

Deep Learning Lab Assignment-2

Name: Saketh Garuda

ID: 16

Introduction:

This lab assignment helps in learning about the working of tensor flow and implementation of text classification with CNN model using python. Visualization charts and graphs are created and displayed in tensor board.

Objective:

The main objective for this lab assignment is to know the

- Implementation of text classification with CNN model
- Creating visualization graphs using tensor board
- To change the hyper parameters such as learning rate and get the results

Configuration:

- PyCharm IDE
- Python 3.6.4
- TensorFlow

Approaches/Methods:

The CNN model is implemented using python libraries 3.6, tensorflow and visualization graphs are created using tensor board.

Workflow:

The workflow for the entire model is as follows,

- Loading dataset using import library modules
- Processing data
- Performing CNN operations on data
- Calculating loss and predicting accuracy
- Using session graph for creating visualization graphs in tensorboard.

Dataset:

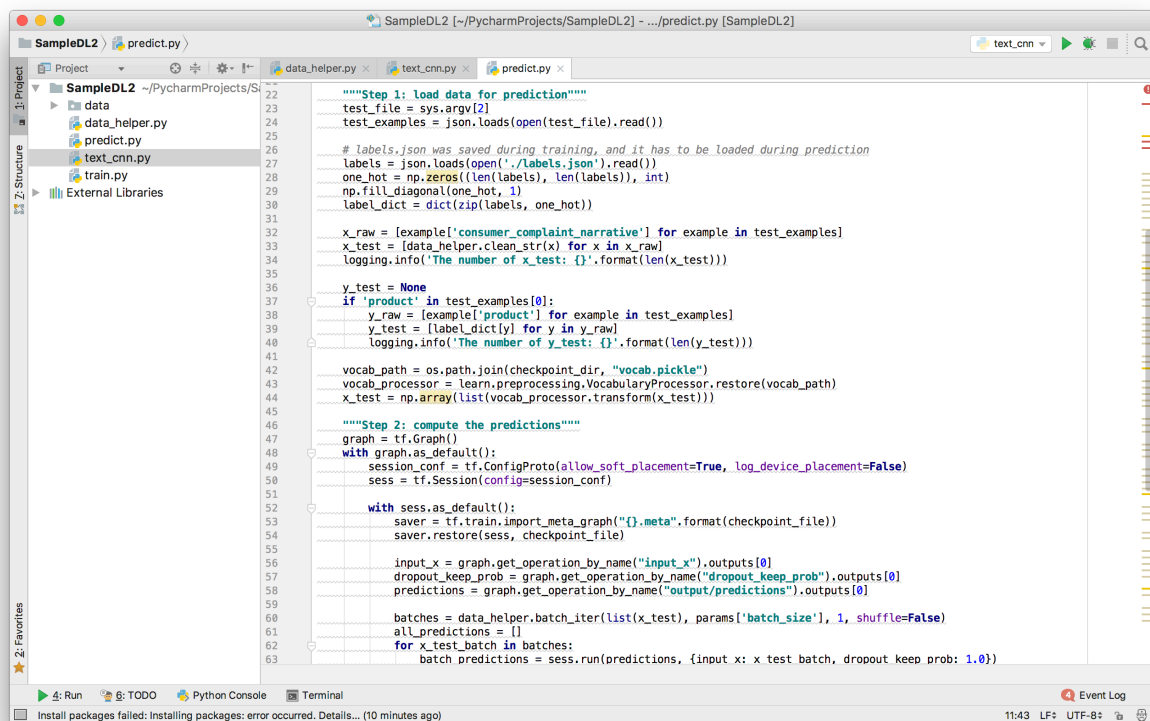
Consumer complaints dataset is used in this exercise, which consists of 11 classes.

Parameters:

The parameters to be considered while constructing the model are,

- Number of epochs
- Batch Size
- Embedding dimensions
- Size of filter
- Probability check

