In [2]:

```
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Embedding, Dense, Conv1D, GlobalMaxPooling1D, Concatenate,

class TextCNN(object):
    def __init__(self, maxlen, max_features, embedding_dims,
                 class_num=14,
                 last_activation='softmax'):
        self.maxlen = maxlen   # 文本词的最大长度
        self.max_features = max_features #最大特征数
        self.embedding_dims = embedding_dims #词向量的大小
        self.class_num = class_num #类别数
        self.last_activation = last_activation #最近的激活函数

    def get_model(self):
        input = Input((self.maxlen,))
        embedding = Embedding(self.max_features, self.embedding_dims, input_length=self.maxlen)
        convs = []
        for kernel_size in [3, 4, 5]:
            c = Conv1D(128, kernel_size, activation='relu')(embedding)
            c = GlobalMaxPooling1D()(c)
            convs.append(c)
        x = Concatenate()(convs)

        output = Dense(self.class_num, activation=self.last_activation)(x)
        model = Model(inputs=input, outputs=output)
        return model
```

In [3]:

```
from tensorflow.keras.preprocessing import sequence
import random
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.utils import to_categorical
from utils import *
from keras.utils.np_utils import *

# 路径等配置
data_dir = './data/train_set.csv'
vocab_file = "./vocab/vocab.txt"
# vocab_size = 40000
vocab_size = 5000
# 神经网络配置
# max_features = 40001
max_features = 5001
maxlen = 100
batch_size = 64
embedding_dims = 50
epochs = 20

print('数据预处理与加载数据...')
# 如果不存在词汇表，重建
if not os.path.exists(vocab_file):
    build_vocab(data_dir, vocab_file, vocab_size)

```

```
Using TensorFlow backend.
```

```
数据预处理与加载数据...
```

In [4]:

```
os.path.exists(vocab_file)
```

Out[4]:

```
True
```

In [5]:

```
# 获得 词汇/类别 与id映射字典
categories, cat_to_id = read_category() #   cat_to_id:字典
'''['科技','股票','体育','娱乐','时政','社会','教育','财经','家居','游戏','房产','时尚','彩票'
words, word_to_id = read_vocab(vocab_file)#  词汇整成字典格式

# 全部数据
x, y = read_csv_file(data_dir)
data = list(zip(x,y))
del x,y
# 乱序
random.shuffle(data)
# 切分训练集和测试集
train_data, test_data = train_test_split(data)
# 对文本的词id和类别id进行编码
x_train = encode_sentences([content[0] for content in train_data], word_to_id)

y_train = to_categorical([content[1] for content in train_data])
```

In [6]:

```
x_train.shape,x_test.shape
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-6-f73e7a7cdf2c> in <module>
----> 1 x_train.shape,x_test.shape

AttributeError: 'list' object has no attribute 'shape'
```

In [7]:

```
x_test = encode_sentences([content[0] for content in test_data], word_to_id)
y_test = to_categorical([content[1] for content in test_data])
print('对序列做padding，保证是 samples*timestep 的维度')
x_train = sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)
```

```
对序列做padding，保证是 samples*timestep 的维度
x_train shape: (15000, 100)
x_test shape: (5000, 100)
```

In [8]:

```
print('x_train shape:', y_train.shape)
print('x_test shape:', y_test.shape)
```

```
x_train shape: (15000, 14)
x_test shape: (5000, 14)
```

In [9]:

```
y_test[0]
```

Out[9]:

```
array([0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
      dtype=float32)
```

