Text Classification with TensorFlow

Jinpeng Zhu

About Me

- Senior Software Engineer
- Working in Huawei since 2011
- Graduated from XiDian University in 2011
- WeChat Public Account: ***deepinthinking***
- Email: zhujinpengvimer@qq.com

Contents

- Basics of Deep Learning
- Text Classification with Low Level API of TensorFlow
- Text Classification with Keras, High Level API of TensorFlow

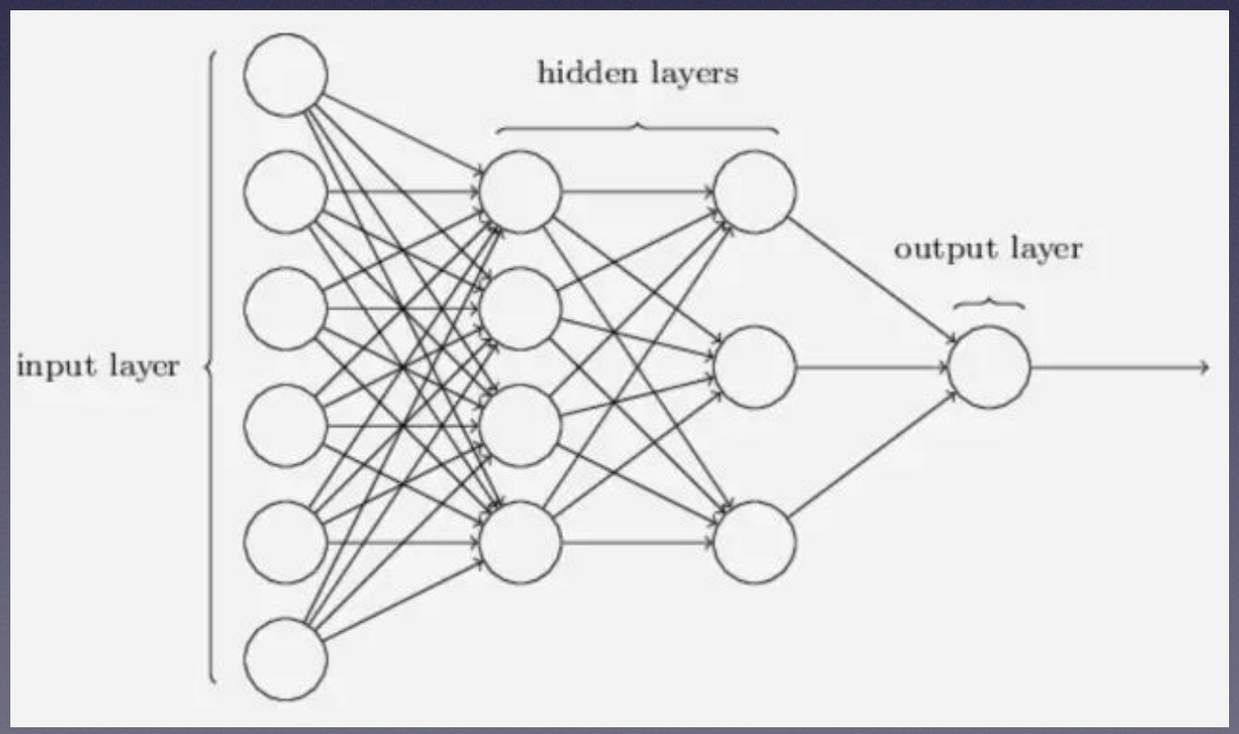
All codes will be represented with ipython notebook

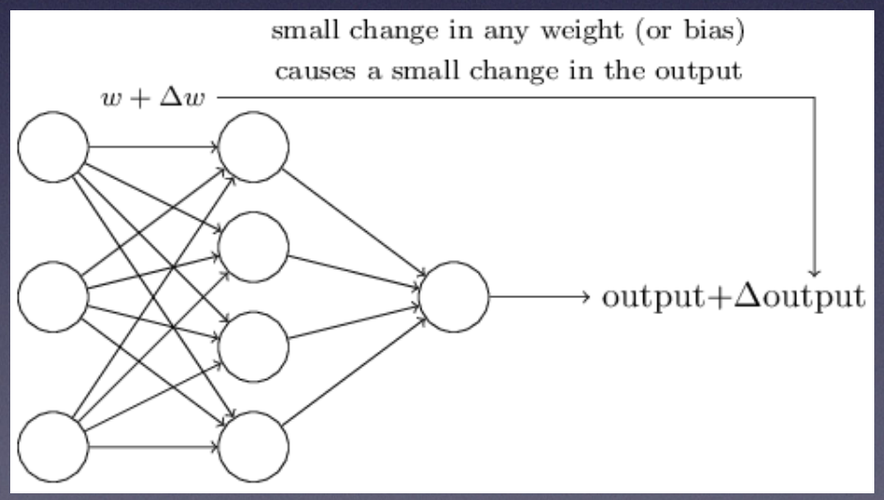
https://github.com/HarryHa/Lightning_Talk_2017

