Research Study - Deep Convolutional models for fake news classification

In this project, I've conducted intense state of art deep convolutional models for fake news classification. The main goal of this project to learn complementary linguistic features of fake news with computationally powerful convolutional neural networks.

In [1]:
```
## import dependencies
import pandas as pd
import numpy as np
import keras.utils
import nlp_util
from nltk.corpus import stopwords
import string
import re

from keras import backend as K

from keras.layers import Conv1D, Dense, Input, Lambda, LSTM
from keras.layers import Bidirectional, Lambda, TimeDistributed
from keras.layers.merge import concatenate
from keras.layers.embeddings import Embedding

from keras.utils import to_categorical
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.text import text_to_word_sequence
from keras.preprocessing import sequence
import _pickle as cPickle

from keras.layers import Concatenate, Input, MaxPooling1D
from keras.models import Model
from keras.preprocessing.sequence import pad_sequences # To make vectors the s
ame size.
# from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Embedding
from keras.layers import Conv1D, GlobalMaxPool1D, MaxPool1D
from keras.optimizers import SGD
from keras.utils import to_categorical
from keras.optimizers import Adam
from keras.callbacks import TensorBoard, CSVLogger, EarlyStopping
import matplotlib.pyplot as plt
```

```
Using TensorFlow backend.
```

In [2]:
```
## Load dataset
train_file=pd.read_csv('train.tsv', sep='\t', header=None, encoding='utf-8')
test_file=pd.read_csv('test.tsv', sep='\t', header=None, encoding='utf-8')
va_file=pd.read_csv('valid.tsv', sep='\t', header=None, encoding='utf-8')

column_names = ['Id', 'Label','Statement','Subject','Speaker','Speaker Job','S
tate Info','Party','BT','FC','HT','MT','PF','Context']
train_file.columns, test_file.columns, va_file.columns=column_names, column_na
mes, column_names

train_data = train_file[train_file.columns[~train_file.columns.isin(['Id','BT'
,'FC','HT','MT','PF'])]]
test_data = test_file[test_file.columns[~test_file.columns.isin(['Id','BT','F
C','HT','MT','PF'])]]
val_data = va_file[va_file.columns[~va_file.columns.isin(['Id','BT','FC','HT',
'MT','PF'])]]
```

In [3]:
```
train_data.head(3)
```

Out[3]:

	Label	Statement	Subject	Speaker	Speaker Job	State Info	Party	Context
0	false	Says the Annies List political group supports ...	abortion	dwayne-bohac	State representative	Texas	republican	a mailer
1	half-true	When did the decline of coal start? It started...	energy,history,job-accomplishments	scott-surovell	State delegate	Virginia	democrat	a floor speech.
2	mostly-true	Hillary Clinton agrees with John McCain "by vo...	foreign-policy	barack-obama	President	Illinois	democrat	Denver

In [4]:
```
multi_labels_dict = {'false':0, 'true':1,'pants-fire':2,'barely-true':3,'half-true':4,'mostly-true':5}
binary_labels = {'false':1, 'true':-1,'pants-fire':1,'barely-true':1,'half-true':0,'mostly-true':-1}


def one_hot_label(label):
    return to_categorical(multi_labels_dict[x], num_classes=6)

train_data['multi_label']=train_data['Label'].apply(lambda x: multi_labels_dict[x])
test_data['multi_label']=test_data['Label'].apply(lambda x: multi_labels_dict[x])
val_data['multi_label']=val_data['Label'].apply(lambda x: multi_labels_dict[x])
```

```
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  # Remove the CWD from sys.path while we load stuff.
```

Metadata processing

In [5]:
```
speakers= ['barack-obama', 'donald-trump', 'hillary-clinton', 'mitt-romney',
                'scott-walker', 'john-mccain', 'rick-perry', 'chain-email',
                'marco-rubio', 'rick-scott', 'ted-cruz', 'bernie-s', 'chris-ch
ristie',
                'facebook-posts', 'charlie-crist', 'newt-gingrich', 'jeb-bush'
,
                'joe-biden', 'blog-posting','paul-ryan']

speaker_dict={}
for cnt, speaker in enumerate(speakers):
    speaker_dict[speaker]=cnt
print(speaker_dict)

def speaker_projection(speaker):
    if isinstance(speaker, str):
        speaker=speaker.lower()
        matched=[s for s in speakers if s in speaker]
        if len(matched)>0:
            return speaker_dict[matched[0]]
        else:
            return len(speakers)

##
train_data['speaker_id']=train_data['Speaker'].apply(speaker_projection)
test_data['speaker_id']=test_data['Speaker'].apply(speaker_projection)
val_data['speaker_id']=val_data['Speaker'].apply(speaker_projection)
```

```
{'barack-obama': 0, 'donald-trump': 1, 'hillary-clinton': 2, 'mitt-romney':
3, 'scott-walker': 4, 'john-mccain': 5, 'rick-perry': 6, 'chain-email': 7, 'm
arco-rubio': 8, 'rick-scott': 9, 'ted-cruz': 10, 'bernie-s': 11, 'chris-chris
tie': 12, 'facebook-posts': 13, 'charlie-crist': 14, 'newt-gingrich': 15, 'je
b-bush': 16, 'joe-biden': 17, 'blog-posting': 18, 'paul-ryan': 19}

C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
```

In [6]:
```
## Map job
job_list = ['president', 'u.s. senator', 'governor', 'president-elect', 'presi
dential candidate',
               'u.s. representative', 'state senator', 'attorney', 'state rep
resentative', 'congress', 'others']


job_dict = {'president':0, 'u.s. senator':1, 'governor':2, 'president-elect':3
, 'presidential candidate':4,
            'u.s. representative':5, 'state senator':6, 'attorney':7, 'state r
epresentative':8, 'congress':9, 'others':10}

## Map job

def job_projection(job):
    if isinstance(job, str):
        job=job.lower()
        matched_job=[j for j in job_list if j in job]
        if len(matched_job)>0:
            return job_dict[matched_job[0]]
        else:
            return 10
    else:
        return 10

## job projection output

train_data['job_id']=train_data['Speaker Job'].apply(job_projection)
test_data['job_id']=test_data['Speaker Job'].apply(job_projection)
val_data['job_id']=val_data['Speaker Job'].apply(job_projection)
```

```
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:26: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
```

In [7]:
```
### Map political parties
party_dict={'republican':0, 'democrat':1, 'none':2, 'organization':3, 'newsmak
er':4, 'rest':5}

def map_political_party(party):
    if party in party_dict:
        return party_dict[party]
    else:
        return 5

##
train_data['party_id']=train_data['Party'].apply(map_political_party)
test_data['party_id']=test_data['Party'].apply(map_political_party)
val_data['party_id']=val_data['Party'].apply(map_political_party)
```

```
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  # This is added back by InteractiveShellApp.init_path()
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  if sys.path[0] == '':
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  del sys.path[0]
```

In [8]:
```
## Map states
all_states = ['Alabama','Alaska','Arizona','Arkansas','California','Colorado',
              'Connecticut','Delaware','Florida','Georgia','Hawaii','Idaho',
              'Illinois','Indiana','Iowa','Kansas','Kentucky','Louisiana',
              'Maine' 'Maryland','Massachusetts','Michigan','Minnesota',
              'Mississippi', 'Missouri','Montana','Nebraska','Nevada',
              'New Hampshire','New Jersey','New Mexico','New York',
              'North Carolina','North Dakota','Ohio',
              'Oklahoma','Oregon','Pennsylvania','Rhode Island',
              'South  Carolina','South Dakota','Tennessee','Texas','Utah',
              'Vermont','Virginia','Washington','West Virginia',
              'Wisconsin','Wyoming']


states_dict = {'wyoming': 48, 'colorado': 5, 'washington': 45, 'hawaii': 10,
               'tennessee': 40, 'wisconsin': 47, 'nevada': 26, 'north dakota':
32,
               'mississippi': 22, 'south dakota': 39, 'new jersey': 28, 'oklah
oma': 34,
               'delaware': 7, 'minnesota': 21, 'north carolina': 31, 'illinoi
s': 12,
               'new york': 30, 'arkansas': 3, 'west virginia': 46, 'indiana':
13,
               'louisiana': 17, 'idaho': 11, 'south  carolina': 38, 'arizona':
2,
               'iowa': 14, 'mainemaryland': 18, 'michigan': 20, 'kansas': 15,
               'utah': 42, 'virginia': 44, 'oregon': 35, 'connecticut': 6, 'mo
ntana': 24,
               'california': 4, 'massachusetts': 19, 'rhode island': 37, 'verm
ont': 43,
               'georgia': 9, 'pennsylvania': 36, 'florida': 8, 'alaska': 1, 'k
entucky': 16,
               'nebraska': 25, 'new hampshire': 27, 'texas': 41, 'missouri': 2
3, 'ohio': 33,
               'alabama': 0, 'new mexico': 29, 'rest':50}


def state_projection(state):
    if isinstance(state, str):
        state=state.lower()
        if state in states_dict:
            return states_dict[state]
        else:
            if 'washington' in state:
                return states_dict['washington']
            else:
                return 50
    else:
        return 50

## state mapping output:
train_data['state_id']=train_data['State Info'].apply(state_projection)
test_data['state_id']=test_data['State Info'].apply(state_projection)
val_data['state_id']=val_data['State Info'].apply(state_projection)
```

```
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:43: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:44: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykerne
l_launcher.py:45: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
```

In [9]:

```
## map subject
subject_list = ['health','tax','immigration','election','education',
    'candidates-biography','economy','gun','jobs','federal-budget','energy','a
bortion','foreign-policy']

subject_dict = {'health':0,'tax':1,'immigration':2,'election':3,'education':4,
                'candidates-biography':5,'economy':6,'gun':7,'jobs':8,'federal
-budget':9,
                'energy':10,'abortion':11,'foreign-policy':12, 'others':13}

## mapping subject
def subject_projection(subject):
    if isinstance(subject, str):
        subject=subject.lower()
        matched_subject=[subj for subj in subject_list if subj in subject]

        if len(matched_subject)>0:
            return subject_dict[matched_subject[0]]
        else:
            return 13
    else:
        return 13

##
train_data['subject_id']=train_data['Subject'].apply(subject_projection)
test_data['subject_id']=test_data['Subject'].apply(subject_projection)
val_data['subject_id']=val_data['Subject'].apply(subject_projection)
```

In [10]:
```
## Context mapping
Context_list = ['news release','interview','tv','radio',
                'campaign','news conference','press conference','press releas
e',
                'tweet','facebook','email']

Context_dict = {'news release':0,'interview':1,'tv':2,'radio':3,
                'campaign':4,'news conference':5,'press conference':6,'press r
elease':7,
                'tweet':8,'facebook':9,'email':10, 'others':11}

def Context_projection(context):
    if isinstance(context, str):
        context=context.lower()
        matched_context=[cntx for cntx in Context_list if cntx in context]
        if len(matched_context)>0:
            return Context_dict[matched_context[0]]
        else:
            return 11
    else:
        return 11

## context projection output
train_data['context_id']=train_data['Context'].apply(Context_projection)
test_data['context_id']=test_data['Context'].apply(Context_projection)
val_data['context_id']=val_data['Context'].apply(Context_projection)
```

In [11]:
```
### tokenize fake news statement and build vocabulary
vocab_dict={}

tokenizer = Tokenizer(num_words=20000)
tokenizer.fit_on_texts(train_data['Statement'])
vocab_dict=tokenizer.word_index
cPickle.dump(tokenizer.word_index, open("vocab.p","wb"))
print("vocab dictionary is created")
print("saved vocan dictionary to pickle file")
```

```
vocab dictionary is created
saved vocan dictionary to pickle file
```

In [12]:
```
## data preprocessing

# def preprocessing_txt(dataset):
#     stop_words = set(stopwords.words('english'))
#     corpus=[]
#     for elm in range(0, len(dataset.index)):
#         res=' '.join([i for i in dataset['Statement'][elm].lower().split() i
f i not in stop_words])
#         res=re.sub("</?.*?>"," <> ",dataset['Statement'][elm])    # remove t
ags
#         res=re.sub("(\\d|\\W)+"," ",dataset['Statement'][elm])        # remo
ve special characte
#         res=re.sub(r'@([A-Za-z0-9_]+)', "",dataset['Statement'][elm])  # rem
ove twitter handler
#         res=re.sub('(\r)+', "", dataset['Statement'][elm])            # remo
ve newline character
#         res=re.sub('[^\x00-\x7F]+', "", dataset['Statement'][elm])    # remo
ve non-ascii characters
#         res=''.join(x for x in dataset['Statement'][elm] if x not in set(str
ing.punctuation))   ## remove punctuation
#         corpus.append(res)
#     return corpus
```

