text_classify

March 3, 2020

```
In [1]: # Importing all the needed modules.
        import os
        from glob import glob
        import matplotlib.pyplot as plt
        import random
        import cv2
        import pandas as pd
        import numpy as np
        import matplotlib.gridspec as gridspec
        import seaborn as sns
        import zlib
        import itertools
        import sklearn
        import itertools
        import scipy
        import skimage
        from skimage.transform import resize
        import csv
        from tqdm import tqdm
        import warnings
        warnings.filterwarnings("ignore")
        from sklearn import model_selection
        from sklearn.model_selection import train_test_split, KFold, cross_val_score, Stratifi
        from sklearn.utils import class_weight
        from sklearn.metrics import confusion_matrix, make_scorer, accuracy_score, classificati
        import keras
        from keras.layers import Embedding,Dense, Dropout, Activation, Flatten, Conv1D,Conv2D,M
        from keras.utils import np_utils
        from keras.utils.np_utils import to_categorical
        from keras.preprocessing.image import ImageDataGenerator
        from keras import models, layers, optimizers
        from keras.engine.input_layer import Input
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import confusion_matrix, accuracy_score
        from sklearn.utils import class_weight
        from keras.optimizers import SGD, RMSprop, Adam, Adagrad, Adadelta, RMSprop
        from keras.models import Sequential, model_from_json
        from keras.layers import Activation,Dense, Dropout, Flatten, Conv2D, MaxPool2D
        from keras.layers import MaxPooling2D,AveragePooling2D, GlobalAveragePooling2D,BatchNo
        from keras.preprocessing.image import array_to_img, img_to_array, load_img, ImageDataG
        from keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
        from keras import backend as K
        from keras.applications.vgg16 import VGG16
        from keras.models import Model
        from keras.applications.mobilenet import MobileNet
        from keras.applications.inception_v3 import InceptionV3
        from imblearn.over_sampling import RandomOverSampler
        from sklearn.metrics import roc_auc_score
        from sklearn.metrics import roc_curve
        from sklearn.metrics import auc
        %matplotlib inline
        import tensorflow as tf
        gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.555)

        sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))
```

```
Using TensorFlow backend.
```

1 Text Classification:

1.1 Data

1.1.1 Preprocessing:

2 Text pre-processing

```
In [2]: import glob
        import nltk
        import re
        from bs4 import BeautifulSoup

        files = glob.glob("documents/*")
```

```
In [3]: len(files)
```

```
Out[3]: 18828
```

```
In [4]: final = []; cread = 0; label = []; mlen = 0; idx = 0; subj = []; orgtxt = []
        for enm,doc in enumerate(files):
            try:
                f = open(doc,'r', encoding='utf8')
                content = f.read()


                soup = BeautifulSoup(content)
                text = soup.get_text()
                p1 = re.findall(r'[a-zA-Z0-9-]+[a-zA-Z0-9-\.]*@([a-zA-Z0-9-]+[a-zA-Z0-9-\.]*)'
                otxt = re.sub(r'[a-zA-Z0-9-]+[a-zA-Z0-9-\.]*@([a-zA-Z0-9-]+[a-zA-Z0-9-\.]*)',"
                em = ""
                for each in p1:
                    for word in each.split('.'):
                        if len(word) > 2 and word != 'com':
                            em += word
                            em += " "

                if len(em) > 2:
                    label.append(doc.split('/')[1].split('_')[0])
                    final.append(em[:-1].lower())
                    if len(em[:-1]) > mlen:
                        mlen = len(em)