Basics of Deep Learning

What is Deep Learning

- Neural networks, a beautiful biologically-inspired programming paradigm which enables a computer to learn from observational data;
- Deep learning, a powerful set of techniques for learning in neural networks.

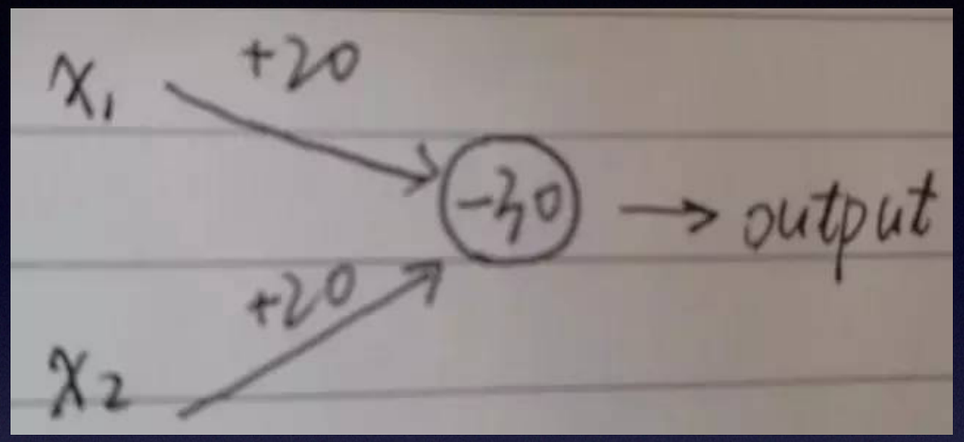


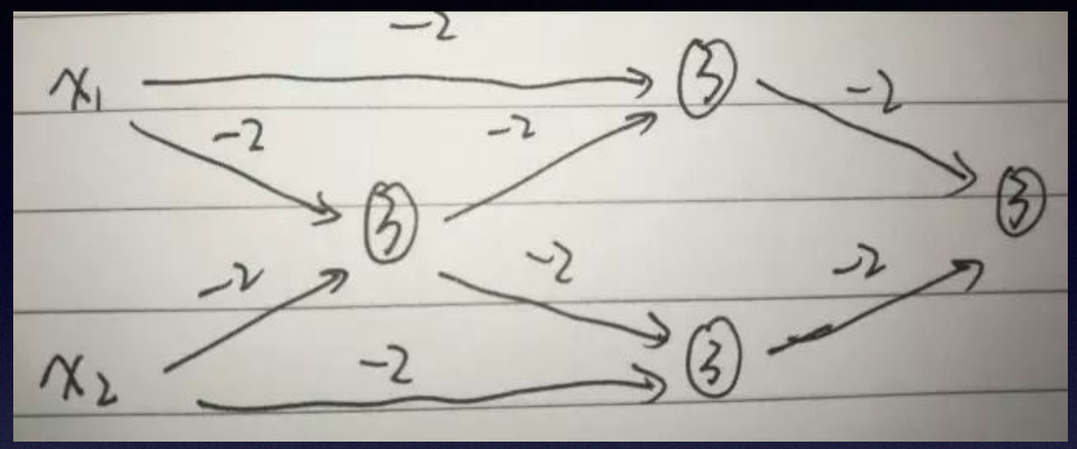


$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

http://neuralnetworksanddeeplearning.com/

Calculations of Deep Learning





OR gate

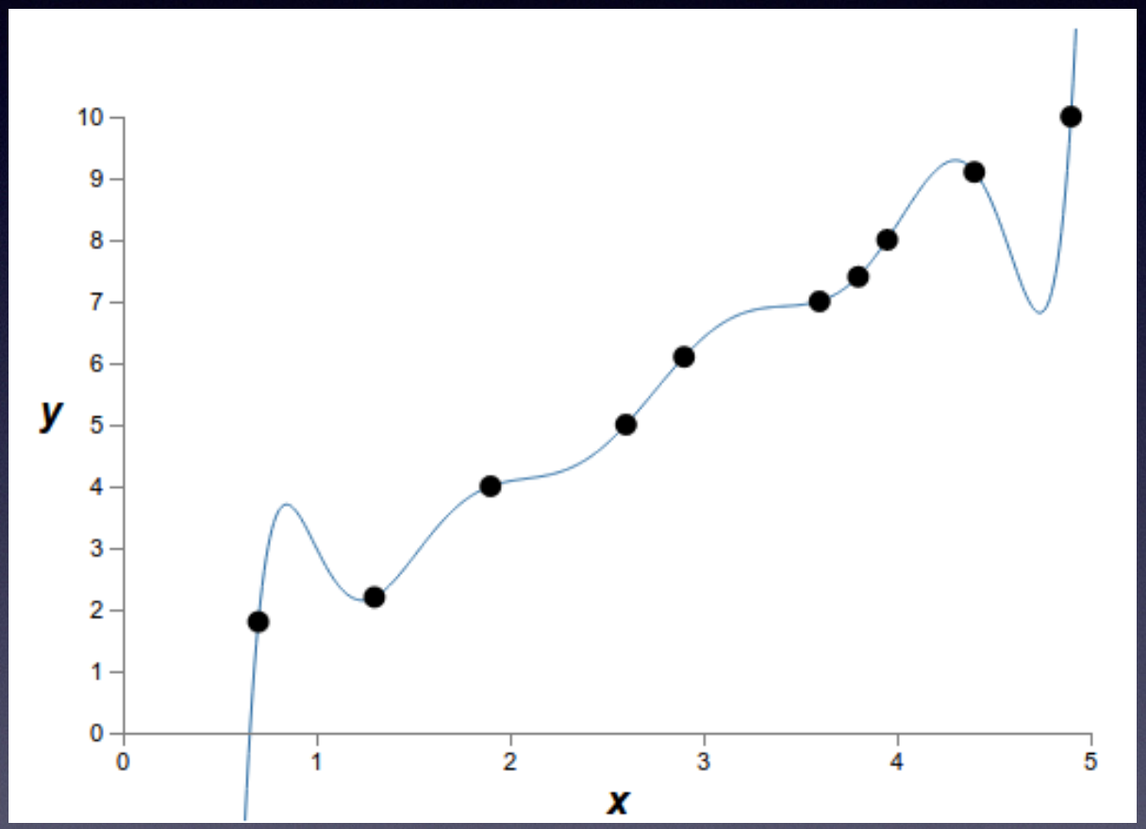
- x1 = 0, x2 = 0, output = 0
- x1 = 1, x2 = 0, output = 0
- x1 = 0, x2 = 1, output = 0
- x1 = 1, x2 = 1, output = 1

NAND gate

- x1 = 0, x2 = 0, output = 0
- x1 = 1, x2 = 0, output = 1
- x1 = 0, x2 = 1, output = 1
- x1 = 1, x2 = 1, output = 0

http://neuralnetworksanddeeplearning.com/

Generalization of Deep Learning



Overfit

- How to solve

① More complete data
② Change data already have
③ Split data to train/test/validation
④ Regularization of cost function
⑤ Dropout
⑥ Train several networks and Voting
⑦ CNN and Other Networks

http://neuralnetworksanddeeplearning.com/

Text Classification
with
Low Level API of TensorFlow

Binary-Classification with CNN (Movie Review)

Positive Examples:
if you sometimes like to go to the movies to have fun , wasabi is a good place to start .
emerges as something rare , an issue movie that's so honest and keenly observed that it doesn't feel like one .
Negative Examples:
not so much farcical as sour .
unfortunately the story and the actors are served with a hack script .

```
In [6]: # Load data
        print("Loading data...")
        x_text, y = data_helpers.load_data_and_labels(FLAGS.positive_data_file, FLAGS.negative_data_file)

        Loading data...

In [7]: x_text