In [13]:
```
### TODO: END
```

In [14]:
```
vocab_length = len(vocab_dict.keys())
hidden_size = 150 #Has to be same as EMBEDDING_DIM
lstm_size = 100
num_steps = 32
num_epochs = 30
batch_size = 64
#Hyperparams for CNN
kernel_sizes = [3,4,5]
filter_size = 128
#Meta data related hyper params
num_party = 6
num_state = 51
num_context = 12
num_job = 11
num_sub = 14
num_speaker = 21
embedding_dims=300
max_features = len(tokenizer.word_index)+1
```

In [15]:
```
vocab_length = len(vocab_dict.keys())
vocab_length
```

Out[15]: 12408

In [16]:
```
### create embedding layer
num_words=len(vocab_dict)+1

def loadGloveModel(gloveFile):
    print("Loading Glove Model")
    embeddings_index = {}
    f = open(gloveFile, encoding='utf8')
    for line in f:
        values = line.split()
        word = ''.join(values[:-300])
        coefs = np.asarray(values[-300:], dtype='float32')
        embeddings_index[word] = coefs
    f.close()
    return embeddings_index

glove_model = loadGloveModel('glove.6B.300d.txt')

def build_glove_embedding_layers():
    embed_matrix=np.zeros((max_features, embedding_dims))
    for word, indx in tokenizer.word_index.items():
        if indx >= max_features:
            continue
        if word in glove_model:
            embed_vec=glove_model[word]
            if embed_vec is not None:
                embed_matrix[indx]=embed_vec
    return embed_matrix

embedding_weights=build_glove_embedding_layers()
```

```
Loading Glove Model
```

In [17]:
```
### data preprocessing
def preprocessing_txt_keras(statement):
    txt=text_to_word_sequence(statement)
    val=[0]*32
    val=[vocab_dict[t] for t in txt if t in vocab_dict]   ##replace unknown wo
rds with zero index
    return val
```

In [18]:
```
## training instances list
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import string
stop = set(stopwords.words('english'))

### remove stopwords fitst
train_data['Statement'] = list(map(' '.join, train_data['Statement'].apply(lam
bda x: [item for item in x.lower().split() if item not in stop])))
test_data['Statement'] = list(map(' '.join, test_data['Statement'].apply(lambd
a x: [item for item in x.lower().split() if item not in stop])))
val_data['Statement'] = list(map(' '.join, val_data['Statement'].apply(lambda
x: [item for item in x.lower().split() if item not in stop])))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\jshayi2\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [19]:
```
train_data['word_id']=train_data['Statement'].apply(preprocessing_txt_keras)
test_data['word_id']=test_data['Statement'].apply(preprocessing_txt_keras)
val_data['word_id']=val_data['Statement'].apply(preprocessing_txt_keras)

x_train=train_data['word_id']
x_test=test_data['word_id']
x_val=val_data['word_id']

y_train=train_data['multi_label']
y_val=val_data['multi_label']

##
x_train=sequence.pad_sequences(x_train, maxlen=num_steps, padding='post', trun
cating='post')
y_train=to_categorical(y_train, num_classes=6)
x_val=sequence.pad_sequences(x_val, maxlen=num_steps, padding='post', truncati
ng='post')
y_val=to_categorical(y_val, num_classes=6)
x_test=sequence.pad_sequences(x_test, maxlen=num_steps, padding='post', trunca
ting='post')

## meta data preparation
tr_party=to_categorical(train_data['party_id'], num_classes=num_party)
tr_state=to_categorical(train_data['state_id'], num_classes=num_state)
tr_cont=to_categorical(train_data['context_id'], num_classes=num_context)
tr_job=to_categorical(train_data['job_id'], num_classes=num_job)
tr_subj=to_categorical(train_data['subject_id'], num_classes=num_sub)
# tr_speaker=to_categorical(train_data['speaker_id'], num_classes=num_speaker)

# ## put all metadata of train data together in one stack
# x_train_metadata=np.hstack(tr_party, tr_state, tr_job, tr_subj, tr_speaker,
 tr_cont)
x_train_metadata=np.hstack((tr_party, tr_state, tr_cont,tr_job, tr_subj))


# #**************************************************************************
*****#
val_party=to_categorical(val_data['party_id'], num_classes=num_party)
val_state=to_categorical(val_data['state_id'], num_classes=num_state)
val_cont=to_categorical(val_data['context_id'], num_classes=num_context)
val_job=to_categorical(val_data['job_id'], num_classes=num_job)
val_subj=to_categorical(val_data['subject_id'], num_classes=num_sub)
# val_speaker=to_categorical(val_data['speaker_id'], num_classes=num_speaker)


# ## put all metadata of train data together in one stack
# x_val_metadata=np.hstack(val_party, val_state, val_job, val_subj, val_speake
r, val_cont)
x_val_metadata=np.hstack((val_party, val_state, val_cont,val_job, val_subj, ))

# #**************************************************************************
*****#
te_party=to_categorical(test_data['party_id'], num_classes=num_party)
te_state=to_categorical(test_data['state_id'], num_classes=num_state)
te_cont=to_categorical(test_data['context_id'], num_classes=num_context)
te_job=to_categorical(test_data['job_id'], num_classes=num_job)
```

```
te_subj=to_categorical(test_data['subject_id'], num_classes=num_sub)
# te_speaker=to_categorical(test_data['speaker_id'], num_classes=num_speaker)

# ## put all metadata of train data together in one stack
x_test_metadata=np.hstack((te_party, te_state, te_cont,te_job, te_subj))
```

1.1 state of art CNN model for fake news classification

In [20]:
```
### to initialize model weight
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

In [22]:
```
kernel_arr = []
statement_input = Input(shape=(num_steps,), dtype='int32', name='main_input')
embed_sequences = Embedding(vocab_length+1,embedding_dims,weights=[embedding_w
eights],input_length=num_steps,trainable=False)(statement_input) #Preloaded gl
ove embeddings
# x = Embedding(output_dim=hidden_size, input_dim=vocab_length+1, input_length
=num_steps)(statement_input) #Train embeddings from scratch

for kernel in kernel_sizes:
    x_1 = Conv1D(filters=filter_size,kernel_size=kernel,
                 padding="valid", activation="relu", strides=1)(embed_sequence
s)
    x_1 = MaxPool1D(3)(x_1)
    x_flat = Flatten()(x_1)
    x_drop = Dropout(0.9)(x_flat)
    kernel_arr.append(x_drop)

conv_in = keras.layers.concatenate(kernel_arr)
# conv_in = Dropout(0.85)(conv_in)
conv_in = Dense(128, activation='relu')(conv_in)

#Meta input
meta_input = Input(shape=(x_train_metadata.shape[1],), name='aux_input')
x_drop = Dropout(0.7)(meta_input)
x_meta = Dense(64, activation='relu')(x_drop)
x = keras.layers.concatenate([conv_in, x_meta])

main_output = Dense(6, activation='softmax', name='main_output')(x)
model = Model(inputs=[statement_input, meta_input], outputs=[main_output])

#*************************************************************************#
adam = Adam(lr=1e-4, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.2)
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(optimizer=sgd,
              loss='categorical_crossentropy',
              metrics=['categorical_accuracy'])

model.summary()

tb = TensorBoard()
csv_logger = keras.callbacks.CSVLogger('training.log')
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=2)
filepath= "weights.best.hdf5"
checkpoint = keras.callbacks.ModelCheckpoint(filepath,
                                             monitor='val_categorical_accurac
y',
                                             verbose=1, save_best_only=True, m
ode='max')

history11 = model.fit({'main_input': x_train, 'aux_input': x_train_metadata},
                   {'main_output': y_train},epochs=20, batch_size=100,
                   validation_data=({'main_input': x_val, 'aux_input': x_val_m
etadata},{'main_output': y_val}),
                  callbacks=[tb,csv_logger,checkpoint, es])
```

```
WARNING:tensorflow:From C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\li
b\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_w
ith (from tensorflow.python.framework.ops) is deprecated and will be removed
in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\li
b\site-packages\keras\backend\tensorflow_backend.py:3445: calling dropout (fr
om tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be rem
oved in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - k
eep_prob`.
```

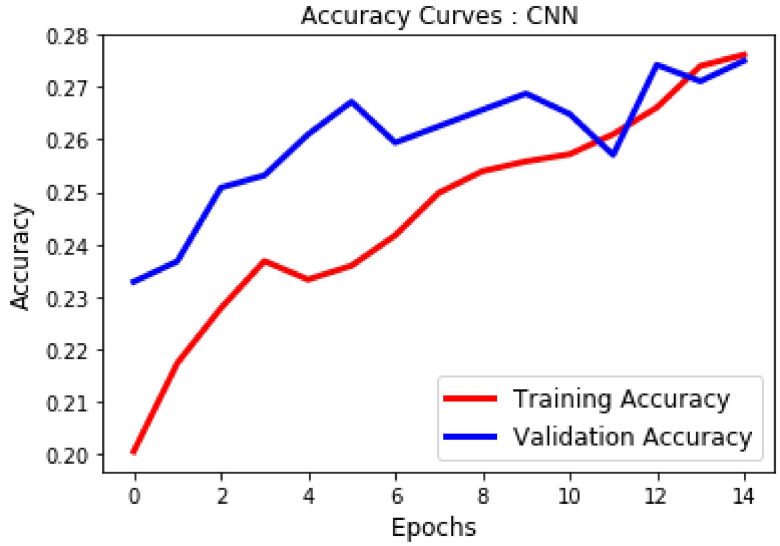
Layer (type)	Output Shape	Param #	Connected to
main_input (InputLayer)	(None, 32)	0	
embedding_1 (Embedding)	(None, 32, 300)	3722700	main_input[0][0]
conv1d_1 (Conv1D)	(None, 30, 128)	115328	embedding_1[0][0]
conv1d_2 (Conv1D)	(None, 29, 128)	153728	embedding_1[0][0]
conv1d_3 (Conv1D)	(None, 28, 128)	192128	embedding_1[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 10, 128)	0	conv1d_1[0][0]
max_pooling1d_2 (MaxPooling1D)	(None, 9, 128)	0	conv1d_2[0][0]
max_pooling1d_3 (MaxPooling1D)	(None, 9, 128)	0	conv1d_3[0][0]
flatten_1 (Flatten)	(None, 1280)	0	max_pooling1d_1[0][0]
flatten_2 (Flatten)	(None, 1152)	0	max_pooling1d_2[0][0]

```
flatten_3 (Flatten)             (None, 1152)         0           max_pooling1
d_3[0][0]
__________________________________________________________________________
_____
dropout_1 (Dropout)             (None, 1280)         0           flatten_1[0]
[0]
__________________________________________________________________________
_____
dropout_2 (Dropout)             (None, 1152)         0           flatten_2[0]
[0]
__________________________________________________________________________
_____
dropout_3 (Dropout)             (None, 1152)         0           flatten_3[0]
[0]
__________________________________________________________________________
_____
aux_input (InputLayer)          (None, 94)           0
__________________________________________________________________________
_____
concatenate_1 (Concatenate)     (None, 3584)         0           dropout_1[0]
[0]
                                                                 dropout_2[0]
[0]
                                                                 dropout_3[0]
[0]
__________________________________________________________________________
_____
dropout_4 (Dropout)             (None, 94)           0           aux_input[0]
[0]
__________________________________________________________________________
_____
dense_1 (Dense)                 (None, 128)          458880      concatenate_
1[0][0]
__________________________________________________________________________
_____
dense_2 (Dense)                 (None, 64)           6080        dropout_4[0]
[0]
__________________________________________________________________________
_____
concatenate_2 (Concatenate)     (None, 192)          0           dense_1[0]
[0]
                                                                 dense_2[0]
[0]
__________________________________________________________________________
_____
main_output (Dense)             (None, 6)            1158        concatenate_
2[0][0]
==========================================================================
==========
Total params: 4,650,002
Trainable params: 927,302
Non-trainable params: 3,722,700
__________________________________________________________________________
_____
WARNING:tensorflow:From C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\li
b\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensor
flow.python.ops.math_ops) is deprecated and will be removed in a future versi
```