                        idx = enm

                    txt = re.findall(r'Subject\:[ Re\:]*([A-Za-z0-9\:\(\)\! ]*)', text)
                    otxt = re.sub(r'Subject\:[ Re\:]*([A-Za-z0-9\:\(\)\! ]*)',"",otxt)
                    sub = ""
                    for each in txt:
                        sub += re.sub(r"[-()\"#/@;:<>{}`+=~|.!?,]", "",each)
                    subj.append(sub)
                    orgtxt.append(otxt)
            except:
                continue

        print(final[0],len(final),len(label),len(subj))

hpuerca atl 18380 18380 18380


In [5]: body = []
        for tmp in orgtxt:
            tmp = re.sub(r'From\:[ Re\:]*([A-Za-z0-9\:\(\)\!\. ]*)',"", tmp)
            tmp = re.sub(r'Write to\:[ Re\:]*([A-Za-z0-9\:\(\)\!\. ]*)',"", tmp)
            tmp = re.sub(r'\([A-Za-z0-9\:\!\.\-\,\;\'\`\n\t\?\/\>\<\$ ]*\)',"", tmp)
            tmp = re.sub(r'\<[A-Za-z0-9\:\!\.\-\,\;\'\`\n\t\?\/ ]*\>',"", tmp)
            tmp = re.sub(r'[\n\t\-\\\/\|]'," ",tmp)
            text = re.sub(r'[A-Za-z0-9]*\:',"",tmp)

            parse_tree = nltk.ne_chunk(nltk.tag.pos_tag(text.split()), binary=True)
            ctxt = ""
            chunk = list(parse_tree)

            for each in chunk:
                if isinstance(each, nltk.tree.Tree):
                    ctxt += '_'.join(k[0] for k in each)
                    ctxt += ' '
                else:
                    ctxt += each[0]+' '

            text = ctxt.lower()
            text = re.sub(r"i'm", "i am", text)
            text = re.sub(r"he's", "he is", text)
            text = re.sub(r"she's", "she is", text)
            text = re.sub(r"it's", "it is", text)
            text = re.sub(r"that's", "that is", text)
            text = re.sub(r"what's", "that is", text)
            text = re.sub(r"where's", "where is", text)
            text = re.sub(r"how's", "how is", text)
            text = re.sub(r"'s", " is", text)
            text = re.sub(r"\'ll", " will", text)
            text = re.sub(r"\'ve", " have", text)
            text = re.sub(r"\'re", " are", text)
            text = re.sub(r"\'d", " would", text)
            text = re.sub(r"\'re", " are", text)
            text = re.sub(r"won't", "will not", text)
            text = re.sub(r"can't", "can not", text)
            text = re.sub(r"n't", " not", text)
            text = re.sub(r"n'", "ng", text)
            text = re.sub(r"'bout", "about", text)
            text = re.sub(r"'til", "until", text)
            text = re.sub(r"[0-9]", "", text)
            text = re.sub(r" [a-zA-Z]\_", " ", text)
            text = re.sub(r" [a-zA-Z][a-zA-Z]\_", " ", text)
            text = re.sub(r"[-()\"#/@;:<>{}`+=~|.!?,$%^&*'/+\[\]]+", "", text)
            text = re.sub(r"\b[a-zA-Z]{1,2}\b", "", text)
            text = re.sub(r"\b[a-zA-Z]{15,}\b", "", text)
            text = re.sub(r' _'," ", text)
            text = re.sub(r'_ '," ", text)
            text = re.sub(r'__+'," ", text)
            text = re.sub(r" +", " ", text)
            body.append(text)

        len(body)

Out[5]: 18380

In [0]: #i am living in the New York
        print("i am living in the New York -->", list(chunks))
        print(" ")
        print("-"*50)
        print(" ")
        #My name is Srikanth Varma
        print("My name is Srikanth Varma -->", list(chunks1))
```

```
i am living in the New York --> [('i', 'NN'), ('am', 'VBP'), ('living', 'VBG'), ('in', 'IN'),

--------------------------------------------------

My name is Srikanth Varma --> [('My', 'PRP$'), ('name', 'NN'), ('is', 'VBZ'), Tree('PERSON', [
```

```
In [6]: len(final),len(subj),len(body),len(label)
```

```
Out[6]: (18380, 18380, 18380, 18380)
```

```
In [7]: data = []
        for a,b,c in zip(final,subj,body):
            data.append(a+" "+b.lower()+" "+c)
        len(data)
```

```
Out[7]: 18380
```

```
In [8]: data[1]
