In [1]:
```
from __future__ import print_function

from sklearn.preprocessing import OneHotEncoder
from keras.layers.core import Dense, Activation, Dropout
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM
from keras.datasets import imdb
import pandas as pd
import numpy as np
import os
```

```
Using TensorFlow backend.
/Users/DuaaTashkandi/anaconda/lib/python3.6/importlib/_bootstrap.py:205: RuntimeWarning: compiletime version 3.5 of module 'tensorflow.python.framework.fast_tensor_util' does not match runtime version 3.6
  return f(*args, **kwds)
```

In [2]:
```
#parameters
maxlen = 30 # FIXME
labels = 2
```

In [3]:
```
input = pd.read_csv("data_new/merged.csv",header=None)
input.columns = ['first', 'last','b_or_n']

# remove encode
input['first'] = input['first'].str[2:]
input['first'] = input['first'].str[:-1]
input['last'] = input['last'].str[2:]
input['last'] = input['last'].str[:-1]

#print(type(b'usdgfuia'.decode('utf-8')))

input['firstlen']= [len(str(i)) for i in input['first']]
input1 = input[(input['firstlen'] >= 2) ] #FIXME
print(input1)
```

	first	last	b_or_n	firstlen
0	ithalohfonseca	chaverinho	1	14
1	gustavo	gon\xc3\xa7alves	1	7
2	rafael	geraldo	1	6
3	anton	prasetyo	1	5
4	cristyan	victor	1	8
5	valsemorborgesdesouza	neto	1	21
6	diegodereknobre	nobre	1	15
7	paulonatan	santos	1	10
8	jardelurgalde	oliveira	1	13

9	deolindo	barbosa	1	8
10	gabrielbitencourt	figueredo	1	17
11	eduardo	carvalho	1	7
12	matheus	rct	1	7
13	fernando	cesar	1	8
14	gustavo	batalha	1	7
15	muhamad	riyandi	1	7
16	oohtalldomatheus	pjl	1	16
17	gabriel	ara\xc3\xbajo	1	7
18	raphael	rocha	1	7
19	thiago	urameshi	1	6
20	juliandra	bimantara	1	9
21	gustavo	araujo	1	7
22	guilherme	francco	1	9
23	jefersonhugo	ribeiro	1	12
24	allan	victor	1	5
25	gabriel	sousa	1	7
26	willian	oliveira	1	7
27	welington	junior	1	9
28	eliasdaniel	amador	1	11
29	ruben	wesley	1	5
...	...	...	...	...
59970	casey	hewett	0	5
59971	karan	mirchandani	0	5
59972	angela	chin	0	6
59973	max	reichard	0	3
59974	rachel	raasch	0	6
59975	jenny	montoya	0	5
59976	kyle	satterfield	0	4
59977	mia	shang	0	3
59978	brianne	vasarhelyi	0	7
59979	william	su	0	7
59980	paige	koning	0	5
59981	mitchell	stevenson	0	8
59982	gretchen	blankinship	0	8
59983	kashish	patel	0	7
59984	rohan	arora	0	5
59985	elly	leidner	0	4
59986	natasha	townsend	0	7
59987	kat	early	0	3
59988	chapel	fowler	0	6
59989	lindsey	pope	0	7
59990	tony	bird	0	4
59991	bharat	modi	0	6
59992	jenny	li	0	5
59993	sharon	navarrete	0	6
59994	sam	lowe	0	3
59995	kevin	zheng	0	5
59996	manjil	thapa	0	6
59997	david	dominic	0	5

```
59998              katherine               ross       0         9
59999          garrettjoseph             bolter       0        13

[59996 rows x 4 columns]
```

In [4]:
```
input1.groupby('b_or_n')['first'].count() #FIXMEBEN
```

Out[4]:
```
b_or_n
0    30000
1    29996
Name: first, dtype: int64
```

In [5]:
```
firsts = input['first']
#print(firsts)
labels = input['b_or_n']
vocab = set(' '.join([str(i) for i in firsts]))
vocab.add('END')
len_vocab = len(vocab)
```