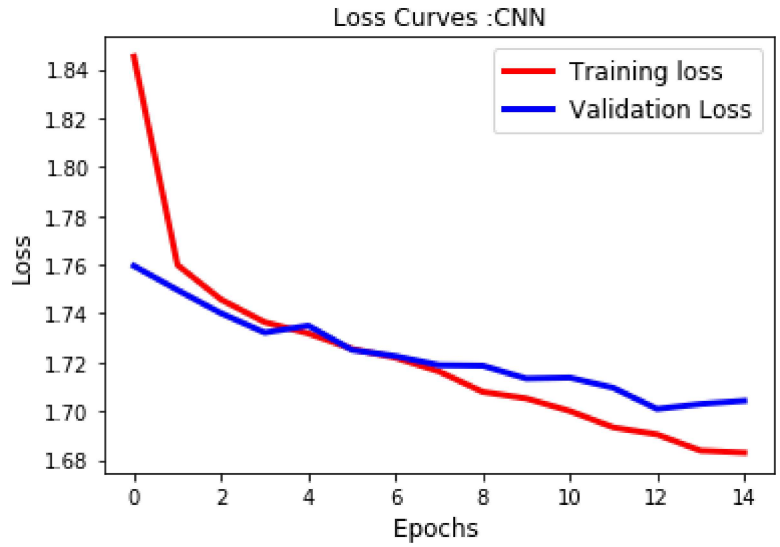
```
on.
Instructions for updating:
Use tf.cast instead.
Train on 10240 samples, validate on 1284 samples
Epoch 1/20
10240/10240 [==============================] - 13s 1ms/step - loss: 1.8452 -
categorical_accuracy: 0.2006 - val_loss: 1.7595 - val_categorical_accuracy:
0.2329

Epoch 00001: val_categorical_accuracy improved from -inf to 0.23287, saving m
odel to weights.best.hdf5
Epoch 2/20
10240/10240 [==============================] - 12s 1ms/step - loss: 1.7598 -
categorical_accuracy: 0.2174 - val_loss: 1.7496 - val_categorical_accuracy:
0.2368

Epoch 00002: val_categorical_accuracy improved from 0.23287 to 0.23676, savin
g model to weights.best.hdf5
Epoch 3/20
10240/10240 [==============================] - 13s 1ms/step - loss: 1.7457 -
categorical_accuracy: 0.2278 - val_loss: 1.7401 - val_categorical_accuracy:
0.2508

Epoch 00003: val_categorical_accuracy improved from 0.23676 to 0.25078, savin
g model to weights.best.hdf5
Epoch 4/20
10240/10240 [==============================] - 12s 1ms/step - loss: 1.7364 -
categorical_accuracy: 0.2368 - val_loss: 1.7322 - val_categorical_accuracy:
0.2531

Epoch 00004: val_categorical_accuracy improved from 0.25078 to 0.25312, savin
g model to weights.best.hdf5
Epoch 5/20
10240/10240 [==============================] - 13s 1ms/step - loss: 1.7318 -
categorical_accuracy: 0.2333 - val_loss: 1.7350 - val_categorical_accuracy:
0.2609

Epoch 00005: val_categorical_accuracy improved from 0.25312 to 0.26090, savin
g model to weights.best.hdf5
Epoch 6/20
10240/10240 [==============================] - 12s 1ms/step - loss: 1.7255 -
categorical_accuracy: 0.2359 - val_loss: 1.7251 - val_categorical_accuracy:
0.2671

Epoch 00006: val_categorical_accuracy improved from 0.26090 to 0.26713, savin
g model to weights.best.hdf5
Epoch 7/20
10240/10240 [==============================] - 13s 1ms/step - loss: 1.7219 -
categorical_accuracy: 0.2417 - val_loss: 1.7225 - val_categorical_accuracy:
0.2593

Epoch 00007: val_categorical_accuracy did not improve from 0.26713
Epoch 8/20
10240/10240 [==============================] - 12s 1ms/step - loss: 1.7163 -
categorical_accuracy: 0.2498 - val_loss: 1.7188 - val_categorical_accuracy:
0.2625
```

```
Epoch 00008: val_categorical_accuracy did not improve from 0.26713
Epoch 9/20
10240/10240 [==============================] - 13s 1ms/step - loss: 1.7079 -
categorical_accuracy: 0.2539 - val_loss: 1.7186 - val_categorical_accuracy:
0.2656

Epoch 00009: val_categorical_accuracy did not improve from 0.26713
Epoch 10/20
10240/10240 [==============================] - 12s 1ms/step - loss: 1.7052 -
categorical_accuracy: 0.2558 - val_loss: 1.7134 - val_categorical_accuracy:
0.2687

Epoch 00010: val_categorical_accuracy improved from 0.26713 to 0.26869, savin
g model to weights.best.hdf5
Epoch 11/20
10240/10240 [==============================] - 12s 1ms/step - loss: 1.7001 -
categorical_accuracy: 0.2571 - val_loss: 1.7137 - val_categorical_accuracy:
0.2648

Epoch 00011: val_categorical_accuracy did not improve from 0.26869
Epoch 12/20
10240/10240 [==============================] - 13s 1ms/step - loss: 1.6934 -
categorical_accuracy: 0.2609 - val_loss: 1.7096 - val_categorical_accuracy:
0.2570

Epoch 00012: val_categorical_accuracy did not improve from 0.26869
Epoch 13/20
10240/10240 [==============================] - 13s 1ms/step - loss: 1.6905 -
categorical_accuracy: 0.2660 - val_loss: 1.7009 - val_categorical_accuracy:
0.2741

Epoch 00013: val_categorical_accuracy improved from 0.26869 to 0.27414, savin
g model to weights.best.hdf5
Epoch 14/20
10240/10240 [==============================] - 13s 1ms/step - loss: 1.6838 -
categorical_accuracy: 0.2739 - val_loss: 1.7029 - val_categorical_accuracy:
0.2710

Epoch 00014: val_categorical_accuracy did not improve from 0.27414
Epoch 15/20
10240/10240 [==============================] - 13s 1ms/step - loss: 1.6830 -
categorical_accuracy: 0.2761 - val_loss: 1.7042 - val_categorical_accuracy:
0.2749

Epoch 00015: val_categorical_accuracy improved from 0.27414 to 0.27492, savin
g model to weights.best.hdf5
Epoch 00015: early stopping
```

In [23]:
```
accu_curve=plt.figure()
plt.plot(history11.history['categorical_accuracy'],'r',linewidth=3.0)
plt.plot(history11.history['val_categorical_accuracy'],'b',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Accuracy',fontsize=12)
plt.title('Accuracy Curves : CNN',fontsize=12)
# accu_curve.savefig('accuracy_cnn_improved_v1.6.best.png')
plt.show()
##^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##
loss_curve = plt.figure()
plt.plot(history11.history['loss'],'r',linewidth=3.0)
plt.plot(history11.history['val_loss'],'b',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Loss',fontsize=12)
plt.title('Loss Curves :CNN',fontsize=12)
# loss_curve.savefig('loss_cnn_improved_v1.6.best.png')
loss_curve.show()
```




```
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\matplotl
ib\figure.py:445: UserWarning: Matplotlib is currently using module://ipykern
el.pylab.backend_inline, which is a non-GUI backend, so cannot show the figur
e.
  % get_backend())
```



1.2 Multi-convolutional CNN model for fake news classification

Let's do small modification on above CNN model, in this notebook I introduce multi convolutional CNN model for fake news classification. let's see how it works and see where is significancy.

In [24]:
```
statement_input = Input(shape=(num_steps,), dtype='int64', name='main_input')
embed_sequence = Embedding(vocab_length+1,embedding_dims,weights=[embedding_we
ights],input_length=num_steps,trainable=False)(statement_input) #Preloaded glo
ve embeddings

## add multi convolutional layer
l_conv1 = Conv1D(128,4,activation="relu", padding="same", strides=1)(embed_seq
uence)
l_pool1 = MaxPooling1D(4)(l_conv1)
l_conv2 = Conv1D(128, 4, activation="relu")(l_pool1)
l_pool2 = MaxPooling1D(4)(l_conv2)
# l_conv3 = Conv1D(128,3,activation="relu")(l_pool2)
# l_pool3 = MaxPooling1D()(l_conv3)
l_flat = Flatten()(l_pool2)
conv_in = Dropout(0.9)(l_flat)
conv_in = Dense(128, activation='relu')(conv_in)

#Meta input
meta_input = Input(shape=(x_train_metadata.shape[1],), name='aux_input')
x_drop = Dropout(0.7)(meta_input)
x_meta = Dense(64, activation='relu')(x_drop)
x = keras.layers.concatenate([conv_in, x_meta])

main_output = Dense(6, activation='softmax', name='main_output')(x)
model_multi = Model(inputs=[statement_input, meta_input], outputs=[main_output
])

#*********************************************************************#
adam = Adam(lr=1e-4, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.2)
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model_multi.compile(optimizer=sgd,
                    loss='categorical_crossentropy',
                    metrics=['categorical_accuracy'])

model_multi.summary()

tb = TensorBoard()
csv_logger = keras.callbacks.CSVLogger('training.log')
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=2)
filepath= "weights.best.hdf5"
checkpoint = keras.callbacks.ModelCheckpoint(filepath,
                                             monitor='val_categorical_accurac
y',
                                             verbose=1, save_best_only=True, m
ode='max')

history12= model_multi.fit({'main_input': x_train, 'aux_input': x_train_metada
ta},
                           {'main_output': y_train},epochs=18, batch_size=64,
                           validation_data=({'main_input': x_val, 'aux_input':
x_val_metadata},{'main_output': y_val}),
                           callbacks=[tb,csv_logger,checkpoint, es])
```

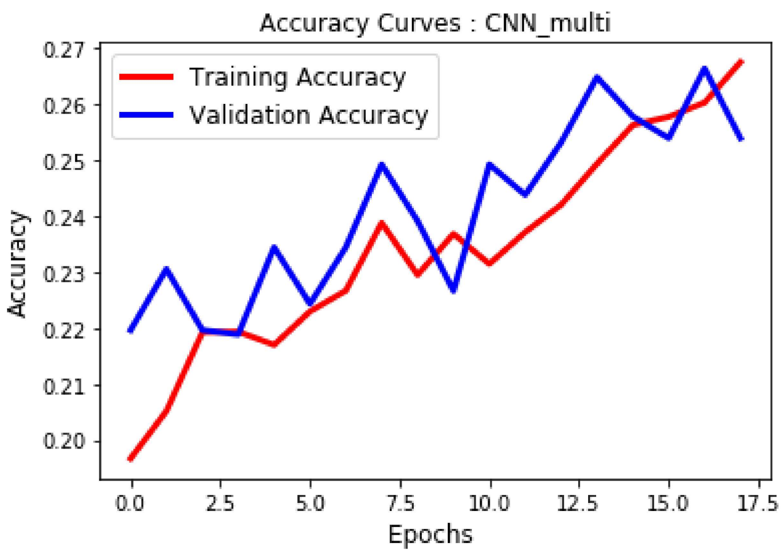
Layer (type)	Output Shape	Param #	Connected to
main_input (InputLayer)	(None, 32)	0	
embedding_2 (Embedding)	(None, 32, 300)	3722700	main_input[0][0]
conv1d_4 (Conv1D)	(None, 32, 128)	153728	embedding_2[0][0]
max_pooling1d_4 (MaxPooling1D)	(None, 8, 128)	0	conv1d_4[0][0]
conv1d_5 (Conv1D)	(None, 5, 128)	65664	max_pooling1d_4[0][0]
max_pooling1d_5 (MaxPooling1D)	(None, 1, 128)	0	conv1d_5[0][0]
flatten_4 (Flatten)	(None, 128)	0	max_pooling1d_5[0][0]
aux_input (InputLayer)	(None, 94)	0	
dropout_5 (Dropout)	(None, 128)	0	flatten_4[0][0]
dropout_6 (Dropout)	(None, 94)	0	aux_input[0][0]
dense_3 (Dense)	(None, 128)	16512	dropout_5[0][0]
dense_4 (Dense)	(None, 64)	6080	dropout_6[0][0]
concatenate_3 (Concatenate)	(None, 192)	0	dense_3[0][0] dense_4[0][0]