Out[7]: ["the rock is destined to be the 21st century 's new conan and that he 's going to make a splash even greater than ar
        nold schwarzenegger , jean claud van damme or steven segal",
         "the gorgeously elaborate continuation of the lord of the rings trilogy is so huge that a column of words cannot ade
        quately describe co writer director peter jackson 's expanded vision of j r r tolkien 's middle earth",
         'effective but too tepid biopic',
         'if you sometimes like to go to the movies to have fun , wasabi is a good place to start',
         "emerges as something rare , an issue movie that 's so honest and keenly observed that it does n't feel like one",
         'the film provides some great insight into the neurotic mindset of all comics even those who have reached the absolu
        te top of the game',
         'offers that rare combination of entertainment and education',
         'perhaps no picture ever made has more literally showed that the road to hell is paved with good intentions',
         "steers turns in a snappy screenplay that curls at the edges it 's so clever you want to hate it but he somehow pull
        s it off",
         'take care of my cat offers a refreshingly different slice of asian cinema',
         'this is a film well worth seeing , talking and singing heads and all',
         'what really surprises about wisegirls is its low key quality and genuine tenderness',
         '\\( wendigo is \\) why we go to the cinema to be fed through the eye , the heart , the mind',
         'one of the greatest family oriented , fantasy adventure movies ever',
         'ultimately , it ponders the reasons we need stories so much',

In [8]: y

Out[8]: array([[0, 1],
               [0, 1],
               [0, 1],
               ...,
               [1, 0],
               [1, 0],
               [1, 0]])
```