```

```
Out[8]: 'wmich edu lab wmich edu prk referral in canada could some please refer someone who ca
```

```
In [9]: import pandas as pd
        df = pd.DataFrame(data, columns = ['text'])
        df['label'] = label

        df.head()
```

```
Out[9]:                                                  text                 label
        0  hpuerca atl super mega automobile sightings ex...             rec.autos
        1  wmich edu lab wmich edu prk referral in canada...               sci.med
        2  netcom cats ucsc edu netcom help duo 230 probl...  comp.sys.mac.hardware
        3  ulkyvx louisville edu camelot bradley edu and ...     rec.sport.baseball
        4  gauss med harvard edu sol ctr columbia edu sta...       rec.sport.hockey
```

2.0.1 Training The models to Classify:

2.1 Data splitting

```
In [10]: from sklearn.model_selection import train_test_split

         X_train, X_test, y_train, y_test = train_test_split(df['text'], df['label'], test_siz
         print(len(X_train),len(X_test),len(y_train),len(y_test))
```

```
13785 4595 13785 4595
```

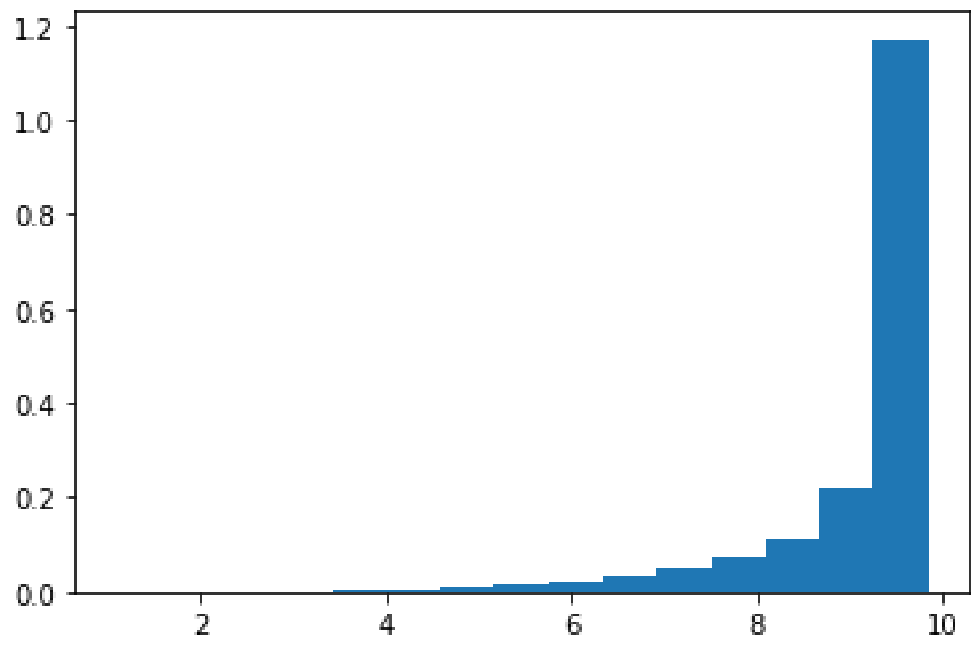
```
In [11]: from sklearn.feature_extraction.text import TfidfVectorizer

         vectorizer = TfidfVectorizer()
         X = vectorizer.fit_transform(X_train)
```

```
In [12]: import matplotlib.pyplot as plt
         import numpy as np
         %matplotlib inline

         plt.hist(vectorizer.idf_, normed=True, bins=15)

Out[12]: (array([1.79770060e-04, 3.26854654e-04, 6.37366575e-04, 1.20936222e-03,
                 2.35335351e-03, 4.52693696e-03, 8.64530559e-03, 1.41364638e-02,
                 2.22424592e-02, 3.27835218e-02, 4.94857946e-02, 7.36893817e-02,
                 1.11718921e-01, 2.18535022e-01, 1.17195370e+00]),
          array([1.07874996, 1.66271741, 2.24668486, 2.83065231, 3.41461975,
                 3.9985872 , 4.58255465, 5.1665221 , 5.75048955, 6.33445699,
                 6.91842444, 7.50239189, 8.08635934, 8.67032679, 9.25429423,
                 9.83826168]),
          <a list of 15 Patch objects>)
```



```
In [13]: voc = []
         for word,idx in vectorizer.vocabulary_.items():
             if vectorizer.idf_[idx] >= 9.3:
                 voc.append(word)


In [14]: len(voc)

Out[14]: 71711
```

```
In [15]: from tensorflow.keras.preprocessing.text import Tokenizer
         from tensorflow.keras.preprocessing.sequence import pad_sequences

         MAX_LEN = 1000

         tokenizer = Tokenizer()
         tokenizer.fit_on_texts(voc)
         seq_train = tokenizer.texts_to_sequences(X_train)
         seq_test = tokenizer.texts_to_sequences(X_test)
         dictionary = tokenizer.word_index

         word2idx = {}
         idx2word = {}
         for k, v in dictionary.items():
             word2idx[k] = v
             idx2word[v] = k

         input_data_train = pad_sequences(seq_train, maxlen=MAX_LEN, dtype='int32', padding='po
         input_data_test = pad_sequences(seq_test, maxlen=MAX_LEN, dtype='int32', padding='post

In [16]: import os
         import numpy as np
         def glove_100d_dictionary(GLOVE_DIR):
             embeddings_index = {}
             f = open(os.path.join(GLOVE_DIR, 'glove.6B.100d.txt'))
             for line in f:
                 values = line.split()
                 word = values[0]
                 coefs = np.asarray(values[1:], dtype='float32')
                 embeddings_index[word] = coefs
             f.close()
             return embeddings_index

         emd_ind = glove_100d_dictionary("GLOVE_DIR")

In [17]: len(word2idx)

Out[17]: 69410

In [18]: def embedding_matrix_creater(embedding_dimention):
             embedding_matrix = np.zeros((len(word2idx) + 1, embedding_dimention))
             for word, i in word2idx.items():
                 embedding_vector = emd_ind.get(word)
                 if embedding_vector is not None:
                   # words not found in embedding index will be all-zeros.
                     embedding_matrix[i] = embedding_vector
             return embedding_matrix
         emd_mat = embedding_matrix_creater(100)
         emd_mat.shape

Out[18]: (69411, 100)

In [19]: from sklearn import preprocessing
         import tensorflow as tf

         label_encoder = preprocessing.LabelEncoder()
         lnum_tr = label_encoder.fit_transform(y_train)
         lnum_te = label_encoder.transform(y_test)

         lab_train = tf.keras.utils.to_categorical(lnum_tr,dtype='float32')
         lab_test = tf.keras.utils.to_categorical(lnum_te,dtype='float32')
         len(lab_train),len(lab_test)

Out[19]: (13785, 4595)
```

2.2 Model

```
In [20]: VOCAB_SIZE = 69411
         EMBEDDING_DIM = 100
         MAX_LEN = 1000

In [21]: def conv_layer(filters,kernel):
             return Conv1D(filters,kernel,activation = 'relu',kernel_initializer="he_normal",ke

In [87]: encoder_inputs = Input(shape=(MAX_LEN, ), dtype='int32',)
         embedding_layer = Embedding(input_dim = VOCAB_SIZE, output_dim = EMBEDDING_DIM,input_
                                     weights = [emd_mat],trainable = False)(encoder_inputs)
         c1 = conv_layer(64,5)(embedding_layer)
         c2 = conv_layer(64,5)(embedding_layer)
         c3 = conv_layer(64,5)(embedding_layer)
         con = Concatenate()([c1,c2,c3])
         pool = MaxPooling1D(5)(con)

         c4 = conv_layer(16,5)(pool)
         c5 = conv_layer(64,5)(pool)
         c6 = conv_layer(32,5)(pool)
         con2 = Concatenate()([c4,c5,c6])
         pool2 = MaxPooling1D(5)(con2)

         c7 = conv_layer(16,5)(pool2)
         c8 = conv_layer(64,5)(pool2)
         c9 = conv_layer(32,5)(pool2)
         con3 = Concatenate()([c7,c8,c9])
         pool3 = MaxPooling1D(5)(con3)

         c7 = conv_layer(32,5)(pool2)
```

```
flat = Flatten()(c7)
drop = Dropout(0.3)(flat)
d1 = Dense(128,activation='tanh')(drop)
drop2 = Dropout(0.2)(d1)
out = Dense(20,activation='softmax')(d1)
model = Model(encoder_inputs,out)
model.summary()
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(0.001),metrics=['accuracy'])
```

```
______________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==============================================================================================
input_18 (InputLayer)           (None, 1000)         0

______________________________________________________________________________________________
embedding_18 (Embedding)        (None, 1000, 100)    6941100     input_18[0][0]

______________________________________________________________________________________________
conv1d_171 (Conv1D)             (None, 996, 64)      32064       embedding_18[0][0]

______________________________________________________________________________________________
conv1d_172 (Conv1D)             (None, 996, 64)      32064       embedding_18[0][0]

______________________________________________________________________________________________
conv1d_173 (Conv1D)             (None, 996, 64)      32064       embedding_18[0][0]

______________________________________________________________________________________________
concatenate_52 (Concatenate)    (None, 996, 192)     0           conv1d_171[0][0]
                                                                 conv1d_172[0][0]
                                                                 conv1d_173[0][0]

______________________________________________________________________________________________
max_pooling1d_52 (MaxPooling1D) (None, 199, 192)     0           concatenate_52[0][0]

______________________________________________________________________________________________
conv1d_174 (Conv1D)             (None, 195, 16)      15376       max_pooling1d_52[0][0]

______________________________________________________________________________________________
conv1d_175 (Conv1D)             (None, 195, 64)      61504       max_pooling1d_52[0][0]