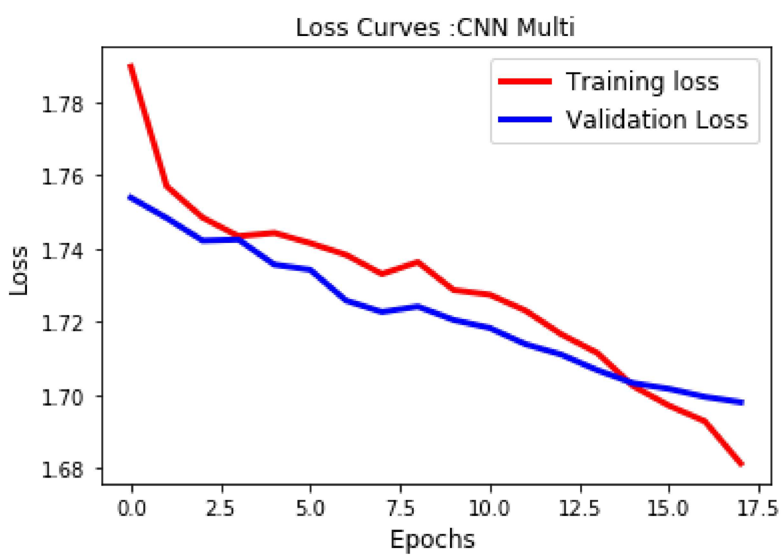
```
main_output (Dense)             (None, 6)            1158        concatenate_
3[0][0]
==========================================================================
======================
Total params: 3,965,842
Trainable params: 243,142
Non-trainable params: 3,722,700
__________________________________________________________________________
_____________________
Train on 10240 samples, validate on 1284 samples
Epoch 1/18
10240/10240 [==============================] - 6s 576us/step - loss: 1.7896 -
categorical_accuracy: 0.1967 - val_loss: 1.7539 - val_categorical_accuracy:
0.2196

Epoch 00001: val_categorical_accuracy improved from -inf to 0.21963, saving m
odel to weights.best.hdf5
Epoch 2/18
10240/10240 [==============================] - 5s 514us/step - loss: 1.7570 -
categorical_accuracy: 0.2052 - val_loss: 1.7483 - val_categorical_accuracy:
0.2305

Epoch 00002: val_categorical_accuracy improved from 0.21963 to 0.23053, savin
g model to weights.best.hdf5
Epoch 3/18
10240/10240 [==============================] - 6s 555us/step - loss: 1.7484 -
categorical_accuracy: 0.2192 - val_loss: 1.7422 - val_categorical_accuracy:
0.2196

Epoch 00003: val_categorical_accuracy did not improve from 0.23053
Epoch 4/18
10240/10240 [==============================] - 5s 515us/step - loss: 1.7434 -
categorical_accuracy: 0.2193 - val_loss: 1.7425 - val_categorical_accuracy:
0.2188

Epoch 00004: val_categorical_accuracy did not improve from 0.23053
Epoch 5/18
10240/10240 [==============================] - 5s 512us/step - loss: 1.7443 -
categorical_accuracy: 0.2170 - val_loss: 1.7356 - val_categorical_accuracy:
0.2344

Epoch 00005: val_categorical_accuracy improved from 0.23053 to 0.23442, savin
g model to weights.best.hdf5
Epoch 6/18
10240/10240 [==============================] - 5s 510us/step - loss: 1.7415 -
categorical_accuracy: 0.2229 - val_loss: 1.7342 - val_categorical_accuracy:
0.2243

Epoch 00006: val_categorical_accuracy did not improve from 0.23442
Epoch 7/18
10240/10240 [==============================] - 5s 508us/step - loss: 1.7383 -
categorical_accuracy: 0.2267 - val_loss: 1.7258 - val_categorical_accuracy:
0.2344

Epoch 00007: val_categorical_accuracy did not improve from 0.23442
Epoch 8/18
10240/10240 [==============================] - 5s 513us/step - loss: 1.7331 -
```

```
categorical_accuracy: 0.2388 - val_loss: 1.7227 - val_categorical_accuracy:
0.2492

Epoch 00008: val_categorical_accuracy improved from 0.23442 to 0.24922, savin
g model to weights.best.hdf5
Epoch 9/18
10240/10240 [==============================] - 5s 516us/step - loss: 1.7363 -
categorical_accuracy: 0.2295 - val_loss: 1.7242 - val_categorical_accuracy:
0.2391

Epoch 00009: val_categorical_accuracy did not improve from 0.24922
Epoch 10/18
10240/10240 [==============================] - 5s 512us/step - loss: 1.7287 -
categorical_accuracy: 0.2368 - val_loss: 1.7205 - val_categorical_accuracy:
0.2266

Epoch 00010: val_categorical_accuracy did not improve from 0.24922
Epoch 11/18
10240/10240 [==============================] - 5s 505us/step - loss: 1.7274 -
categorical_accuracy: 0.2314 - val_loss: 1.7184 - val_categorical_accuracy:
0.2492

Epoch 00011: val_categorical_accuracy did not improve from 0.24922
Epoch 12/18
10240/10240 [==============================] - 5s 505us/step - loss: 1.7231 -
categorical_accuracy: 0.2371 - val_loss: 1.7139 - val_categorical_accuracy:
0.2438

Epoch 00012: val_categorical_accuracy did not improve from 0.24922
Epoch 13/18
10240/10240 [==============================] - 5s 501us/step - loss: 1.7166 -
categorical_accuracy: 0.2420 - val_loss: 1.7111 - val_categorical_accuracy:
0.2531

Epoch 00013: val_categorical_accuracy improved from 0.24922 to 0.25312, savin
g model to weights.best.hdf5
Epoch 14/18
10240/10240 [==============================] - 5s 510us/step - loss: 1.7116 -
categorical_accuracy: 0.2493 - val_loss: 1.7068 - val_categorical_accuracy:
0.2648

Epoch 00014: val_categorical_accuracy improved from 0.25312 to 0.26480, savin
g model to weights.best.hdf5
Epoch 15/18
10240/10240 [==============================] - 5s 503us/step - loss: 1.7025 -
categorical_accuracy: 0.2563 - val_loss: 1.7033 - val_categorical_accuracy:
0.2578

Epoch 00015: val_categorical_accuracy did not improve from 0.26480
Epoch 16/18
10240/10240 [==============================] - 5s 513us/step - loss: 1.6972 -
categorical_accuracy: 0.2577 - val_loss: 1.7017 - val_categorical_accuracy:
0.2539

Epoch 00016: val_categorical_accuracy did not improve from 0.26480
Epoch 17/18
10240/10240 [==============================] - 5s 517us/step - loss: 1.6929 -
```

```
categorical_accuracy: 0.2603 - val_loss: 1.6996 - val_categorical_accuracy:
0.2664

Epoch 00017: val_categorical_accuracy improved from 0.26480 to 0.26636, savin
g model to weights.best.hdf5
Epoch 18/18
10240/10240 [==============================] - 5s 505us/step - loss: 1.6814 -
categorical_accuracy: 0.2675 - val_loss: 1.6981 - val_categorical_accuracy:
0.2539

Epoch 00018: val_categorical_accuracy did not improve from 0.26636
```

In [25]:
```
## to see performance difference
accu_curve=plt.figure()
plt.plot(history12.history['categorical_accuracy'],'r',linewidth=3.0)
plt.plot(history12.history['val_categorical_accuracy'],'b',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Accuracy',fontsize=12)
plt.title('Accuracy Curves : CNN_multi',fontsize=12)
# accu_curve.savefig('accuracy_multi_conv_cnn_v.1.6.png')
plt.show()
##^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##
loss_curve = plt.figure()
plt.plot(history12.history['loss'],'r',linewidth=3.0)
plt.plot(history12.history['val_loss'],'b',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Loss',fontsize=12)
plt.title('Loss Curves :CNN Multi',fontsize=12)
# loss_curve.savefig('loss_multi_conv_cnnv.1.6.png')
loss_curve.show()
```





1.3 Hybrid CNN model for fake news classification

In [26]:
```
kernel_arr = []
statement_input = Input(shape=(num_steps,), dtype='int32', name='main_input')
embed_sequences = Embedding(vocab_length+1,embedding_dims,weights=[embedding_w
eights],input_length=num_steps,trainable=False)(statement_input) #Preloaded gl
ove embeddings

for kernel in kernel_sizes:
    x_1 = Conv1D(filters=filter_size,kernel_size=kernel,
                 padding="same", activation="relu", strides=1)(embed_sequences
)
    x_1 = MaxPool1D(3)(x_1)
    x_flat = Flatten()(x_1)
    x_drop = Dropout(0.85)(x_flat)
    kernel_arr.append(x_drop)

conv_ins = keras.layers.concatenate(kernel_arr)
# conv_ins = Dropout(0.85)(conv_ins)
conv_ins = Dense(128, activation='relu')(conv_ins)

## add multi convolutional layer
l_conv1 = Conv1D(128,3,activation="relu", padding="valid", strides=1)(embed_se
quences)
l_pool1 = MaxPooling1D(3)(l_conv1)
l_conv2 = Conv1D(128, 3, activation="relu")(l_pool1)
l_pool2 = MaxPooling1D(3)(l_conv2)
# l_conv3 = Conv1D(128,3,activation="relu")(l_pool2)
# l_pool3 = MaxPool1D(3)(l_conv3)
l_flat = Flatten()(l_pool2)
conv_in1 = Dropout(0.85)(l_flat)
conv_in1 = Dense(128, activation='relu')(conv_in1)

## do merge
conv_merged= keras.layers.concatenate([conv_ins, conv_in1])

#Meta input
meta_input = Input(shape=(x_train_metadata.shape[1],), name='aux_input')
x_drop = Dropout(0.85)(meta_input)
x_meta = Dense(64, activation='relu')(x_drop)
x = keras.layers.concatenate([conv_merged, x_meta])


main_output = Dense(6, activation='softmax', name='main_output')(x)
model_hybird = Model(inputs=[statement_input, meta_input], outputs=[main_outpu
t])

#*********************************************************************#
adam = Adam(lr=1e-4, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.2)
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)


model_hybird.compile(optimizer=sgd,
                     loss='categorical_crossentropy',
                     metrics=['categorical_accuracy'])

model_hybird.summary()
```

```
##
tb = TensorBoard()
csv_logger = keras.callbacks.CSVLogger('training.log')
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=1)
filepath= "weights.best.hdf5"
checkpoint = keras.callbacks.ModelCheckpoint(filepath,
                                             monitor='val_categorical_accurac
y',
                                             verbose=1, save_best_only=True, m
ode='max')

history13= model_hybird.fit({'main_input': x_train, 'aux_input': x_train_metad
ata},
                            {'main_output': y_train},epochs=12, batch_size=100,
                            validation_data=({'main_input': x_val, 'aux_input':
x_val_metadata},{'main_output': y_val}),
                            callbacks=[tb,csv_logger,checkpoint, es])
```

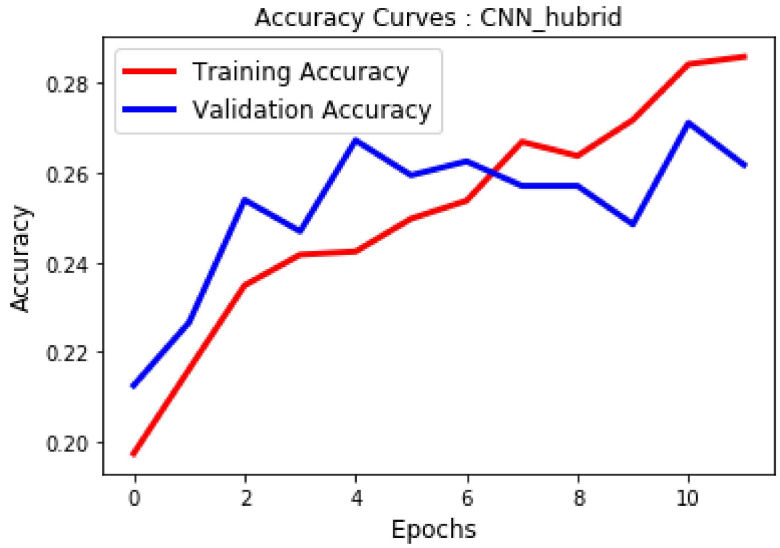
```
__________________________________________________________________________________________
____________________
Layer (type)                 Output Shape              Param #     Connected to
==========================================================================================
====================
main_input (InputLayer)      (None, 32)                0
__________________________________________________________________________________________
____________________
embedding_6 (Embedding)      (None, 32, 300)           3722700     main_input
[0][0]
__________________________________________________________________________________________
____________________
conv1d_21 (Conv1D)           (None, 30, 128)           115328      embedding_6
[0][0]
__________________________________________________________________________________________
____________________
conv1d_18 (Conv1D)           (None, 32, 128)           115328      embedding_6
[0][0]
__________________________________________________________________________________________
____________________
conv1d_19 (Conv1D)           (None, 32, 128)           153728      embedding_6
[0][0]
__________________________________________________________________________________________
____________________
conv1d_20 (Conv1D)           (None, 32, 128)           192128      embedding_6
[0][0]
__________________________________________________________________________________________
____________________
max_pooling1d_21 (MaxPooling1D) (None, 10, 128)        0           conv1d_21[0]
[0]
__________________________________________________________________________________________
____________________
max_pooling1d_18 (MaxPooling1D) (None, 10, 128)        0           conv1d_18[0]
[0]
__________________________________________________________________________________________
____________________
max_pooling1d_19 (MaxPooling1D) (None, 10, 128)        0           conv1d_19[0]
[0]
__________________________________________________________________________________________
____________________
max_pooling1d_20 (MaxPooling1D) (None, 10, 128)        0           conv1d_20[0]
[0]
__________________________________________________________________________________________
____________________
conv1d_22 (Conv1D)           (None, 8, 128)            49280       max_pooling1
d_21[0][0]
__________________________________________________________________________________________
____________________
flatten_10 (Flatten)         (None, 1280)              0           max_pooling1
d_18[0][0]
__________________________________________________________________________________________
____________________
flatten_11 (Flatten)         (None, 1280)              0           max_pooling1
d_19[0][0]
__________________________________________________________________________________________
____________________
flatten_12 (Flatten)         (None, 1280)              0           max_pooling1
```

```
d_20[0][0]
__________________________________________________________________________________________________
max_pooling1d_22 (MaxPooling1D) (None, 2, 128)       0           conv1d_22[0]
[0]
__________________________________________________________________________________________________
dropout_17 (Dropout)            (None, 1280)         0           flatten_10
[0][0]
__________________________________________________________________________________________________
dropout_18 (Dropout)            (None, 1280)         0           flatten_11
[0][0]
__________________________________________________________________________________________________
dropout_19 (Dropout)            (None, 1280)         0           flatten_12
[0][0]
__________________________________________________________________________________________________
flatten_13 (Flatten)            (None, 256)          0           max_pooling1
d_22[0][0]
__________________________________________________________________________________________________
concatenate_9 (Concatenate)     (None, 3840)         0           dropout_17
[0][0]
                                                                 dropout_18
[0][0]
                                                                 dropout_19
[0][0]
__________________________________________________________________________________________________
dropout_20 (Dropout)            (None, 256)          0           flatten_13
[0][0]
__________________________________________________________________________________________________
aux_input (InputLayer)          (None, 94)           0
__________________________________________________________________________________________________
dense_9 (Dense)                 (None, 128)          491648      concatenate_
9[0][0]
__________________________________________________________________________________________________
dense_10 (Dense)                (None, 128)          32896       dropout_20
[0][0]
__________________________________________________________________________________________________
dropout_21 (Dropout)            (None, 94)           0           aux_input[0]
[0]
__________________________________________________________________________________________________
concatenate_10 (Concatenate)    (None, 256)          0           dense_9[0]
[0]
                                                                 dense_10[0]
[0]
__________________________________________________________________________________________________
dense_11 (Dense)                (None, 64)           6080        dropout_21
```