Loading Data

https://github.com/dennybritz/cnn-text-classification-tf

Binary-Classification with CNN (Movie Review)

In [9]:
```
# Build vocabulary
max_document_length = max([len(x.split(" ")) for x in x_text])
vocab_processor = learn.preprocessing.VocabularyProcessor(max_document_length)
x = np.array(list(vocab_processor.fit_transform(x_text)))
```

In [10]:
```
max_document_length
```

Out[10]:
```
56
```

In [11]:
```
vocab_processor.vocabulary_
```

Out[11]:
```
<tensorflow.contrib.learn.python.learn.preprocessing.categorical_vocabulary.CategoricalVocabulary at 0x1150db450>
```

In [12]:
```
x
```

Out[12]:
```
array([[    1,     2,     3, ...,     0,     0,     0],
       [    1,    31,    32, ...,     0,     0,     0],
       [   57,    58,    59, ...,     0,     0,     0],
       ...,
       [   75,    84,  1949, ...,     0,     0,     0],
       [    1,  2191,  2690, ...,     0,     0,     0],
       [11512,     3,   147, ...,     0,     0,     0]])
```

In [13]:
```
# Randomly shuffle data
np.random.seed(10)
shuffle_indices = np.random.permutation(np.arange(len(y)))
x_shuffled = x[shuffle_indices]
y_shuffled = y[shuffle_indices]
```

In [14]:
```
shuffle_indices
```

Out[14]:
```
array([ 7359,  5573, 10180, ...,  1344,  7293,  1289])
```

In [15]:
```
x_shuffled
```

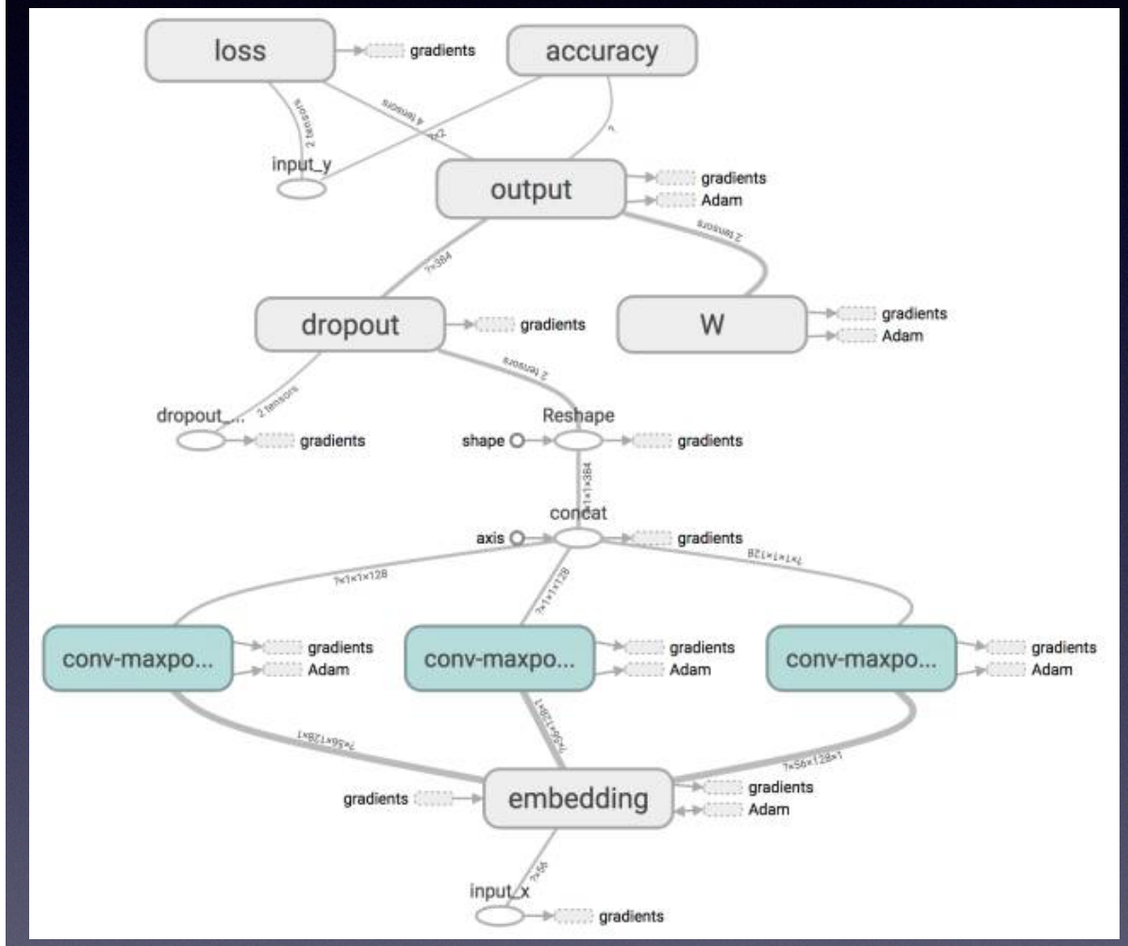
Out[15]:
```
array([[4719,   59,  182, ...,    0,    0,    0],
       [ 129, 7044,  284, ...,    0,    0,    0],
       [ 146,    3,  453, ...,    0,    0,    0],
       ...,
       [  84,    9,   17, ...,    0,    0,    0],
       [2519, 1532,    9, ...,    0,    0,    0],
```