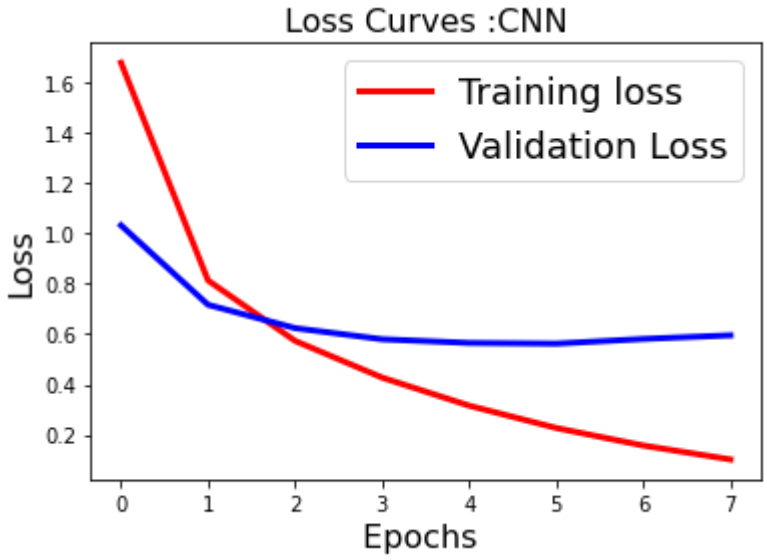
In [10]:

```
# import tensorflow as tf
# tf.config.gpu.set_per_process_memory_growth(enabled=True)


print('构建模型...')
model = TextCNN(maxlen, max_features, embedding_dims).get_model()
model.compile('adam', 'categorical_crossentropy', metrics=['accuracy'])

print('训练...')
# 设定callbacks回调函数
my_callbacks = [
    ModelCheckpoint('./cnn_model.h5', verbose=1),
    EarlyStopping(monitor='val_accuracy', patience=2, mode='max')
]

# fit拟合数据
history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          callbacks=my_callbacks,
          validation_data=(x_test, y_test))

#print('对测试集预测...')
result = model.predict(x_test)
```

```
curacy: 0.8259 - val_loss: 0.6235 - val_accuracy: 0.8046
Epoch 4/8
14976/15000 [============================>.] - ETA: 0s - loss: 0.4272 - accuracy:
0.8750
Epoch 00004: saving model to ./cnn_model.h5
15000/15000 [==============================] - 3s 196us/sample - loss: 0.4271 - ac
curacy: 0.8751 - val_loss: 0.5793 - val_accuracy: 0.8220
Epoch 5/8
14784/15000 [============================>.] - ETA: 0s - loss: 0.3157 - accuracy:
0.9123
Epoch 00005: saving model to ./cnn_model.h5
15000/15000 [==============================] - 3s 188us/sample - loss: 0.3157 - ac
curacy: 0.9124 - val_loss: 0.5647 - val_accuracy: 0.8280
Epoch 6/8
14784/15000 [============================>.] - ETA: 0s - loss: 0.2271 - accuracy:
0.9426
Epoch 00006: saving model to ./cnn_model.h5
15000/15000 [==============================] - 3s 191us/sample - loss: 0.2271 - ac
curacy: 0.9425 - val_loss: 0.5620 - val_accuracy: 0.8300
```

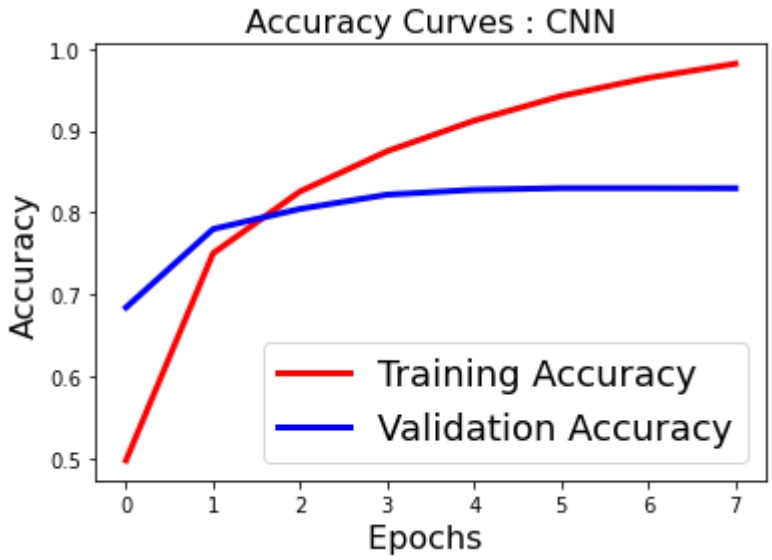
In [11]:

```
import matplotlib.pyplot as plt
plt.switch_backend('agg')
%matplotlib inline

fig1 = plt.figure()
plt.plot(history.history['loss'],'r',linewidth=3.0)
plt.plot(history.history['val_loss'],'b',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Loss',fontsize=16)
plt.title('Loss Curves :CNN',fontsize=16)
fig1.savefig('loss_cnn.png')
plt.show()
```