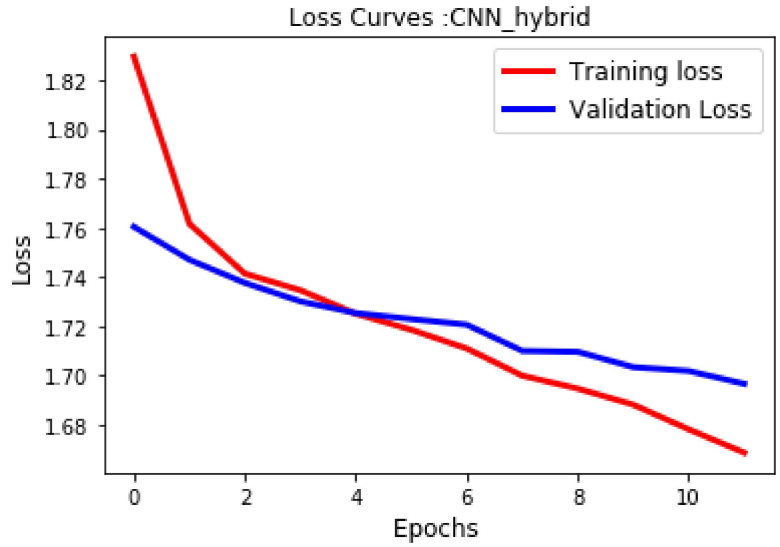
```
[0][0]
__________________________________________________________________________
____________________
concatenate_11 (Concatenate)    (None, 320)          0           concatenate_
10[0][0]
                                                                 dense_11[0]
[0]
__________________________________________________________________________
____________________
main_output (Dense)             (None, 6)            1926        concatenate_
11[0][0]
==========================================================================
====================
Total params: 4,881,042
Trainable params: 1,158,342
Non-trainable params: 3,722,700
__________________________________________________________________________
____________________
Train on 10240 samples, validate on 1284 samples
Epoch 1/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.8295 -
categorical_accuracy: 0.1975 - val_loss: 1.7604 - val_categorical_accuracy:
0.2126

Epoch 00001: val_categorical_accuracy improved from -inf to 0.21262, saving m
odel to weights.best.hdf5
Epoch 2/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.7616 -
categorical_accuracy: 0.2162 - val_loss: 1.7470 - val_categorical_accuracy:
0.2266

Epoch 00002: val_categorical_accuracy improved from 0.21262 to 0.22664, savin
g model to weights.best.hdf5
Epoch 3/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.7414 -
categorical_accuracy: 0.2349 - val_loss: 1.7376 - val_categorical_accuracy:
0.2539

Epoch 00003: val_categorical_accuracy improved from 0.22664 to 0.25389, savin
g model to weights.best.hdf5
Epoch 4/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.7346 -
categorical_accuracy: 0.2417 - val_loss: 1.7301 - val_categorical_accuracy:
0.2469

Epoch 00004: val_categorical_accuracy did not improve from 0.25389
Epoch 5/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.7251 -
categorical_accuracy: 0.2424 - val_loss: 1.7253 - val_categorical_accuracy:
0.2671

Epoch 00005: val_categorical_accuracy improved from 0.25389 to 0.26713, savin
g model to weights.best.hdf5
Epoch 6/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.7185 -
categorical_accuracy: 0.2497 - val_loss: 1.7229 - val_categorical_accuracy:
0.2593
```

```
Epoch 00006: val_categorical_accuracy did not improve from 0.26713
Epoch 7/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.7109 -
categorical_accuracy: 0.2537 - val_loss: 1.7206 - val_categorical_accuracy:
0.2625

Epoch 00007: val_categorical_accuracy did not improve from 0.26713
Epoch 8/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.6999 -
categorical_accuracy: 0.2668 - val_loss: 1.7100 - val_categorical_accuracy:
0.2570

Epoch 00008: val_categorical_accuracy did not improve from 0.26713
Epoch 9/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.6947 -
categorical_accuracy: 0.2637 - val_loss: 1.7096 - val_categorical_accuracy:
0.2570

Epoch 00009: val_categorical_accuracy did not improve from 0.26713
Epoch 10/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.6882 -
categorical_accuracy: 0.2717 - val_loss: 1.7034 - val_categorical_accuracy:
0.2484

Epoch 00010: val_categorical_accuracy did not improve from 0.26713
Epoch 11/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.6781 -
categorical_accuracy: 0.2841 - val_loss: 1.7018 - val_categorical_accuracy:
0.2710

Epoch 00011: val_categorical_accuracy improved from 0.26713 to 0.27103, savin
g model to weights.best.hdf5
Epoch 12/12
10240/10240 [==============================] - 17s 2ms/step - loss: 1.6686 -
categorical_accuracy: 0.2857 - val_loss: 1.6966 - val_categorical_accuracy:
0.2617

Epoch 00012: val_categorical_accuracy did not improve from 0.27103
```

In [27]:
```
## to see performance difference
accu_curve=plt.figure()
plt.plot(history13.history['categorical_accuracy'],'r',linewidth=3.0)
plt.plot(history13.history['val_categorical_accuracy'],'b',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Accuracy',fontsize=12)
plt.title('Accuracy Curves : CNN_hubrid',fontsize=12)
# accu_curve.savefig('accuracy_CNN_hubrid.png')
plt.show()
##^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##
loss_curve = plt.figure()
plt.plot(history13.history['loss'],'r',linewidth=3.0)
plt.plot(history13.history['val_loss'],'b',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Loss',fontsize=12)
plt.title('Loss Curves :CNN_hybrid',fontsize=12)
# loss_curve.savefig('loss_CNN_hybrid.png')
loss_curve.show()
```




```
C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\lib\site-packages\matplotl
ib\figure.py:445: UserWarning: Matplotlib is currently using module://ipykern
el.pylab.backend_inline, which is a non-GUI backend, so cannot show the figur
e.
  % get_backend())
```



1.4 C-LSTM model for fake news classification

To better understand performance gap between different convolutional models, here I want to see Convolutional -LSTM model for fake news classification. Note that I didn't run multiple convolutional filers for fake news texts because it raise strange error, so I'll proceed somehow simple but powerful model for fake news classification.

In [28]:
```
## C-LSTM model implementation (trail version)
from keras.layers import LSTM
from keras.optimizers import Adam
from keras.layers import LSTM, Bidirectional

statement_input = Input(shape=(num_steps,), dtype='int64', name='main_input')
# embed_layer = Embedding(output_dim=hidden_size, input_dim=vocab_length+1, in
put_length=num_steps)(statement_input) #Train embeddings from scratch
embed_layer = Embedding(vocab_length+1,embedding_dims,weights=[embedding_weigh
ts],input_length=num_steps,trainable=False)(statement_input) #Preloaded glove
 embeddings
l_lstm = Bidirectional(LSTM(100, return_sequences=True, dropout=0.25, recurren
t_dropout=0.1))(embed_layer)
# lstm_pool = GlobalMaxPool1D()(l_lstm)

## let's add convolutional layer on it
l_conv1 = Conv1D(128, 3, activation="relu")(l_lstm)
l_pool1 = MaxPooling1D(3)(l_conv1)
l_conv2 = Conv1D(128,3,activation="relu")(l_pool1)
l_pool2 = MaxPool1D(3)(l_conv2)
l_flat = Flatten()(l_pool2)
l_drop = Dropout(0.8)(l_flat)
conv_in = Dense(hidden_size, activation="relu")(l_drop)

### metadata
meta_input = Input(shape=(x_train_metadata.shape[1],), name='aux_input')
x_meta_hidden = Dense(64, activation='relu')(meta_input)
x_meta_reg = Dropout(0.6)(x_meta_hidden)

conv_final = keras.layers.concatenate([conv_in, x_meta_reg])
model_output = Dense(6, activation='softmax', name='main_output')(conv_final)
model_CLSTM = Model(inputs=[statement_input, meta_input], outputs=[model_outpu
t])

## compile model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model_CLSTM.compile(optimizer=sgd,
                    loss='categorical_crossentropy',
                    metrics=['categorical_accuracy'])

model_CLSTM.summary()

tb = TensorBoard()
csv_logger = keras.callbacks.CSVLogger('training.log')
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=2)
filepath= "weights.best.hdf5"
checkpoint = keras.callbacks.ModelCheckpoint(filepath,
                                             monitor='val_categorical_accurac
y',
                                             verbose=1, save_best_only=True, m
ode='max')

history_clstm = model_CLSTM.fit({'main_input': x_train, 'aux_input': x_train_m
etadata},
                                {'main_output': y_train},epochs=num_epochs, ba
tch_size=batch_size,
```

```
                                    validation_data=({'main_input': x_val, 'aux_in
put': x_val_metadata},{'main_output': y_val}),
                                    callbacks=[tb,csv_logger,checkpoint, es])
```

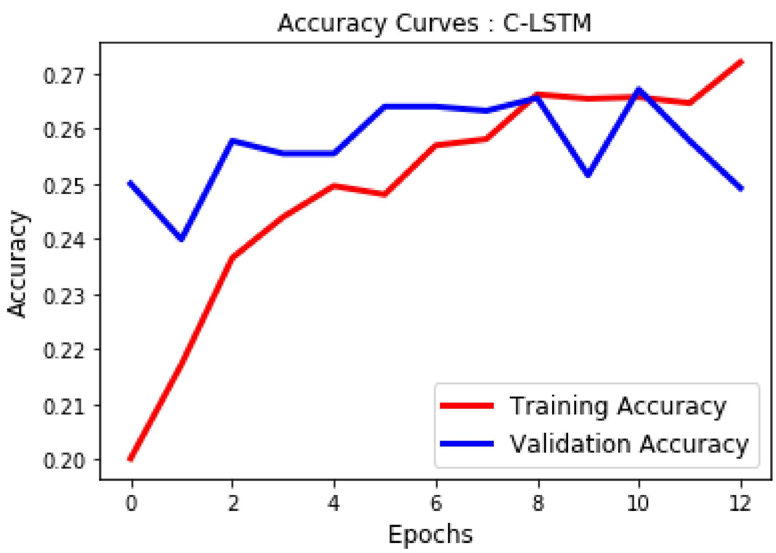
Layer (type)	Output Shape	Param #	Connected to
main_input (InputLayer)	(None, 32)	0	
embedding_7 (Embedding)	(None, 32, 300)	3722700	main_input[0][0]
bidirectional_1 (Bidirectional)	(None, 32, 200)	320800	embedding_7[0][0]
conv1d_23 (Conv1D)	(None, 30, 128)	76928	bidirectional_1[0][0]
max_pooling1d_23 (MaxPooling1D)	(None, 10, 128)	0	conv1d_23[0][0]
conv1d_24 (Conv1D)	(None, 8, 128)	49280	max_pooling1d_23[0][0]
max_pooling1d_24 (MaxPooling1D)	(None, 2, 128)	0	conv1d_24[0][0]
flatten_14 (Flatten)	(None, 256)	0	max_pooling1d_24[0][0]
aux_input (InputLayer)	(None, 94)	0	
dropout_22 (Dropout)	(None, 256)	0	flatten_14[0][0]
dense_13 (Dense)	(None, 64)	6080	aux_input[0][0]
dense_12 (Dense)	(None, 150)	38550	dropout_22[0][0]
dropout_23 (Dropout)	(None, 64)	0	dense_13[0][0]
concatenate_12 (Concatenate)	(None, 214)	0	dense_12[0][0]