Formating Data

https://github.com/dennybritz/cnn-text-classification-tf

Binary-Classification with CNN (Movie Review)

```
# Embedding layer
with tf.device('/cpu:0'), tf.name_scope("embedding"):
    self.W = tf.Variable(
        tf.random_uniform([vocab_size, embedding_size], -1.0, 1.0),
        name="W")
    self.embedded_chars = tf.nn.embedding_lookup(self.W, self.input_x)
    self.embedded_chars_expanded = tf.expand_dims(self.embedded_chars, -1)

# Create a convolution + maxpool layer for each filter size
pooled_outputs = []
for i, filter_size in enumerate(filter_sizes):
    with tf.name_scope("conv-maxpool-%s" % filter_size):
        # Convolution Layer
        filter_shape = [filter_size, embedding_size, 1, num_filters]
        W = tf.Variable(tf.truncated_normal(filter_shape, stddev=0.1), name="W")
        b = tf.Variable(tf.constant(0.1, shape=[num_filters]), name="b")
        conv = tf.nn.conv2d(
            self.embedded_chars_expanded,
            W,
            strides=[1, 1, 1, 1],
            padding="VALID",
            name="conv")
        # Apply nonlinearity
        h = tf.nn.relu(tf.nn.bias_add(conv, b), name="relu")
        # Maxpooling over the outputs
        pooled = tf.nn.max_pool(
            h,
            ksize=[1, sequence_length - filter_size + 1, 1, 1],
            strides=[1, 1, 1, 1],
            padding='VALID',
            name="pool")
        pooled_outputs.append(pooled)

# Combine all the pooled features
num_filters_total = num_filters * len(filter_sizes)
self.h_pool = tf.concat(pooled_outputs, 3)
self.h_pool_flat = tf.reshape(self.h_pool, [-1, num_filters_total])

# Add dropout
with tf.name_scope("dropout"):
    self.h_drop = tf.nn.dropout(self.h_pool_flat, self.dropout_keep_prob)
```



Designing Model

https://github.com/dennybritz/cnn-text-classification-tf

Binary-Classification with CNN (Movie Review)

