training_iters = 10000
display_step = 1000
n_input = 3

# number of units in RNN cell
n_hidden = 512
```

CNN Parameters

BATCH_SIZE=64
CHECKPOINT_EVERY=100
DEV_SAMPLE_PERCENTAGE=0.1
DROPOUT_KEEP_PROB=0.5
EMBEDDING_DIM=128
EVALUATE_EVERY=100
FILTER_SIZES=3,4,5
L2_REG_LAMBDA=0.0
NUM_CHECKPOINTS=5
NUM_EPOCHS=200
NUM_FILTERS=128

Conclusion

After running CNN, RNN, and SLTM with the same datasets to classify the text in the file and to check which model is better for my text datasets. Based on the results we got using RNN with LSTM give better accuracy and less loss compared to the other models. Also, I run the RNN with multiple layers LSTM, and I got a little bit better result than RNN with basic LSTM, Figure

4. However, to decide which models are better for text classification, it will depend on the text dataset style, negative and positive reviews, yes or no kind of reviews, story, etc.

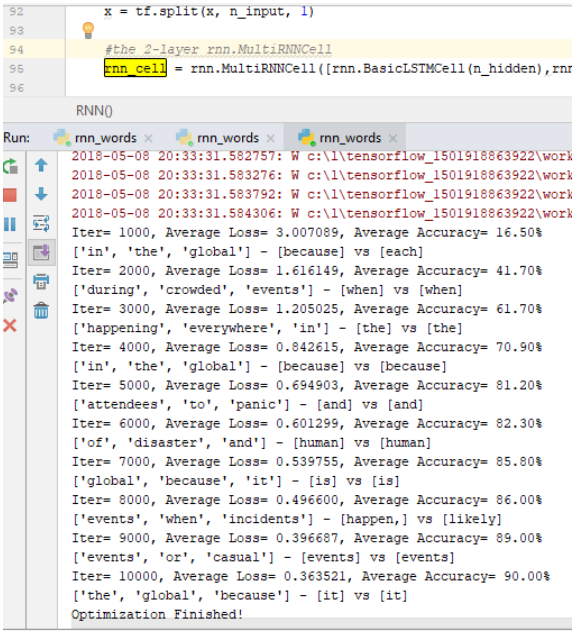


Figure 4, 2-layers LSTM

References

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/

https://github.com/dennybritz/cnn-text-classification-tf/tree/master/data/rt-polaritydata

https://github.com/jiegzhan/multi-class-text-classification-cnn

https://towardsdatascience.com/how-to-do-text-classification-using-tensorflow-word-embeddings-and-cnn-edae13b3e575