Evaluation and Discussion:



```
22 """Step 1: load data for prediction"""
23 test_file = sys.argv[2]
24 test_examples = json.loads(open(test_file).read())
25
26 # labels.json was saved during training, and it has to be loaded during prediction
27 labels = json.loads(open('./Labels.json').read())
28 one_hot = np.zeros((len(labels), len(labels)), int)
29 np.fill_diagonal(one_hot, 1)
30 label_dict = dict(zip(labels, one_hot))
31
32 x_raw = [example['consumer_complaint_narrative'] for example in test_examples]
33 x_test = [data_helper.clean_str(x) for x in x_raw]
34 logging.info('The number of x_test: {}'.format(len(x_test)))
35
36 y_test = None
37 if 'product' in test_examples[0]:
38     y_raw = [example['product'] for example in test_examples]
39     y_test = [label_dict[y] for y in y_raw]
40     logging.info('The number of y_test: {}'.format(len(y_test)))
41
42 vocab_path = os.path.join(checkpoint_dir, "vocab.pickle")
43 vocab_processor = learn.preprocessing.VocabularyProcessor.restore(vocab_path)
44 x_test = np.array(list(vocab_processor.transform(x_test)))
45
46 """Step 2: compute the predictions"""
47 graph = tf.Graph()
48 with graph.as_default():
49     session_conf = tf.ConfigProto(allow_soft_placement=True, log_device_placement=False)
50     sess = tf.Session(config=session_conf)
51
52     with sess.as_default():
53         saver = tf.train.import_meta_graph("{}_meta".format(checkpoint_file))
54         saver.restore(sess, checkpoint_file)
55
56         input_x = graph.get_operation_by_name("input_x").outputs[0]
57         dropout_keep_prob = graph.get_operation_by_name("dropout_keep_prob").outputs[0]
58         predictions = graph.get_operation_by_name("output/predictions").outputs[0]
59
60         batches = data_helper.batch_iter(list(x_test), params['batch_size'], 1, shuffle=False)
61         all_predictions = []
62         for x_test_batch in batches:
63             batch_predictions = sess.run(predictions, {input_x: x_test_batch, dropout_keep_prob: 1.0})
```



```

# Data Loading params
tf.flags.DEFINE_float("dev_sample_percentage", .2, "Percentage of the training data to use for validation")
tf.flags.DEFINE_string("traindata", "./data/traindata", "Data source for the training data.")
tf.flags.DEFINE_string("testdata", "./data/testdata", "Data source for the testing data.")

# Model Hyperparameters
tf.flags.DEFINE_integer("embedding_dim", 128, "Dimensionality of character embedding (default: 128)")
tf.flags.DEFINE_string("filter_sizes", "3,4,5", "Comma-separated filter sizes (default: '3,4,5')")
tf.flags.DEFINE_integer("num_filters", 128, "Number of filters per filter size (default: 128)")
tf.flags.DEFINE_float("dropout_keep_prob", 0.5, "Dropout keep probability (default: 0.5)")
tf.flags.DEFINE_float("l2_reg_lambda", 0.0, "L2 regularization lambda (default: 0.0)")

# Training parameters
tf.flags.DEFINE_integer("batch_size", 32, "Batch Size (default: 64)")
tf.flags.DEFINE_integer("num_epochs", 40, "Number of training epochs (default: 200)")
tf.flags.DEFINE_integer("evaluate_every", 100, "Evaluate model on dev set after this many steps (default: 100)")
tf.flags.DEFINE_integer("checkpoint_every", 100, "Save model after this many steps (default: 100)")
tf.flags.DEFINE_integer("num_checkpoints", 5, "Number of checkpoints to store (default: 5)")
# Misc Parameters
tf.flags.DEFINE_boolean("allow_soft_placement", True, "Allow device soft device placement")
tf.flags.DEFINE_boolean("log_device_placement", False, "Log placement of ops on devices")

```

```

# Training parameters
tf.flags.DEFINE_integer("batch_size", 128, "Batch Size (default: 64)")
tf.flags.DEFINE_integer("num_epochs", 10, "Number of training epochs (default: 200)")
tf.flags.DEFINE_integer("evaluate_every", 100, "Evaluate model on dev set after this many steps (default: 100)")
tf.flags.DEFINE_integer("checkpoint_every", 100, "Save model after this many steps (default: 100)")
tf.flags.DEFINE_integer("num_checkpoints", 5, "Number of checkpoints to store (default: 5)")

```

```

# Training parameters
tf.flags.DEFINE_integer("batch_size", 64, "Batch Size (default: 64)")
tf.flags.DEFINE_integer("num_epochs", 20, "Number of training epochs (default: 200)")
tf.flags.DEFINE_integer("evaluate_every", 100, "Evaluate model on dev set after this many steps (default: 100)")
tf.flags.DEFINE_integer("checkpoint_every", 100, "Save model after this many steps (default: 100)")
tf.flags.DEFINE_integer("num_checkpoints", 5, "Number of checkpoints to store (default: 5)")