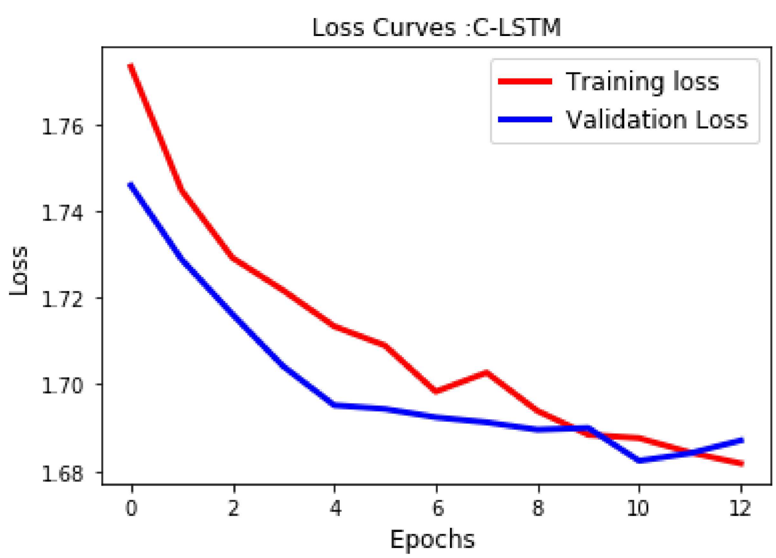
```
                                                                 dropout_23
[0][0]
__________________________________________________________________________
_____________________
main_output (Dense)             (None, 6)            1290        concatenate_
12[0][0]
==========================================================================
====================
Total params: 4,215,628
Trainable params: 492,928
Non-trainable params: 3,722,700
__________________________________________________________________________
_____________________
Train on 10240 samples, validate on 1284 samples
Epoch 1/30
10240/10240 [==============================] - 21s 2ms/step - loss: 1.7732 -
categorical_accuracy: 0.2001 - val_loss: 1.7459 - val_categorical_accuracy:
0.2500

Epoch 00001: val_categorical_accuracy improved from -inf to 0.25000, saving m
odel to weights.best.hdf5
Epoch 2/30
10240/10240 [==============================] - 18s 2ms/step - loss: 1.7447 -
categorical_accuracy: 0.2172 - val_loss: 1.7288 - val_categorical_accuracy:
0.2399

Epoch 00002: val_categorical_accuracy did not improve from 0.25000
Epoch 3/30
10240/10240 [==============================] - 18s 2ms/step - loss: 1.7292 -
categorical_accuracy: 0.2365 - val_loss: 1.7161 - val_categorical_accuracy:
0.2578

Epoch 00003: val_categorical_accuracy improved from 0.25000 to 0.25779, savin
g model to weights.best.hdf5
Epoch 4/30
10240/10240 [==============================] - 19s 2ms/step - loss: 1.7216 -
categorical_accuracy: 0.2439 - val_loss: 1.7041 - val_categorical_accuracy:
0.2555

Epoch 00004: val_categorical_accuracy did not improve from 0.25779
Epoch 5/30
10240/10240 [==============================] - 19s 2ms/step - loss: 1.7134 -
categorical_accuracy: 0.2496 - val_loss: 1.6952 - val_categorical_accuracy:
0.2555

Epoch 00005: val_categorical_accuracy did not improve from 0.25779
Epoch 6/30
10240/10240 [==============================] - 19s 2ms/step - loss: 1.7089 -
categorical_accuracy: 0.2480 - val_loss: 1.6943 - val_categorical_accuracy:
0.2640

Epoch 00006: val_categorical_accuracy improved from 0.25779 to 0.26402, savin
g model to weights.best.hdf5
Epoch 7/30
10240/10240 [==============================] - 18s 2ms/step - loss: 1.6983 -
categorical_accuracy: 0.2569 - val_loss: 1.6924 - val_categorical_accuracy:
0.2640
```

```
Epoch 00007: val_categorical_accuracy did not improve from 0.26402
Epoch 8/30
10240/10240 [==============================] - 19s 2ms/step - loss: 1.7027 -
categorical_accuracy: 0.2581 - val_loss: 1.6912 - val_categorical_accuracy:
0.2632

Epoch 00008: val_categorical_accuracy did not improve from 0.26402
Epoch 9/30
10240/10240 [==============================] - 19s 2ms/step - loss: 1.6939 -
categorical_accuracy: 0.2662 - val_loss: 1.6895 - val_categorical_accuracy:
0.2656

Epoch 00009: val_categorical_accuracy improved from 0.26402 to 0.26558, savin
g model to weights.best.hdf5
Epoch 10/30
10240/10240 [==============================] - 18s 2ms/step - loss: 1.6884 -
categorical_accuracy: 0.2654 - val_loss: 1.6899 - val_categorical_accuracy:
0.2516

Epoch 00010: val_categorical_accuracy did not improve from 0.26558
Epoch 11/30
10240/10240 [==============================] - 19s 2ms/step - loss: 1.6876 -
categorical_accuracy: 0.2657 - val_loss: 1.6824 - val_categorical_accuracy:
0.2671

Epoch 00011: val_categorical_accuracy improved from 0.26558 to 0.26713, savin
g model to weights.best.hdf5
Epoch 12/30
10240/10240 [==============================] - 20s 2ms/step - loss: 1.6843 -
categorical_accuracy: 0.2646 - val_loss: 1.6841 - val_categorical_accuracy:
0.2578

Epoch 00012: val_categorical_accuracy did not improve from 0.26713
Epoch 13/30
10240/10240 [==============================] - 21s 2ms/step - loss: 1.6817 -
categorical_accuracy: 0.2721 - val_loss: 1.6871 - val_categorical_accuracy:
0.2492

Epoch 00013: val_categorical_accuracy did not improve from 0.26713
Epoch 00013: early stopping
```

In [ ]:
```
### Experimental modification on c-lstm model for fake news classification

"""
## let's add lstm
l_lstm = Bidirectional(LSTM(100, return_sequences=True, dropout=0.25, recurren
t_dropout=0.1))(embed_layer)
lstm_pool = GlobalMaxPool1D()(l_lstm)
lstm_drop = Dropout(0.6)(lstm_pool)
lstm_dense = Dense(hidden_size, activation="relu")(lstm_drop)

## let's add convolutional layer
l_conv1 = Conv1D(128, 3, activation="relu")(embed_layer)
l_pool1 = MaxPooling1D(3)(l_conv1)
l_conv2 = Conv1D(128,3,activation="relu")(l_pool1)
l_pool2 = MaxPool1D(3)(l_conv2)
l_flat = Flatten()(l_pool2)
l_drop = Dropout(0.6)(l_flat)
conv_in = Dense(hidden_size, activation="relu")(l_drop)

## merge lstm and convs
conv_ins = keras.layers.concatenate([conv_in, lstm_dense])

### metadata
meta_input = Input(shape=(x_train_metadata.shape[1],), name='aux_input')
x_meta_hidden = Dense(64, activation='relu')(meta_input)
x_meta_reg = Dropout(0.6)(x_meta_hidden)

conv_final = keras.layers.concatenate([conv_ins, x_meta_reg])
"""
```

In [29]:
```
### visualize c-lstm model output
accu_curve=plt.figure()
plt.plot(history_clstm.history['categorical_accuracy'],'r',linewidth=3.0)
plt.plot(history_clstm.history['val_categorical_accuracy'],'b',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Accuracy',fontsize=12)
plt.title('Accuracy Curves : C-LSTM',fontsize=12)
# accu_curve.savefig('accuracy_clstm_improved_v1.6.png')
plt.show()
##^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##
loss_curve = plt.figure()
plt.plot(history_clstm.history['loss'],'r',linewidth=3.0)
plt.plot(history_clstm.history['val_loss'],'b',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Loss',fontsize=12)
plt.title('Loss Curves :C-LSTM',fontsize=12)
# loss_curve.savefig('loss_clstm_improved_v1.6.png')
loss_curve.show()
```





Modified version of C-LSTM model for fake news classification

In [30]:
```
from keras.layers import LSTM
from keras.optimizers import Adam
from keras.layers import LSTM, Bidirectional

##+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
kernel_arr = []
statement_input = Input(shape=(num_steps,), dtype='int32', name='main_input')
embedded_sequences = Embedding(vocab_length+1,embedding_dims,weights=[embeddin
g_weights],input_length=num_steps,trainable=False)(statement_input) #Preloaded
glove embeddings
# x = Embedding(output_dim=hidden_size, input_dim=vocab_length+1, input_length
=num_steps)(statement_input) #Train embeddings from scratch
x_lstm = Bidirectional(LSTM(128, return_sequences=True, dropout=0.4, recurrent
_dropout=0.5))(embedded_sequences)

for kernel in kernel_sizes:
    x_1 = Conv1D(filters=filter_size,kernel_size=kernel,
                 padding="valid", activation="relu", strides=1)(x_lstm)
    x_pool = MaxPool1D(4)(x_1)
    #x_pool = GlobalMaxPool1D()(x_1)
    x_flat = Flatten()(x_pool)
    x_drop = Dropout(0.85)(x_flat)
    kernel_arr.append(x_drop)

conv_in = keras.layers.concatenate(kernel_arr)
# conv_in = Dropout(0.7)(conv_in)
conv_in = Dense(128, activation='relu')(conv_in)

"""
## use bidirectional lstm
x_lstm = Bidirectional(LSTM(100, return_sequences=True, dropout=0.25, recurren
t_dropout=0.1))(x_1)
lstm_pool = GlobalMaxPool1D()(x_lstm)
lstm_dense = Dense(100, activation="relu")(lstm_pool)

## first concatenate
conv_ins= keras.layers.concatenate([conv_in, lstm_dense])
"""

#Meta input
meta_input = Input(shape=(x_train_metadata.shape[1],), name='aux_input')
x_drop = Dropout(0.8)(meta_input)
x_meta = Dense(64, activation='relu')(x_drop)
x = keras.layers.concatenate([conv_in,x_meta])

main_output = Dense(6, activation='softmax', name='main_output')(x)
model_clstm_ = Model(inputs=[statement_input, meta_input], outputs=[main_outpu
t])


## compile model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model_clstm_.compile(optimizer=sgd,
                     loss='categorical_crossentropy',
                     metrics=['categorical_accuracy'])
```

```
model_clstm_.summary()

##+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++##
tb = TensorBoard()
csv_logger = keras.callbacks.CSVLogger('training.log')
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
filepath= "weights.best.hdf5"
checkpoint = keras.callbacks.ModelCheckpoint(filepath,
                                             monitor='val_categorical_accurac
y',
                                             verbose=1, save_best_only=True, m
ode='max')

history_clstm_ = model_clstm_.fit({'main_input': x_train, 'aux_input': x_train
_metadata},
                                  {'main_output': y_train},epochs=num_epochs, ba
tch_size=64,
                                  validation_data=({'main_input': x_val, 'aux_in
put': x_val_metadata},{'main_output': y_val}),
                                  callbacks=[tb,csv_logger,checkpoint, es])
```

Layer (type)	Output Shape	Param #	Connected to
main_input (InputLayer)	(None, 32)	0	
embedding_8 (Embedding)	(None, 32, 300)	3722700	main_input[0][0]
bidirectional_2 (Bidirectional)	(None, 32, 256)	439296	embedding_8[0][0]
conv1d_25 (Conv1D)	(None, 30, 128)	98432	bidirectional_2[0][0]
conv1d_26 (Conv1D)	(None, 29, 128)	131200	bidirectional_2[0][0]
conv1d_27 (Conv1D)	(None, 28, 128)	163968	bidirectional_2[0][0]
max_pooling1d_25 (MaxPooling1D)	(None, 7, 128)	0	conv1d_25[0][0]
max_pooling1d_26 (MaxPooling1D)	(None, 7, 128)	0	conv1d_26[0][0]
max_pooling1d_27 (MaxPooling1D)	(None, 7, 128)	0	conv1d_27[0][0]
flatten_15 (Flatten)	(None, 896)	0	max_pooling1d_25[0][0]
flatten_16 (Flatten)	(None, 896)	0	max_pooling1d_26[0][0]
flatten_17 (Flatten)	(None, 896)	0	max_pooling1d_27[0][0]
dropout_24 (Dropout)	(None, 896)	0	flatten_15[0][0]
dropout_25 (Dropout)	(None, 896)	0	flatten_16