In [4]:

```
# Training
# ==================================================

with tf.Graph().as_default():
    session_conf = tf.ConfigProto(
      allow_soft_placement=FLAGS.allow_soft_placement,
      log_device_placement=FLAGS.log_device_placement)
    sess = tf.Session(config=session_conf)
    with sess.as_default():
        cnn = TextCNN(
            sequence_length=x_train.shape[1],
            num_classes=y_train.shape[1],
            vocab_size=len(vocab_processor.vocabulary_),
            embedding_size=FLAGS.embedding_dim,
            filter_sizes=list(map(int, FLAGS.filter_sizes.split(","))),
            num_filters=FLAGS.num_filters,
            l2_reg_lambda=FLAGS.l2_reg_lambda)

        # Define Training procedure
        global_step = tf.Variable(0, name="global_step", trainable=False)
        optimizer = tf.train.AdamOptimizer(1e-3)
        grads_and_vars = optimizer.compute_gradients(cnn.loss)
        train_op = optimizer.apply_gradients(grads_and_vars, global_step=global_step)
```

```
        # Initialize all variables
        sess.run(tf.global_variables_initializer())

        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,
            train_summary_writer.add_summary(summaries, step)
```

```
        # Generate batches
        batches = data_helpers.batch_iter(
            list(zip(x_train, y_train)), FLAGS.batch_size, FLAGS.num_epochs)
        # Training loop. For each batch...
        for batch in batches:
            x_batch, y_batch = zip(*batch)
            train_step(x_batch, y_batch)
            current_step = tf.train.global_step(sess, global_step)
            if current_step % FLAGS.evaluate_every == 0:
                print("\nEvaluation:")
                dev_step(x_dev, y_dev, writer=dev_summary_writer)
                print("")
            if current_step % FLAGS.checkpoint_every == 0:
                path = saver.save(sess, checkpoint_prefix, global_step=current_step)
                print("Saved model checkpoint to {}\n".format(path))
```

Training Model

```
Total number of test examples: 10662
Accuracy: 0.972332
Saving evaluation to ./runs/149888504
```

https://github.com/dennybritz/cnn-text-classification-tf

Multi-Classification with CNN(Consumer Finance Complains)

待补充

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

Binary-Classication with CNN (Chinese Book Review)

好评例子：

本人是一名大一学生，大一的生活一直处于浑浑噩噩的状态，直到我看到了这本书。它对于我的意义远远大于一本书。《杜拉拉升职记》让我开始重新审视自己的生活，开始规划自己的未来，在今后，我希望我能向拉拉一样所向无敌。人际关系的交际技巧，与上司下属的巧妙沟通，善于利用学习机会，每一次的金玉良言，我想说，杜拉拉不仅仅给在职的白领以启迪，她势必将改变我的生活。真的谢谢作者。

当时是因为要写读书报告 所以上网到处抄 就发现这本书感觉以前自己没看过中文版的骆驼祥子就看过电影所以决定买个英文的看看本来以为 是中文翻译过来的 不会太地道结果 令我惊讶翻译的非常好！忠实了原著！很多很多复杂的复合句 而且 用词也很得体觉得 是个 非常不错的拓宽知识面的一本小说推荐！