In [12]:

```
fig2=plt.figure()
plt.plot(history.history['accuracy'],'r',linewidth=3.0)
plt.plot(history.history['val_accuracy'],'b',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Accuracy',fontsize=16)
plt.title('Accuracy Curves : CNN',fontsize=16)
fig2.savefig('../accuracy_cnn.png')
plt.show()
```



In [ ]:

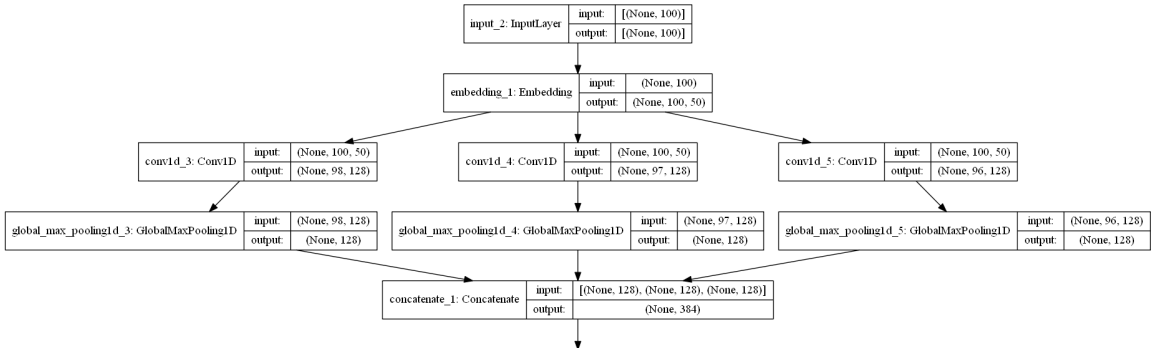
```

```

In [25]:

```
from tensorflow.keras.utils import plot_model
model.summary()
plot_model(model, show_shapes=True, show_layer_names=True)
```

```
Model: "model_1"
__________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
input_2 (InputLayer)            [(None, 100)]        0
__________________________________________________________________________________________
embedding_1 (Embedding)         (None, 100, 50)      250050      input_2[0][0]
__________________________________________________________________________________________
conv1d_3 (Conv1D)               (None, 98, 128)      19328       embedding_1[0][0]
__________________________________________________________________________________________
conv1d_4 (Conv1D)               (None, 97, 128)      25728       embedding_1[0][0]
__________________________________________________________________________________________
conv1d_5 (Conv1D)               (None, 96, 128)      32128       embedding_1[0][0]
__________________________________________________________________________________________
global_max_pooling1d_3 (GlobalM (None, 128)          0           conv1d_3[0][0]
__________________________________________________________________________________________
global_max_pooling1d_4 (GlobalM (None, 128)          0           conv1d_4[0][0]
__________________________________________________________________________________________
global_max_pooling1d_5 (GlobalM (None, 128)          0           conv1d_5[0][0]
__________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 384)          0           global_max_pooling1
d_3[0][0]
                                                                 global_max_pooling1
d_4[0][0]
                                                                 global_max_pooling1
d_5[0][0]
__________________________________________________________________________________________
dense_1 (Dense)                 (None, 14)           5390        concatenate_1[0][0]
==========================================================================================
Total params: 332,624
Trainable params: 332,624
Non-trainable params: 0
__________________________________________________________________________________________
```

Out[25]:


input_2: InputLayer
input: [(None, 100)]
output: [(None, 100)]
embedding_1: Embedding
input: (None, 100)
output: (None, 100, 50)
conv1d_3: Conv1D
input: (None, 100, 50)
output: (None, 98, 128)
conv1d_4: Conv1D
input: (None, 100, 50)
output: (None, 97, 128)
conv1d_5: Conv1D
input: (None, 100, 50)
output: (None, 96, 128)
global_max_pooling1d_3: GlobalMaxPooling1D
input: (None, 98, 128)
output: (None, 128)
global_max_pooling1d_4: GlobalMaxPooling1D
input: (None, 97, 128)
output: (None, 128)
global_max_pooling1d_5: GlobalMaxPooling1D
input: (None, 96, 128)
output: (None, 128)
concatenate_1: Concatenate
input: [(None, 128), (None, 128), (None, 128)]
output: (None, 384)

In [ ]:

```
1
```