In [6]:
```
print(vocab)
print("vocab length is ",len_vocab)
print ("length of input is ",len(input1))
```

```
{'6', 'j', 'END', 'i', '9', 'e', '4', '7', 'y', 'v', '5', '\\', 'u', '
c', 'q', 'd', 'b', 'm', 't', 'w', 'o', 'f', 'h', 'p', '1', '8', 'g', '
r', '3', 'l', 'a', '2', 'k', 's', '0', 'x', ' ', 'n', 'z'}
vocab length is  39
length of input is  59996
```

In [7]:
```
char_index = dict((c, i) for i, c in enumerate(vocab))
```

In [8]:
```
print(char_index)
```

```
{'6': 0, 'j': 1, 'END': 2, 'i': 3, '9': 4, 'e': 5, '4': 6, '7': 7, 'y'
: 8, 'v': 9, '5': 10, '\\': 11, 'u': 12, 'c': 13, 'q': 14, 'd': 15, 'b
': 16, 'm': 17, 't': 18, 'w': 19, 'o': 20, 'f': 21, 'h': 22, 'p': 23,
'1': 24, '8': 25, 'g': 26, 'r': 27, '3': 28, 'l': 29, 'a': 30, '2': 31
, 'k': 32, 's': 33, '0': 34, 'x': 35, ' ': 36, 'n': 37, 'z': 38}
```

In [9]:
```
#train test split
msk = np.random.rand(len(input1)) < 0.8

# we dont want to split
#msk = [True]*len(input1)

train = input1[msk]
test = input1[~msk]

print(train)
```

```
print(train)
```

	first	last	b_or_n	firstlen
0	ithalohfonseca	chaverinho	1	14
2	rafael	geraldo	1	6
3	anton	prasetyo	1	5
4	cristyan	victor	1	8
5	valsemorborgesdesouza	neto	1	21
6	diegodereknobre	nobre	1	15
7	paulonatan	santos	1	10
8	jardelurgalde	oliveira	1	13
9	deolindo	barbosa	1	8
10	gabrielbitencourt	figueredo	1	17
11	eduardo	carvalho	1	7
12	matheus	rct	1	7
13	fernando	cesar	1	8
15	muhamad	riyandi	1	7
16	oohtalldomatheus	pjl	1	16
17	gabriel	ara\xc3\xbajo	1	7
18	raphael	rocha	1	7
20	juliandra	bimantara	1	9
21	gustavo	araujo	1	7
22	guilherme	francco	1	9
23	jefersonhugo	ribeiro	1	12
24	allan	victor	1	5
25	gabriel	sousa	1	7
26	willian	oliveira	1	7
27	welington	junior	1	9
28	eliasdaniel	amador	1	11
29	ruben	wesley	1	5
30	hazard	id	1	6
31	lucas	emanuel	1	5
32	pedro	lucas	1	5
...	...	...	...	...
59962	susanna	azzoni	0	7
59966	josh	boone	0	4
59967	tamikiamccullers	mcneill	0	16
59968	ronya	wehbe	0	5
59969	patton	orr	0	6
59970	casey	hewett	0	5
59972	angela	chin	0	6
59973	max	reichard	0	3
59974	rachel	raasch	0	6
59975	jenny	montoya	0	5
59976	kyle	satterfield	0	4
59977	mia	shang	0	3
59978	brianne	vasarhelyi	0	7
59979	william	su	0	7
59980	paige	koning	0	5
59981	mitchell	stevenson	0	8
59982	gretchen	blankinship	0	8

```
59983                kashish          patel       0         7
59985                   elly        leidner       0         4
59986                natasha       townsend       0         7
59989                lindsey           pope       0         7
59990                   tony           bird       0         4
59991                 bharat           modi       0         6
59992                  jenny             li       0         5
59994                    sam           lowe       0         3
59995                  kevin          zheng       0         5
59996                 manjil          thapa       0         6
59997                  david        dominic       0         5
59998              katherine           ross       0         9
59999          garrettjoseph         bolter       0        13

[47976 rows x 4 columns]
```