差评例子：

如果你在外资公司工作过，尤其你在外资公司担任人力资源的话，就会觉得作者写的东西不是新野，拉拉口中讲的大道理，都是书中有的。作者对人物的描写也太肤浅，真的很失望。

小熊宝宝我觉得孩子不喜欢，能换别的吗

质量还不错，内容只适合边上厕所便消遣了

相对原模型变化不大，只是增加了一个Highway层

定义Highway层

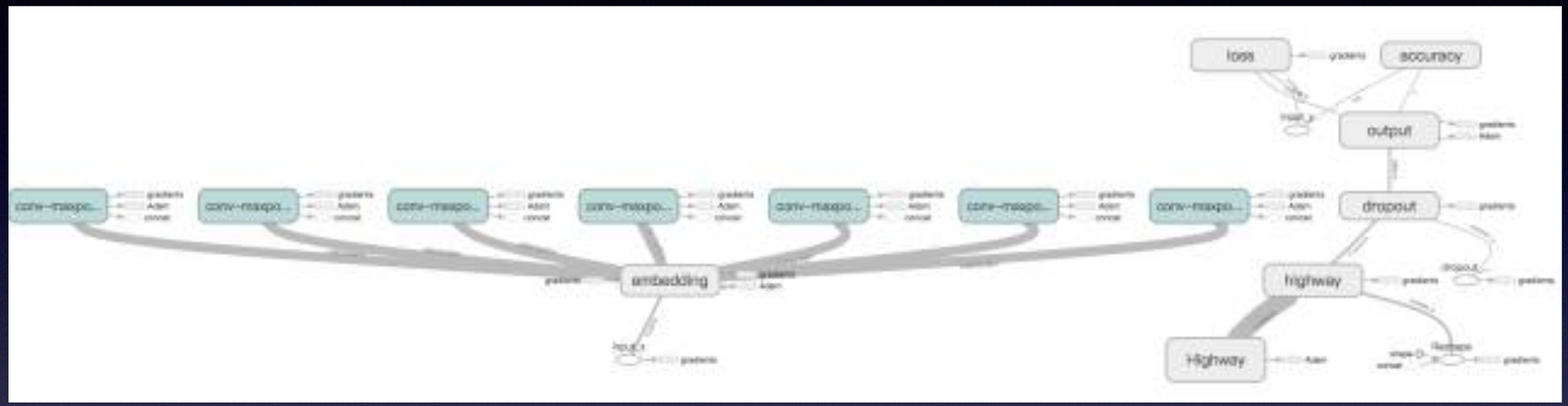
```
def highway(input_, size, num_layers=1, bias=-2.0, f=tf.nn.relu, scope='Highway'):
    """Highway Network (cf. http://arxiv.org/abs/1505.00387).
    t = sigmoid(Wy + b)
    z = t * g(Wy + b) + (1 - t) * y
    where g is nonlinearity, t is transform gate, and (1 - t) is carry gate.
    """

    with tf.variable_scope(scope):
        for idx in range(num_layers):
            g = f(linear(input_, size, scope='highway_lin_%d' % idx))

            t = tf.sigmoid(linear(input_, size, scope='highway_gate_%d' % idx) + bias)

            output = t * g + (1. - t) * input_
            input_ = output

    return output
```

将Highway层添加到模型中

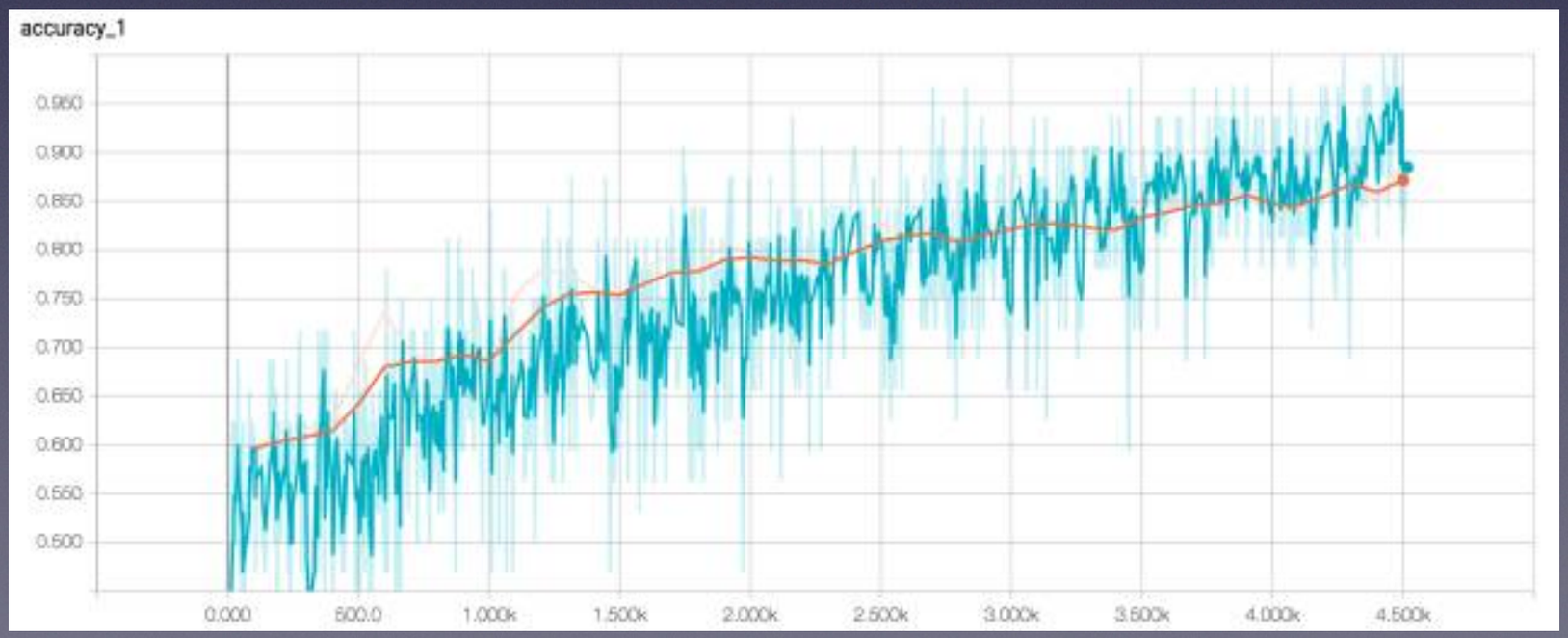
```
# Add highway
with tf.name_scope("highway"):
    self.h_highway = highway(self.h_pool_flat, self.h_pool_flat.get_shape()[1], 1, 0)
```