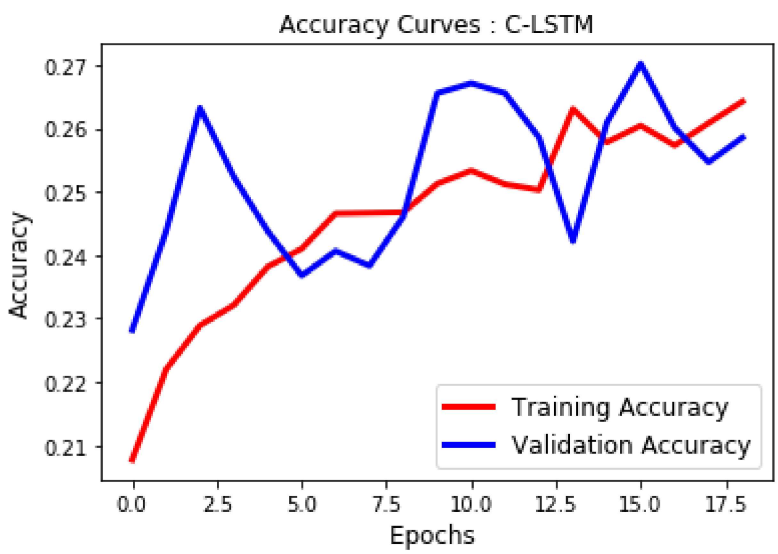
```
[0][0]
==================================================================================================
dropout_26 (Dropout)            (None, 896)          0           flatten_17
[0][0]
__________________________________________________________________________________________________
aux_input (InputLayer)          (None, 94)           0
__________________________________________________________________________________________________
concatenate_13 (Concatenate)    (None, 2688)         0           dropout_24
[0][0]
                                                                 dropout_25
[0][0]
                                                                 dropout_26
[0][0]
__________________________________________________________________________________________________
dropout_27 (Dropout)            (None, 94)           0           aux_input[0]
[0]
__________________________________________________________________________________________________
dense_14 (Dense)                (None, 128)          344192      concatenate_
13[0][0]
__________________________________________________________________________________________________
dense_15 (Dense)                (None, 64)           6080        dropout_27
[0][0]
__________________________________________________________________________________________________
concatenate_14 (Concatenate)    (None, 192)          0           dense_14[0]
[0]
                                                                 dense_15[0]
[0]
__________________________________________________________________________________________________
main_output (Dense)             (None, 6)            1158        concatenate_
14[0][0]
==================================================================================================
Total params: 4,907,026
Trainable params: 1,184,326
Non-trainable params: 3,722,700
__________________________________________________________________________________________________
Train on 10240 samples, validate on 1284 samples
Epoch 1/30
10240/10240 [==============================] - 38s 4ms/step - loss: 1.7798 -
categorical_accuracy: 0.2077 - val_loss: 1.7592 - val_categorical_accuracy:
0.2282

Epoch 00001: val_categorical_accuracy improved from -inf to 0.22819, saving m
odel to weights.best.hdf5
Epoch 2/30
10240/10240 [==============================] - 36s 4ms/step - loss: 1.7560 -
categorical_accuracy: 0.2220 - val_loss: 1.7374 - val_categorical_accuracy:
0.2438
```

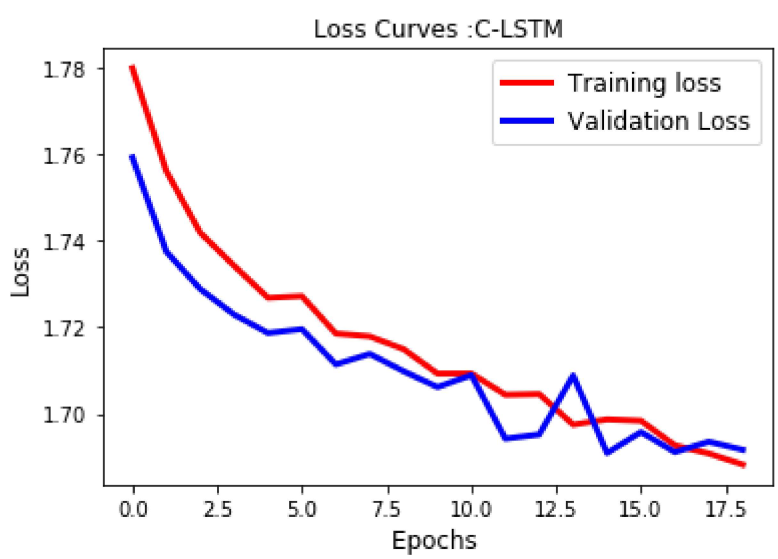
```
Epoch 00002: val_categorical_accuracy improved from 0.22819 to 0.24377, savin
g model to weights.best.hdf5
Epoch 3/30
10240/10240 [==============================] - 35s 3ms/step - loss: 1.7418 -
categorical_accuracy: 0.2289 - val_loss: 1.7288 - val_categorical_accuracy:
0.2632

Epoch 00003: val_categorical_accuracy improved from 0.24377 to 0.26324, savin
g model to weights.best.hdf5
Epoch 4/30
10240/10240 [==============================] - 35s 3ms/step - loss: 1.7342 -
categorical_accuracy: 0.2321 - val_loss: 1.7229 - val_categorical_accuracy:
0.2523

Epoch 00004: val_categorical_accuracy did not improve from 0.26324
Epoch 5/30
10240/10240 [==============================] - 36s 3ms/step - loss: 1.7268 -
categorical_accuracy: 0.2382 - val_loss: 1.7186 - val_categorical_accuracy:
0.2438

Epoch 00005: val_categorical_accuracy did not improve from 0.26324
Epoch 6/30
10240/10240 [==============================] - 38s 4ms/step - loss: 1.7271 -
categorical_accuracy: 0.2410 - val_loss: 1.7195 - val_categorical_accuracy:
0.2368

Epoch 00006: val_categorical_accuracy did not improve from 0.26324
Epoch 7/30
10240/10240 [==============================] - 36s 4ms/step - loss: 1.7185 -
categorical_accuracy: 0.2466 - val_loss: 1.7114 - val_categorical_accuracy:
0.2407

Epoch 00007: val_categorical_accuracy did not improve from 0.26324
Epoch 8/30
10240/10240 [==============================] - 37s 4ms/step - loss: 1.7178 -
categorical_accuracy: 0.2467 - val_loss: 1.7138 - val_categorical_accuracy:
0.2383

Epoch 00008: val_categorical_accuracy did not improve from 0.26324
Epoch 9/30
10240/10240 [==============================] - 36s 3ms/step - loss: 1.7149 -
categorical_accuracy: 0.2468 - val_loss: 1.7098 - val_categorical_accuracy:
0.2461

Epoch 00009: val_categorical_accuracy did not improve from 0.26324
Epoch 10/30
10240/10240 [==============================] - 34s 3ms/step - loss: 1.7092 -
categorical_accuracy: 0.2513 - val_loss: 1.7061 - val_categorical_accuracy:
0.2656

Epoch 00010: val_categorical_accuracy improved from 0.26324 to 0.26558, savin
g model to weights.best.hdf5
Epoch 11/30
10240/10240 [==============================] - 36s 4ms/step - loss: 1.7093 -
categorical_accuracy: 0.2533 - val_loss: 1.7089 - val_categorical_accuracy:
0.2671
```

```
Epoch 00011: val_categorical_accuracy improved from 0.26558 to 0.26713, savin
g model to weights.best.hdf5
Epoch 12/30
10240/10240 [==============================] - 36s 4ms/step - loss: 1.7044 -
categorical_accuracy: 0.2512 - val_loss: 1.6942 - val_categorical_accuracy:
0.2656

Epoch 00012: val_categorical_accuracy did not improve from 0.26713
Epoch 13/30
10240/10240 [==============================] - 35s 3ms/step - loss: 1.7045 -
categorical_accuracy: 0.2503 - val_loss: 1.6952 - val_categorical_accuracy:
0.2586

Epoch 00013: val_categorical_accuracy did not improve from 0.26713
Epoch 14/30
10240/10240 [==============================] - 36s 4ms/step - loss: 1.6975 -
categorical_accuracy: 0.2631 - val_loss: 1.7088 - val_categorical_accuracy:
0.2422

Epoch 00014: val_categorical_accuracy did not improve from 0.26713
Epoch 15/30
10240/10240 [==============================] - 36s 4ms/step - loss: 1.6987 -
categorical_accuracy: 0.2578 - val_loss: 1.6909 - val_categorical_accuracy:
0.2609

Epoch 00015: val_categorical_accuracy did not improve from 0.26713
Epoch 16/30
10240/10240 [==============================] - 37s 4ms/step - loss: 1.6983 -
categorical_accuracy: 0.2604 - val_loss: 1.6957 - val_categorical_accuracy:
0.2702

Epoch 00016: val_categorical_accuracy improved from 0.26713 to 0.27025, savin
g model to weights.best.hdf5
Epoch 17/30
10240/10240 [==============================] - 37s 4ms/step - loss: 1.6927 -
categorical_accuracy: 0.2573 - val_loss: 1.6911 - val_categorical_accuracy:
0.2601

Epoch 00017: val_categorical_accuracy did not improve from 0.27025
Epoch 18/30
10240/10240 [==============================] - 36s 3ms/step - loss: 1.6908 -
categorical_accuracy: 0.2608 - val_loss: 1.6935 - val_categorical_accuracy:
0.2547

Epoch 00018: val_categorical_accuracy did not improve from 0.27025
Epoch 19/30
10240/10240 [==============================] - 34s 3ms/step - loss: 1.6882 -
categorical_accuracy: 0.2643 - val_loss: 1.6917 - val_categorical_accuracy:
0.2586

Epoch 00019: val_categorical_accuracy did not improve from 0.27025
Epoch 00019: early stopping
```

In [31]:

```
### visualize c-lstm model output
accu_curve=plt.figure()
plt.plot(history_clstm_.history['categorical_accuracy'],'r',linewidth=3.0)
plt.plot(history_clstm_.history['val_categorical_accuracy'],'b',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Accuracy',fontsize=12)
plt.title('Accuracy Curves : C-LSTM',fontsize=12)
# accu_curve.savefig('accuracy_clstm_improved_v1.6.png')
plt.show()
##^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##
loss_curve = plt.figure()
plt.plot(history_clstm_.history['loss'],'r',linewidth=3.0)
plt.plot(history_clstm_.history['val_loss'],'b',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Loss',fontsize=12)
plt.title('Loss Curves :C-LSTM',fontsize=12)
# loss_curve.savefig('loss_clstm_improved_v1.6.png')
loss_curve.show()
```





Character level CNN model for fake news classification

In [ ]:
```
import tensorflow as tf
from keras import layers
from keras import backend as K
from math import sqrt
K.set_image_dim_ordering('th')

from keras import Model
from keras.layers import Input, Dense, Concatenate, Embedding, Flatten
from keras.layers import AlphaDropout
from keras.layers import ThresholdedReLU
from keras.callbacks import TensorBoard
from keras.layers import Convolution1D, GlobalMaxPooling1D
```

In [ ]:
```
import os
os.environ['KERAS_BACKEND'] = 'theano'
import keras as K
```

character level classification (simple version)

In [ ]:
```
conv_layers=[[256, 7, 3],
             [256, 7, 3],
             [256, 3, -1],
             [256, 3, -1],
             [256, 3, -1],
             [256, 3, 3]]

fully_connected_layers=[1024, 1024]
dropout_p=0.5
threshold=1e-06
input_size=70
alphabet='abcdefghijklmnopqrstuvwxyz0123456789-,;.!?:\'"/\\|_@#$%^&*~`+-=<>()
[]{}'

batch_size=128
checkpoint_every=70
epochs=5000
# evaluate_every=100
batch_size=128
checkpoint_every=100
epochs=5000
# evaluate_every=100
embedding_size=128
alphabet_size=69
```

In [ ]:
```
## define model input, create embedding layer
sent_inputs = Input(shape=(input_size,), name='sent_input', dtype='int64')  #
 shape=(?, 1014)

conv = Embedding(alphabet_size+1, embedding_size, input_length=input_size, tra
inable=False)(sent_inputs)
# Conv
conv_lay=[]
for filter_num, filter_size, pooling_size in conv_layers:
    x_conv = Conv1D(filter_num, filter_size, padding="same")(conv)
    x_pool  = GlobalMaxPooling1D()(x_conv)
    x_drop = Dropout(0.6)(x_pool)
    conv_lay.append(x_drop)
#     if pooling_size != -1:
#         conv = MaxPooling1D(pool_size=pooling_size)(conv) # Final shape=(Non
e, 34, 256)

# x = Flatten()(conv_lay) # (None, 8704)
x=concatenate(conv_lay)

# Fully connected layers
for dense_size in fully_connected_layers:
    x = Dense(dense_size, activation='relu')(x) # dense_size == 1024
    x = Dropout(dropout_p)(x)

model_output = Dense(6, activation='softmax')(x)
# Build model
adam = Adam(lr=1e-4, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.2)

model_char = Model(inputs=sent_inputs, outputs=model_output)
model_char.compile(optimizer=adam,
                   loss='categorical_crossentropy',
                   metrics=['categorical_accuracy'])

model_char.summary()
tb = TensorBoard()
csv_logger = keras.callbacks.CSVLogger('training.log')
es=EarlyStopping(monitor="val_loss", mode='min', verbose=1, patience=3 )
filepath= "weights.best.hdf5"
checkpoint = keras.callbacks.ModelCheckpoint(filepath,
                                             monitor='val_categorical_accurac
y',
                                             verbose=1, save_best_only=True, m
ode='max')


# history_char= model_char.fit(x_train,y_train,epochs=num_epochs, batch_size=b
atch_size,
#                                 validation_data=(x_val,y_val))
```