In [10]:
```
# take input upto max and truncate rest
# encode to vector space(one hot encoding)
# padd 'END' to shorter sequences
train_X = []
trunc_train_first = [str(i)[0:30] for i in train['first']]
for i in trunc_train_first:
    tmp = [char_index[j] for j in str(i)]
    for k in range(0,maxlen - len(str(i))):
        tmp.append(char_index["END"])
    train_X.append(tmp)
```

In [ ]:
```

```

In [11]:
```
np.asarray(train_X).shape
```

Out[11]:
```
(47976, 30)
```

In [12]:
```
def set_flag(i):
    tmp = np.zeros(39);
    tmp[i] = 1
    return(tmp)
```

In [13]:
```
set_flag(3)
```

Out[13]:
```
array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.
,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.
,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.
])
```

modify the code above to also convert each index to one-hot encoded

modify the code above to also convert each index to one-hot encoded representation

In [14]:
```
#take input upto max and truncate rest
#encode to vector space(one hot encoding)
#padd 'END' to shorter sequences
#also convert each index to one-hot encoding
train_X = []
train_Y = []
trunc_train_first = [str(i)[0:maxlen] for i in train['first']]
for i in trunc_train_first:
    tmp = [set_flag(char_index[j]) for j in str(i)]
    for k in range(0,maxlen - len(str(i))):
        tmp.append(set_flag(char_index["END"]))
    train_X.append(tmp)
for i in train['b_or_n']:
    if i == 1:
        train_Y.append([1,0])
    else:
        train_Y.append([0,1])
```

In [15]:
```
np.asarray(train_X).shape
```

Out[15]: (47976, 30, 39)

In [16]:
```
np.asarray(train_Y).shape
```

Out[16]: (47976, 2)

build model in keras (a stacked LSTM model with many-to-one arch) here 30 sequence and 2 output each for one category(m/f)

In [17]:
```
#build the model: 2 stacked LSTM
print('Build model...')
model = Sequential()
model.add(LSTM(512, return_sequences=True, input_shape=(maxlen,len_vocab)
model.add(Dropout(0.2))
model.add(LSTM(512, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(2))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=[
```

```
Build model...
```

In [18]:
```
test_X = []
test_Y = []
trunc_test_first = [str(i)[0:maxlen] for i in test['first']]
for i in trunc_test_first:
    tmp = [set_flag(char_index[j]) for j in str(i)]
    for k in range(0,maxlen - len(str(i))):
        tmp.append(set_flag(char_index["END"]))
    test_X.append(tmp)
for i in test['b_or_n']:
    if i == 1:
        test_Y.append([1,0])
    else:
        test_Y.append([0,1])
```

In [19]:
```
print(np.asarray(test_X).shape)
print(np.asarray(test_Y).shape)
```

```
(12020, 30, 39)
(12020, 2)
```