```

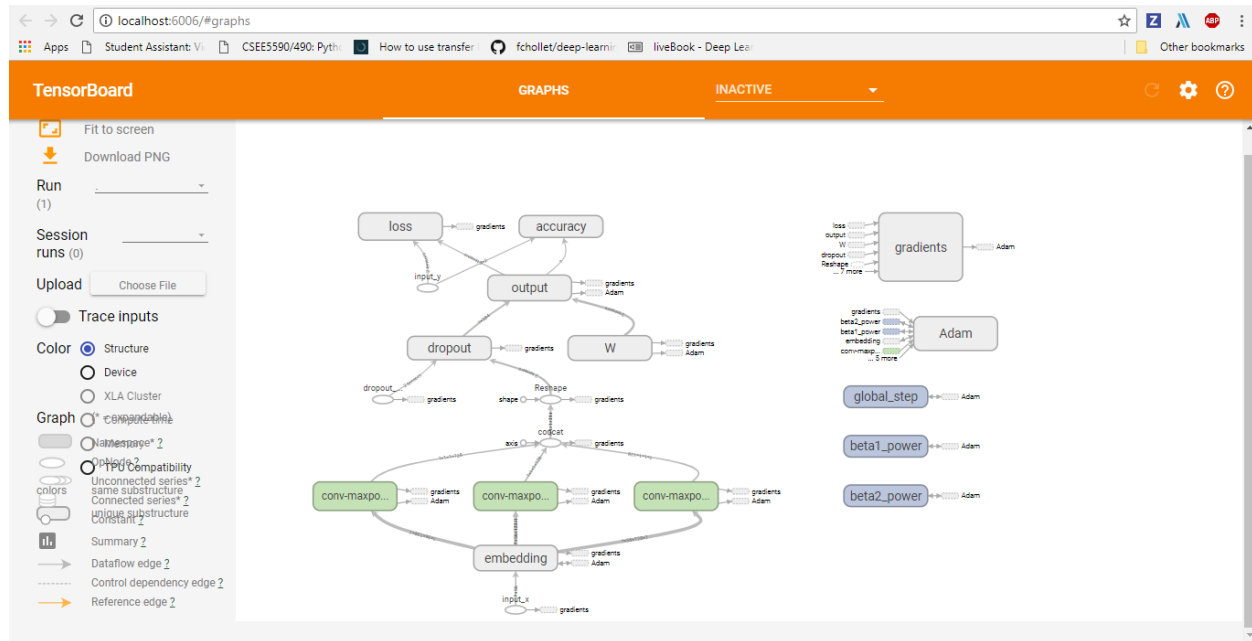
```

def train_step(x_batch, y_batch):
    """
    A single training step
    """
    feed_dict = {
        cnn.input_x: x_batch,
        cnn.input_y: y_batch,
        cnn.dropout_keep_prob: FLAGS.dropout_keep_prob
    }
    _, step, summaries, loss, accuracy = sess.run(
        [train_op, global_step, train_summary_op, cnn.loss, cnn.accuracy],
        feed_dict)
    time_str = datetime.datetime.now().isoformat()
    print("{}: step {}, loss {:g}, acc {:g}".format(time_str, step, loss, accuracy))
    train_summary_writer.add_summary(summaries, step)

```

```
C:\WINDOWS\system32\cmd.exe
2018-04-23T19:52:57.264350: step 726, loss 0.359526, acc 0.835938
2018-04-23T19:52:58.147991: step 727, loss 0.248865, acc 0.898438
2018-04-23T19:52:59.109674: step 728, loss 0.315458, acc 0.851562
2018-04-23T19:53:00.039832: step 729, loss 0.302336, acc 0.890625
2018-04-23T19:53:01.040043: step 730, loss 0.304672, acc 0.890625
2018-04-23T19:53:02.037272: step 731, loss 0.296508, acc 0.882812
2018-04-23T19:53:03.089019: step 732, loss 0.280337, acc 0.875
2018-04-23T19:53:04.043194: step 733, loss 0.329338, acc 0.867188
2018-04-23T19:53:04.910310: step 734, loss 0.37443, acc 0.820312
2018-04-23T19:53:05.848476: step 735, loss 0.416879, acc 0.8125
2018-04-23T19:53:06.755118: step 736, loss 0.386182, acc 0.828125
2018-04-23T19:53:07.576701: step 737, loss 0.318781, acc 0.851562
2018-04-23T19:53:08.395204: step 738, loss 0.35367, acc 0.84375
2018-04-23T19:53:09.224306: step 739, loss 0.305283, acc 0.859375
2018-04-23T19:53:10.026387: step 740, loss 0.366216, acc 0.84375
2018-04-23T19:53:10.857480: step 741, loss 0.306961, acc 0.890625
2018-04-23T19:53:11.786636: step 742, loss 0.309526, acc 0.898438
2018-04-23T19:53:12.647746: step 743, loss 0.335762, acc 0.84375
2018-04-23T19:53:13.480337: step 744, loss 0.395916, acc 0.804688
2018-04-23T19:53:14.291912: step 745, loss 0.305199, acc 0.882812
2018-04-23T19:53:15.103596: step 746, loss 0.359615, acc 0.84375
2018-04-23T19:53:15.925205: step 747, loss 0.297534, acc 0.898438
2018-04-23T19:53:16.753310: step 748, loss 0.348543, acc 0.867188
2018-04-23T19:53:17.657951: step 749, loss 0.302975, acc 0.859375
2018-04-23T19:53:18.566594: step 750, loss 0.361681, acc 0.870968
```

When we change the hyper parameters the as decrease in intervals and increase in batch size then the accuracy decreases accordingly and vice versa. Increasing the batch size to larger number can increase accuracy to greater scale.



References:

- <https://www.tensorflow.org/tutorials/wide>
- <https://www.kaggle.com/cfpb/us-consumer-finance-complaints>
- <https://github.com/jiegzhan/multi-class-text-classification-cnn>