______________________________________________________________________________________________
conv1d_176 (Conv1D)             (None, 195, 32)      30752       max_pooling1d_52[0][0]

______________________________________________________________________________________________
concatenate_53 (Concatenate)    (None, 195, 112)     0           conv1d_174[0][0]
                                                                 conv1d_175[0][0]
                                                                 conv1d_176[0][0]

______________________________________________________________________________________________
max_pooling1d_53 (MaxPooling1D) (None, 39, 112)      0           concatenate_53[0][0]

______________________________________________________________________________________________
conv1d_180 (Conv1D)             (None, 35, 32)       17952       max_pooling1d_53[0][0]

______________________________________________________________________________________________
flatten_18 (Flatten)            (None, 1120)         0           conv1d_180[0][0]

______________________________________________________________________________________________
dropout_35 (Dropout)            (None, 1120)         0           flatten_18[0][0]

______________________________________________________________________________________________
dense_35 (Dense)                (None, 128)          143488      dropout_35[0][0]
```

```
--------------------------------------------------------------------------------------------------
dense_36 (Dense)                (None, 20)           2580        dense_35[0][0]
==================================================================================================
Total params: 7,308,944
Trainable params: 367,844
Non-trainable params: 6,941,100
--------------------------------------------------------------------------------------------------
```

```
In [88]: filepath="weights_email.h5"
         checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, save_best_only=Tr
         callbacks_list = [checkpoint]
```

```
In [89]: class_weights = class_weight.compute_class_weight('balanced',
                                                           np.unique(lnum_tr),
                                                           lnum_tr)
```

```
In [90]: history = model.fit(x=input_data_train, y=lab_train, batch_size=32, epochs=30, verbose
                             validation_data=(input_data_test,lab_test),class_weight=class_weig
```

```
Train on 13785 samples, validate on 4595 samples
Epoch 1/30
13785/13785 [==============================] - 13s 962us/step - loss: 3.4490 - acc: 0.1719 - va

Epoch 00001: val_acc improved from -inf to 0.28901, saving model to weights_email.h5
Epoch 2/30
13785/13785 [==============================] - 10s 707us/step - loss: 2.0753 - acc: 0.3238 - va

Epoch 00002: val_acc improved from 0.28901 to 0.39391, saving model to weights_email.h5
Epoch 3/30
13785/13785 [==============================] - 8s 606us/step - loss: 1.7841 - acc: 0.4398 - val

Epoch 00003: val_acc improved from 0.39391 to 0.49402, saving model to weights_email.h5
Epoch 4/30
13785/13785 [==============================] - 10s 701us/step - loss: 1.6330 - acc: 0.5151 - va

Epoch 00004: val_acc improved from 0.49402 to 0.53754, saving model to weights_email.h5
Epoch 5/30
13785/13785 [==============================] - 8s 613us/step - loss: 1.5236 - acc: 0.5734 - val

Epoch 00005: val_acc improved from 0.53754 to 0.59217, saving model to weights_email.h5
Epoch 6/30
13785/13785 [==============================] - 9s 638us/step - loss: 1.4495 - acc: 0.6021 - val

Epoch 00006: val_acc improved from 0.59217 to 0.61545, saving model to weights_email.h5
Epoch 7/30
13785/13785 [==============================] - 9s 674us/step - loss: 1.3838 - acc: 0.6254 - val

Epoch 00007: val_acc improved from 0.61545 to 0.61828, saving model to weights_email.h5
Epoch 8/30
13785/13785 [==============================] - 8s 611us/step - loss: 1.3520 - acc: 0.6389 - val

Epoch 00008: val_acc improved from 0.61828 to 0.62568, saving model to weights_email.h5
Epoch 9/30