Binary-Classication with CNN (Chinese Book Review)

模型结构



训练12小时后，准确率达到了89%



Text Classification with Keras High Level API of TensorFlow

Multi-Classification with CNN

```
print('Build model...')
model = Sequential()

# we start off with an efficient embedding layer which maps
# our vocab indices into embedding_dims dimensions
model.add(Embedding(max_features,
                    embedding_dims,
                    input_length=maxlen))
model.add(Dropout(0.2))

# we add a Convolution1D, which will learn filters
# word group filters of size filter_length:
model.add(Conv1D(filters,
                 kernel_size,
                 padding='valid',
                 activation='relu',
                 strides=1))
# we use max pooling:
model.add(GlobalMaxPooling1D())

# We add a vanilla hidden layer:
model.add(Dense(hidden_dims))
model.add(Dropout(0.2))
model.add(Activation('relu'))

# We project onto a single unit output layer, and squash it with a sigmoid:
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

```
Build model...
```

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              
=================================================================
embedding_1 (Embedding)      (None, 400, 50)           
_________________________________________________________________
dropout_1 (Dropout)          (None, 400, 50)           
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 398, 250)          
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 250)               
_________________________________________________________________
dense_1 (Dense)              (None, 250)               
_________________________________________________________________
dropout_2 (Dropout)          (None, 250)               
_________________________________________________________________
activation_1 (Activation)    (None, 250)               
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 
_________________________________________________________________
activation_2 (Activation)    (None, 1)                 
=================================================================
Total params: 350,751
Trainable params: 350,751
Non-trainable params: 0
_________________________________________________________________
```

Designing Model

https://github.com/fchollet/keras/tree/master/examples

Multi-Classification with CNN

```
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(x_test, y_test))
```

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/2
25000/25000 [==============================] - 260s 10ms/step - loss: 0.4159 - acc: 0.7885 - val_loss: 0.2904 - val_a
cc: 0.8746
Epoch 2/2
25000/25000 [==============================] - 263s 11ms/step - loss: 0.2313 - acc: 0.9066 - val_loss: 0.2595 - val_a
cc: 0.8918

<keras.callbacks.History at 0x109b61e10>
```

```
score, acc = model.evaluate(x_test, y_test,
                            batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

```
25000/25000 [==============================] - 27s 1ms/step
Test score: 0.259518016238
Test accuracy: 0.8918
```

Training Model and Evaluating

https://github.com/fchollet/keras/tree/master/examples

Multi-Classification with LSTM(RNN)

```
print('Build model...')
model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
```

```
Build model...
```

```
# try using different optimizers and different optimizer configs
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 128)         2560000
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               131584
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 2,691,713
Trainable params: 2,691,713
Non-trainable params: 0
_________________________________________________________________
```

Designing Model

https://github.com/fchollet/keras/tree/master/examples

Multi-Classification with LSTM(RNN)

```
print('Train...')
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=1,
          validation_data=(x_test, y_test))
```

```
Train...
Train on 25000 samples, validate on 25000 samples
Epoch 1/1
25000/25000 [==============================] - 119s 5ms/step - loss: 0.2898 - acc: 0.8818 - val_loss: 0.4090 - val_ac
c: 0.8338

<keras.callbacks.History at 0x116a89e90>
```

```
score, acc = model.evaluate(x_test, y_test,
                            batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

```
25000/25000 [==============================] - 21s 850us/step
Test score: 0.408991140723
Test accuracy: 0.83376
```

Training Model and Evaluating

https://github.com/fchollet/keras/tree/master/examples

deepinthinking

- Deeplearning
- TensorFlow
- How to learn

Thank You

TensorFlow实践

- TensorFlow学习篇（一）--利用现有model进行图像识别
- TensorFlow学习篇（二）--利用CNN处理MNIST图像
- TensorFlow学习篇（三）--使用CNN对电影评论文本进行分类
- TensorFlow学习篇（四）--使用CNN对消费者财务投诉进行多分类
- TensorFlow学习篇（五）--使用CNN对中文书评进行好评差评分类