In [20]:
```
batch_size=1000
model.fit(train_X, train_Y,batch_size=batch_size,nb_epoch=10,validation_d
```

```
/Users/DuaaTashkandi/anaconda/lib/python3.6/site-packages/keras/models
.py:939: UserWarning: The `nb_epoch` argument in `fit` has been rename
d `epochs`.
  warnings.warn('The `nb_epoch` argument in `fit` '

Train on 47976 samples, validate on 12020 samples
Epoch 1/10
47976/47976 [==============================] - 811s 17ms/step - loss:
0.6145 - acc: 0.6522 - val_loss: 0.5507 - val_acc: 0.7389
Epoch 2/10
47976/47976 [==============================] - 811s 17ms/step - loss:
0.5427 - acc: 0.7466 - val_loss: 0.5277 - val_acc: 0.7666
Epoch 3/10
47976/47976 [==============================] - 807s 17ms/step - loss:
0.5099 - acc: 0.7673 - val_loss: 0.4818 - val_acc: 0.7822
Epoch 4/10
47976/47976 [==============================] - 813s 17ms/step - loss:
0.4797 - acc: 0.7822 - val_loss: 0.4658 - val_acc: 0.7928
Epoch 5/10
47976/47976 [==============================] - 814s 17ms/step - loss:
0.4644 - acc: 0.7903 - val_loss: 0.4633 - val_acc: 0.7924
Epoch 6/10
47976/47976 [==============================] - 812s 17ms/step - loss:
0.4487 - acc: 0.8000 - val_loss: 0.4431 - val_acc: 0.8027
Epoch 7/10
47976/47976 [==============================] - 820s 17ms/step - loss:
0.4424 - acc: 0.8030 - val_loss: 0.4356 - val_acc: 0.8072
Epoch 8/10
47976/47976 [==============================] - 813s 17ms/step - loss:
0.4284 - acc: 0.8111 - val_loss: 0.4259 - val_acc: 0.8121
Epoch 9/10
47976/47976 [==============================] - 815s 17ms/step - loss:
0.4114 - acc: 0.8203 - val_loss: 0.4148 - val_acc: 0.8188
Epoch 10/10
47976/47976 [==============================] - 818s 17ms/step - loss:
0.4045 - acc: 0.8220 - val_loss: 0.4077 - val_acc: 0.8282
```

Out[20]:
```
<keras.callbacks.History at 0x14e9e17f0>
```

In [21]:
```
score, acc = model.evaluate(test_X, test_Y)
print('Test score:', score)
print('Test accuracy:', acc)
```

```
12020/12020 [==============================] - 92s 8ms/step
Test score: 0.407679110275
Test accuracy: 0.828202994969
```

In [24]:
```
name = ["sandhya","jaspreet","rajesh"]
#name = ["Aurélia", "Emma", "Gabriela", "Beatriz", "Olivia", "João", "Igo
X=[]
trunc_name = [i[0:maxlen] for i in name]
for i in trunc_name:
    tmp = [set_flag(char_index[j]) for j in str(i)]
    for k in range(0,maxlen - len(str(i))):
        tmp.append(set_flag(char_index["END"]))
    X.append(tmp)
pred=model.predict(np.asarray(X))
```

In [25]:
```
pred
```

Out[25]:
```
array([[ 0.05398228,  0.94601774],
       [ 0.04582489,  0.95417517],
       [ 0.08217151,  0.91782844]], dtype=float32)
```

Lets train more, clearly some very simple female names it doesnt get right like mentioned above (inspite it exists in training data)

In [26]:
```
batch_size=1000
model.fit(train_X, train_Y,batch_size=batch_size,nb_epoch=50,validation_d
```

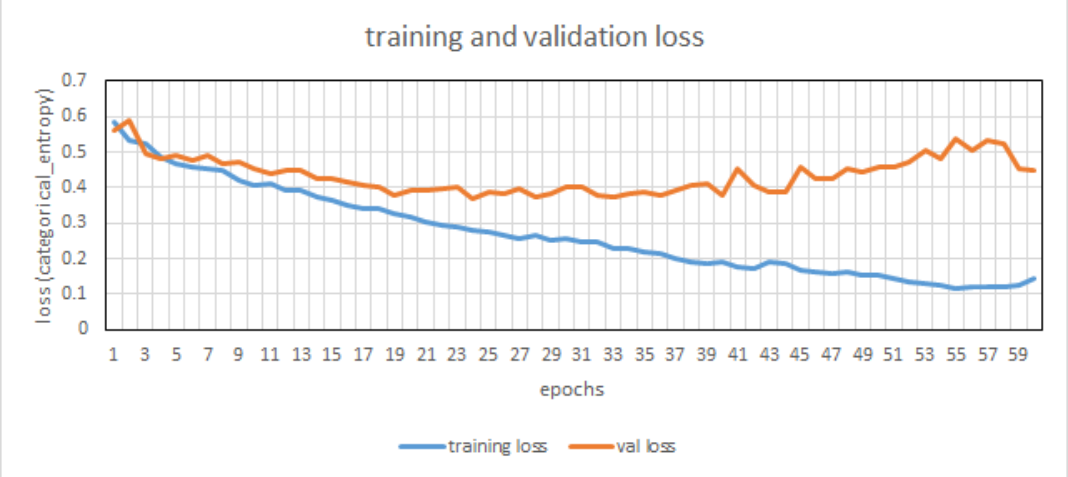
```
0.1949 - acc: 0.9209 - val_loss: 0.3870 - val_acc: 0.8676
Epoch 45/50
47976/47976 [==============================] - 821s 17ms/step - loss:
0.1975 - acc: 0.9210 - val_loss: 0.4133 - val_acc: 0.8608
Epoch 46/50
47976/47976 [==============================] - 827s 17ms/step - loss:
0.1896 - acc: 0.9245 - val_loss: 0.3926 - val_acc: 0.8694
Epoch 47/50
47976/47976 [==============================] - 829s 17ms/step - loss:
0.1887 - acc: 0.9241 - val_loss: 0.4162 - val_acc: 0.8651
Epoch 48/50
47976/47976 [==============================] - 825s 17ms/step - loss:
0.1870 - acc: 0.9245 - val_loss: 0.4058 - val_acc: 0.8647
Epoch 49/50
47976/47976 [==============================] - 826s 17ms/step - loss:
0.1823 - acc: 0.9272 - val_loss: 0.4186 - val_acc: 0.8643
Epoch 50/50
47976/47976 [==============================] - 825s 17ms/step - loss:
0.1785 - acc: 0.9280 - val_loss: 0.4159 - val_acc: 0.8645
```

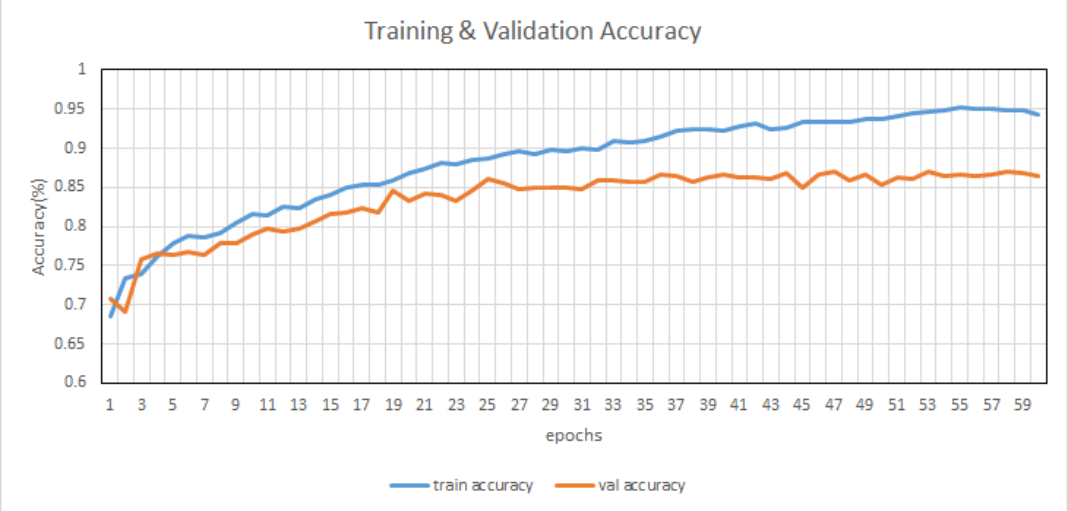
Out[26]:

In [27]:
```
score, acc = model.evaluate(test_X, test_Y)
print('Test score:', score)
print('Test accuracy:', acc)
```

```
12020/12020 [==============================] - 91s 8ms/step
Test score: 0.415939519558
Test accuracy: 0.864475873544
```

lets look at the loss and accuracy chart as a function of epochs





In [36]:
```
# name=["sandhya","jaspreet","rajesh","kaveri","aditi deepak","arihant","
# X=[]
# trunc_name = [i[0:maxlen] for i in name]
# for i in trunc_name:
#     tmp = [set_flag(char_index[j]) for j in str(i)]
#     for k in range(0,maxlen - len(str(i))):
#         tmp.append(set_flag(char_index["END"]))
#     X.append(tmp)
# pred=model.predict(np.asarray(X))
# pred
```

Out[36]:
```
array([[  7.85566587e-03,   9.92144346e-01],
       [  5.72708528e-03,   9.94272888e-01],
       [  8.59012544e-01,   1.40987411e-01],
       [  9.32210311e-03,   9.90677953e-01],
       [  9.09995317e-01,   9.00047421e-02],
       [  6.67434011e-04,   9.99332488e-01],
       [  8.42063571e-04,   9.99157906e-01],
       [  3.66371567e-03,   9.96336341e-01],
       [  9.95979428e-01,   4.02050652e-03]], dtype=float32)
```

In [37]:
```
# name=["abhi","abhi deepak","mr. abhi"]
# X=[]
# trunc_name = [i[0:maxlen] for i in name]
# for i in trunc_name:
#     tmp = [set_flag(char_index[j]) for j in str(i)]
#     for k in range(0,maxlen - len(str(i))):
#         tmp.append(set_flag(char_index["END"]))
#     X.append(tmp)
# pred=model.predict(np.asarray(X))
# pred
```

```
---------------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
<ipython-input-37-d33a696849b9> in <module>()
      3 trunc_name = [i[0:maxlen] for i in name]
      4 for i in trunc_name:
----> 5     tmp = [set_flag(char_index[j]) for j in str(i)]
      6     for k in range(0,maxlen - len(str(i))):
      7         tmp.append(set_flag(char_index["END"]))

<ipython-input-37-d33a696849b9> in <listcomp>(.0)
      3 trunc_name = [i[0:maxlen] for i in name]
      4 for i in trunc_name:
----> 5     tmp = [set_flag(char_index[j]) for j in str(i)]
      6     for k in range(0,maxlen - len(str(i))):
      7         tmp.append(set_flag(char_index["END"]))

KeyError: '.'
```

In [38]:
```
# name=["rajini","rajinikanth","mr. rajini"]
# X=[]
# trunc_name = [i[0:maxlen] for i in name]
# for i in trunc_name:
#     tmp = [set_flag(char_index[j]) for j in str(i)]
#     for k in range(0,maxlen - len(str(i))):
#         tmp.append(set_flag(char_index["END"]))
#     X.append(tmp)
# pred=model.predict(np.asarray(X))
# pred
```

```
---------------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
<ipython-input-38-e23256134d24> in <module>()
      3 trunc_name = [i[0:maxlen] for i in name]
      4 for i in trunc_name:
----> 5     tmp = [set_flag(char_index[j]) for j in str(i)]
      6     for k in range(0,maxlen - len(str(i))):
      7         tmp.append(set_flag(char_index["END"]))

<ipython-input-38-e23256134d24> in <listcomp>(.0)
      3 trunc_name = [i[0:maxlen] for i in name]
      4 for i in trunc_name:
----> 5     tmp = [set_flag(char_index[j]) for j in str(i)]
      6     for k in range(0,maxlen - len(str(i))):
      7         tmp.append(set_flag(char_index["END"]))

KeyError: '.'
```

In [39]:
```
#save our model and data
model.save_weights('gender_model',overwrite=True)
train.to_csv("train_split.csv")
test.to_csv("test_split.csv")
```

In [40]:
```
evals = model.predict(test_X)
prob_m = [i[0] for i in evals]
```

In [48]:
```
out = pd.DataFrame(prob_m)
out['first'] = test['first'].reset_index()['first']
out['b_or_n']=test.b_or_n.reset_index()['b_or_n']
```

In [50]:
```
out.head(10)
out.columns = ['prob_m','name','actual']
out.head(10)
out.to_csv("pred_out.csv")
```

In [ ]:
```

```