13785/13785 [==============================] - 10s 692us/step - loss: 1.3005 - acc: 0.6562 - va

Epoch 00009: val_acc improved from 0.62568 to 0.64831, saving model to weights_email.h5
Epoch 10/30
13785/13785 [==============================] - 8s 613us/step - loss: 1.2854 - acc: 0.6675 - val

Epoch 00010: val_acc improved from 0.64831 to 0.67116, saving model to weights_email.h5
Epoch 11/30
13785/13785 [==============================] - 10s 697us/step - loss: 1.2496 - acc: 0.6817 - va

Epoch 00011: val_acc did not improve from 0.67116
Epoch 12/30
13785/13785 [==============================] - 8s 600us/step - loss: 1.2048 - acc: 0.6955 - val

Epoch 00012: val_acc did not improve from 0.67116
Epoch 13/30
13785/13785 [==============================] - 8s 611us/step - loss: 1.1847 - acc: 0.7047 - val

Epoch 00013: val_acc did not improve from 0.67116
Epoch 14/30
13785/13785 [==============================] - 9s 689us/step - loss: 1.1627 - acc: 0.7128 - val

Epoch 00014: val_acc did not improve from 0.67116
Epoch 15/30
13785/13785 [==============================] - 8s 600us/step - loss: 1.1570 - acc: 0.7140 - val

Epoch 00015: val_acc improved from 0.67116 to 0.68727, saving model to weights_email.h5
Epoch 16/30
13785/13785 [==============================] - 10s 705us/step - loss: 1.1409 - acc: 0.7209 - va

Epoch 00016: val_acc did not improve from 0.68727
Epoch 17/30
13785/13785 [==============================] - 9s 622us/step - loss: 1.1213 - acc: 0.7288 - val

Epoch 00017: val_acc improved from 0.68727 to 0.69097, saving model to weights_email.h5
Epoch 18/30
13785/13785 [==============================] - 8s 606us/step - loss: 1.1094 - acc: 0.7370 - val

Epoch 00018: val_acc did not improve from 0.69097
Epoch 19/30
13785/13785 [==============================] - 10s 697us/step - loss: 1.0953 - acc: 0.7395 - va

Epoch 00019: val_acc did not improve from 0.69097
Epoch 20/30
13785/13785 [==============================] - 8s 614us/step - loss: 1.0983 - acc: 0.7445 - val

Epoch 00020: val_acc improved from 0.69097 to 0.70403, saving model to weights_email.h5
Epoch 21/30
13785/13785 [==============================] - 10s 701us/step - loss: 1.0611 - acc: 0.7552 - va

Epoch 00021: val_acc did not improve from 0.70403
Epoch 22/30
13785/13785 [==============================] - 8s 602us/step - loss: 1.0524 - acc: 0.7542 - val

Epoch 00022: val_acc did not improve from 0.70403
Epoch 23/30
13785/13785 [==============================] - 8s 609us/step - loss: 1.0531 - acc: 0.7566 - val

Epoch 00023: val_acc did not improve from 0.70403
Epoch 24/30
13785/13785 [==============================] - 10s 694us/step - loss: 1.0417 - acc: 0.7622 - va

Epoch 00024: val_acc improved from 0.70403 to 0.70773, saving model to weights_email.h5
Epoch 25/30
13785/13785 [==============================] - 8s 607us/step - loss: 1.0331 - acc: 0.7652 - val

Epoch 00025: val_acc did not improve from 0.70773
Epoch 26/30
13785/13785 [==============================] - 10s 692us/step - loss: 1.0315 - acc: 0.7713 - va

Epoch 00026: val_acc did not improve from 0.70773
Epoch 27/30
13785/13785 [==============================] - 8s 601us/step - loss: 1.0247 - acc: 0.7727 - val

Epoch 00027: val_acc improved from 0.70773 to 0.72165, saving model to weights_email.h5
Epoch 28/30
13785/13785 [==============================] - 8s 615us/step - loss: 1.0210 - acc: 0.7756 - val

Epoch 00028: val_acc did not improve from 0.72165
Epoch 29/30
13785/13785 [==============================] - 10s 694us/step - loss: 1.0088 - acc: 0.7782 - va

Epoch 00029: val_acc did not improve from 0.72165
Epoch 30/30
13785/13785 [==============================] - 9s 617us/step - loss: 0.9932 - acc: 0.7854 - val

Epoch 00030: val_acc did not improve from 0.72165
```

```
In [91]: from keras.utils.vis_utils import plot_model
         plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

2.2.1 Go through this blog, if you have any doubt on using predefined Embedding values in Embedding layer - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

ref: 'https://i.imgur.com/fv1GvFJ.png'