In [ ]:
```
model_char.fit(x_train,y_train,epochs=10, batch_size=64, validation_data=(x_va
l, y_val))
```

Recurrent CNN model for fake news classification

In [32]:
```
hidden_dim_1 = 128
hidden_dim_2 = 128

kernel_arr = []

statement_input = Input(shape=(num_steps,), dtype='int32', name='main_input')
embed_sequences = Embedding(vocab_length+1,embedding_dims,
                            weights=[embedding_weights],input_length=num_steps
,
                            trainable=False)(statement_input)

L1 = Bidirectional(LSTM(hidden_dim_1, return_sequences=True))(embed_sequences)
L2 = TimeDistributed(Dense(hidden_dim_2, activation = "tanh"))(L1)

l_conv1 = Conv1D(128, 3, activation="relu")(L2)
l_pool1 = MaxPooling1D(3)(l_conv1)
l_conv2 = Conv1D(128,3,activation="relu")(l_pool1)
l_pool2 = MaxPool1D(3)(l_conv2)
pool_rnn = Lambda(lambda x: K.max(x, axis = 1),
                  output_shape = (hidden_dim_2, ))(l_pool2)
# l_flat = Flatten()(pool_rnn)
l_drop = Dropout(0.8)(pool_rnn)
conv_in = Dense(hidden_size, activation="relu")(l_drop)

### metadata
meta_input = Input(shape=(x_train_metadata.shape[1],), name='aux_input')
x_meta_hidden = Dense(64, activation='relu')(meta_input)
x_meta_reg = Dropout(0.6)(x_meta_hidden)

conv_final = keras.layers.concatenate([conv_in, x_meta_reg])
model_output = Dense(6, activation='softmax', name='main_output')(conv_final)
model_RCNN = Model(inputs=[statement_input, meta_input], outputs=[model_output
])

## compile model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model_RCNN.compile(optimizer=sgd,
                   loss='categorical_crossentropy',
                   metrics=['categorical_accuracy'])

model_RCNN.summary()

tb = TensorBoard()
csv_logger = keras.callbacks.CSVLogger('training.log')
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=2)
filepath= "weights.best.hdf5"
checkpoint = keras.callbacks.ModelCheckpoint(filepath,
                                             monitor='val_categorical_accurac
y',
                                             verbose=1, save_best_only=True, m
ode='max')

history_RCNN = model_RCNN.fit({'main_input': x_train, 'aux_input': x_train_met
adata},
                              {'main_output': y_train},epochs=20, batch_size=
batch_size,
```

```
                                    validation_data=({'main_input': x_val, 'aux_inp
ut': x_val_metadata},{'main_output': y_val}),
                                    callbacks=[tb,csv_logger,checkpoint, es])
```

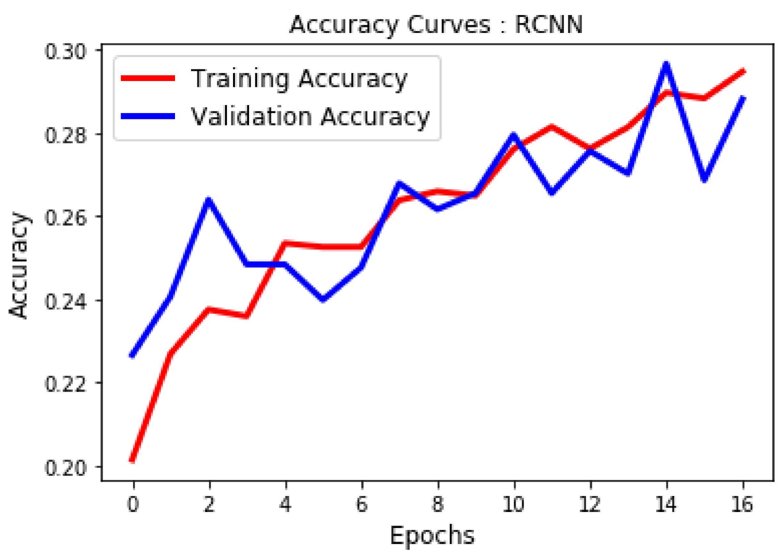
Layer (type)	Output Shape	Param #	Connected to
main_input (InputLayer)	(None, 32)	0	
embedding_9 (Embedding)	(None, 32, 300)	3722700	main_input[0][0]
bidirectional_3 (Bidirectional)	(None, 32, 256)	439296	embedding_9[0][0]
time_distributed_1 (TimeDistributed_3[0][0]	(None, 32, 128)	32896	bidirectional_3[0][0]
conv1d_28 (Conv1D)	(None, 30, 128)	49280	time_distributed_1[0][0]
max_pooling1d_28 (MaxPooling1D)	(None, 10, 128)	0	conv1d_28[0][0]
conv1d_29 (Conv1D)	(None, 8, 128)	49280	max_pooling1d_28[0][0]
max_pooling1d_29 (MaxPooling1D)	(None, 2, 128)	0	conv1d_29[0][0]
lambda_1 (Lambda)	(None, 128)	0	max_pooling1d_29[0][0]
aux_input (InputLayer)	(None, 94)	0	
dropout_28 (Dropout)	(None, 128)	0	lambda_1[0][0]
dense_18 (Dense)	(None, 64)	6080	aux_input[0][0]
dense_17 (Dense)	(None, 150)	19350	dropout_28[0][0]
dropout_29 (Dropout)	(None, 64)	0	dense_18[0][0]

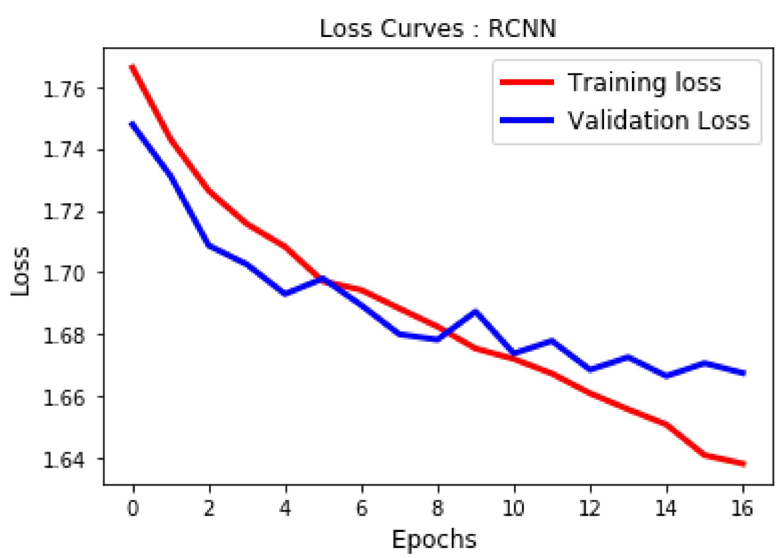
```
__________________________________________________________________________________________
concatenate_15 (Concatenate)    (None, 214)          0           dense_17[0]
[0]
                                                                 dropout_29
[0][0]
__________________________________________________________________________________________
main_output (Dense)             (None, 6)            1290        concatenate_
15[0][0]
==========================================================================================
Total params: 4,320,172
Trainable params: 597,472
Non-trainable params: 3,722,700
__________________________________________________________________________________________
WARNING:tensorflow:From C:\Users\jshayi2\AppData\Local\Continuum\anaconda3\li
b\site-packages\tensorflow\python\ops\math_grad.py:102: div (from tensorflow.
python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Deprecated in favor of operator or tf.math.divide.
Train on 10240 samples, validate on 1284 samples
Epoch 1/20
10240/10240 [==============================] - 22s 2ms/step - loss: 1.7661 -
categorical_accuracy: 0.2015 - val_loss: 1.7478 - val_categorical_accuracy:
0.2266

Epoch 00001: val_categorical_accuracy improved from -inf to 0.22664, saving m
odel to weights.best.hdf5
Epoch 2/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.7430 -
categorical_accuracy: 0.2269 - val_loss: 1.7311 - val_categorical_accuracy:
0.2407

Epoch 00002: val_categorical_accuracy improved from 0.22664 to 0.24065, savin
g model to weights.best.hdf5
Epoch 3/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.7264 -
categorical_accuracy: 0.2376 - val_loss: 1.7087 - val_categorical_accuracy:
0.2640

Epoch 00003: val_categorical_accuracy improved from 0.24065 to 0.26402, savin
g model to weights.best.hdf5
Epoch 4/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.7157 -
categorical_accuracy: 0.2359 - val_loss: 1.7026 - val_categorical_accuracy:
0.2484

Epoch 00004: val_categorical_accuracy did not improve from 0.26402
Epoch 5/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.7083 -
categorical_accuracy: 0.2535 - val_loss: 1.6930 - val_categorical_accuracy:
0.2484

Epoch 00005: val_categorical_accuracy did not improve from 0.26402
Epoch 6/20
```

```
10240/10240 [==============================] - 21s 2ms/step - loss: 1.6971 -
categorical_accuracy: 0.2526 - val_loss: 1.6980 - val_categorical_accuracy:
0.2399

Epoch 00006: val_categorical_accuracy did not improve from 0.26402
Epoch 7/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.6944 -
categorical_accuracy: 0.2526 - val_loss: 1.6894 - val_categorical_accuracy:
0.2477

Epoch 00007: val_categorical_accuracy did not improve from 0.26402
Epoch 8/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.6884 -
categorical_accuracy: 0.2639 - val_loss: 1.6800 - val_categorical_accuracy:
0.2679

Epoch 00008: val_categorical_accuracy improved from 0.26402 to 0.26791, savin
g model to weights.best.hdf5
Epoch 9/20
10240/10240 [==============================] - 22s 2ms/step - loss: 1.6825 -
categorical_accuracy: 0.2660 - val_loss: 1.6783 - val_categorical_accuracy:
0.2617

Epoch 00009: val_categorical_accuracy did not improve from 0.26791
Epoch 10/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.6754 -
categorical_accuracy: 0.2649 - val_loss: 1.6874 - val_categorical_accuracy:
0.2656

Epoch 00010: val_categorical_accuracy did not improve from 0.26791
Epoch 11/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.6721 -
categorical_accuracy: 0.2762 - val_loss: 1.6738 - val_categorical_accuracy:
0.2796

Epoch 00011: val_categorical_accuracy improved from 0.26791 to 0.27960, savin
g model to weights.best.hdf5
Epoch 12/20
10240/10240 [==============================] - 22s 2ms/step - loss: 1.6674 -
categorical_accuracy: 0.2815 - val_loss: 1.6779 - val_categorical_accuracy:
0.2656

Epoch 00012: val_categorical_accuracy did not improve from 0.27960
Epoch 13/20
10240/10240 [==============================] - 22s 2ms/step - loss: 1.6610 -
categorical_accuracy: 0.2763 - val_loss: 1.6686 - val_categorical_accuracy:
0.2757

Epoch 00013: val_categorical_accuracy did not improve from 0.27960
Epoch 14/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.6558 -
categorical_accuracy: 0.2814 - val_loss: 1.6726 - val_categorical_accuracy:
0.2702

Epoch 00014: val_categorical_accuracy did not improve from 0.27960
Epoch 15/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.6509 -
```

```
categorical_accuracy: 0.2897 - val_loss: 1.6665 - val_categorical_accuracy:
0.2967

Epoch 00015: val_categorical_accuracy improved from 0.27960 to 0.29673, savin
g model to weights.best.hdf5
Epoch 16/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.6410 -
categorical_accuracy: 0.2884 - val_loss: 1.6707 - val_categorical_accuracy:
0.2687

Epoch 00016: val_categorical_accuracy did not improve from 0.29673
Epoch 17/20
10240/10240 [==============================] - 21s 2ms/step - loss: 1.6383 -
categorical_accuracy: 0.2948 - val_loss: 1.6676 - val_categorical_accuracy:
0.2882

Epoch 00017: val_categorical_accuracy did not improve from 0.29673
Epoch 00017: early stopping
```

In [33]:
```
## visualize RCNN

### visualize c-lstm model output
accu_curve=plt.figure()
plt.plot(history_RCNN.history['categorical_accuracy'],'r',linewidth=3.0)
plt.plot(history_RCNN.history['val_categorical_accuracy'],'b',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Accuracy',fontsize=12)
plt.title('Accuracy Curves : RCNN',fontsize=12)
# accu_curve.savefig('accuracy_clstm_improved_v1.4.4.png')
plt.show()
##^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##
loss_curve = plt.figure()
plt.plot(history_RCNN.history['loss'],'r',linewidth=3.0)
plt.plot(history_RCNN.history['val_loss'],'b',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=12)
plt.xlabel('Epochs ',fontsize=12)
plt.ylabel('Loss',fontsize=12)
plt.title('Loss Curves : RCNN',fontsize=12)
# loss_curve.savefig('loss_clstm_improved_v1.4.4.png')
loss_curve.show()
```





In [ ]: