In [6]:
```
import nltk
nltk.download('words')
```

```
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\Atta\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\words.zip.
```

Out[6]: True

In [9]:
```
pip install plotly
```

```
Collecting plotly
  Downloading https://files.pythonhosted.org/packages/70/19/8437e22c84083a6d5
d8a3c80f4edc73c9dcbb89261d07e6bd13b48752bbd/plotly-4.1.1-py2.py3-none-any.whl
(7.1MB)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packa
ges (from plotly) (1.12.0)
Collecting retrying>=1.3.3 (from plotly)
  Downloading https://files.pythonhosted.org/packages/44/ef/beae4b4ef80902f22
e3af073397f079c96969c69b2c7d52a57ea9ae61c9d/retrying-1.3.3.tar.gz
Building wheels for collected packages: retrying
  Building wheel for retrying (setup.py): started
  Building wheel for retrying (setup.py): finished with status 'done'
  Stored in directory: C:\Users\Atta\AppData\Local\pip\Cache\wheels\d7\a9\33
\acc7b709e2a35caa7d4cae442f6fe6fbf2c43f80823d46460c
Successfully built retrying
Installing collected packages: retrying, plotly
Successfully installed plotly-4.1.1 retrying-1.3.3
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
import pandas as pd
import numpy as np
import re
import os
from IPython.display import HTML

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction import text
from sklearn.decomposition import PCA

from tensorflow.python.keras.models import Sequential, load_model
from tensorflow.python.keras.layers import Dense, Dropout
from tensorflow.python.keras import optimizers

import nltk
from nltk.stem.porter import PorterStemmer
from nltk.stem.snowball import SnowballStemmer
from nltk.corpus import words
from nltk.corpus import wordnet
allEnglishWords = words.words() + [w for w in wordnet.words()]
allEnglishWords = np.unique([x.lower() for x in allEnglishWords])

import plotly.offline as py
import plotly.graph_objs as go
py.init_notebook_mode(connected=True)

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import os
path = "D:/Deeplearning/datasets/imdb-movie-reviews-dataset/aclImdb/"
positiveFiles = [x for x in os.listdir(path+"train/pos/") if x.endswith(".txt"
)]
negativeFiles = [x for x in os.listdir(path+"train/neg/") if x.endswith(".txt"
)]
testFiles = [x for x in os.listdir(path+"test/") if x.endswith(".txt")]
```

In [3]:
```
positiveReviews, negativeReviews, testReviews = [], [], []
for pfile in positiveFiles:
    with open(path+"train/pos/"+pfile, encoding="latin1") as f:
        positiveReviews.append(f.read())
for nfile in negativeFiles:
    with open(path+"train/neg/"+nfile, encoding="latin1") as f:
        negativeReviews.append(f.read())
for tfile in testFiles:
    with open(path+"test/"+tfile, encoding="latin1") as f:
        testReviews.append(f.read())
```

In [4]:
```
reviews = pd.concat([
    pd.DataFrame({"review":positiveReviews, "label":1, "file":positiveFiles}),
    pd.DataFrame({"review":negativeReviews, "label":0, "file":negativeFiles}),
    pd.DataFrame({"review":testReviews, "label":-1, "file":testFiles})
], ignore_index=True).sample(frac=1, random_state=1)
reviews.head()
```

Out[4]:

	review	label	file
21939	Gwyneth Paltrow is absolutely great in this mo...	0	7246_4.txt
24113	I own this movie. Not by choice, I do. I was r...	0	9202_1.txt
4633	Well I guess it supposedly not a classic becau...	1	2920_8.txt
17240	I am, as many are, a fan of Tony Scott films. ...	0	3016_1.txt
4894	I wish "that '70s show" would come back on tel...	1	3155_10.txt

In [5]:
```
reviews = reviews[["review", "label", "file"]].sample(frac=1, random_state=1)
train = reviews[reviews.label!=-1].sample(frac=0.6, random_state=1)
valid = reviews[reviews.label!=-1].drop(train.index)
test = reviews[reviews.label==-1]
```

In [6]:
```
print(train.shape)
print(valid.shape)
print(test.shape)
```

```
(15000, 3)
(10000, 3)
(2, 3)
```

In [7]:
```
HTML(train.review.iloc[0])
```

Out[7]: One word can describe this movie and that is weird. I recorded this movie one day because it was a Japanese animation and it was old so I thought it would be interesting. Well it was, the movie is about a young boy who travels the universe to get a metal body so he can seek revenge. On the way he meets very colorful characters and must ultimately decide if he wants the body or not. Very strange, if you are a fan of animation/science-fiction you might want to check this out.

In [8]:
```
class Preprocessor(object):
    ''' Preprocess data for NLP tasks. '''

    def __init__(self, alpha=True, lower=True, stemmer=True, english=False):
        self.alpha = alpha
        self.lower = lower
        self.stemmer = stemmer
        self.english = english

        self.uniqueWords = None
        self.uniqueStems = None

    def fit(self, texts):
        texts = self._doAlways(texts)

        allwords = pd.DataFrame({"word": np.concatenate(texts.apply(lambda x:
x.split()).values)})
        self.uniqueWords = allwords.groupby(["word"]).size().rename("count").r
eset_index()
        self.uniqueWords = self.uniqueWords[self.uniqueWords["count"]>1]
        if self.stemmer:
            self.uniqueWords["stem"] = self.uniqueWords.word.apply(lambda x: P
orterStemmer().stem(x)).values
            self.uniqueWords.sort_values(["stem", "count"], inplace=True, asce
nding=False)
            self.uniqueStems = self.uniqueWords.groupby("stem").first()

        #if self.english: self.words["english"] = np.in1d(self.words["mode"],
 allEnglishWords)
        print("Fitted.")

    def transform(self, texts):
        texts = self._doAlways(texts)
        if self.stemmer:
            allwords = np.concatenate(texts.apply(lambda x: x.split()).values)
            uniqueWords = pd.DataFrame(index=np.unique(allwords))
            uniqueWords["stem"] = pd.Series(uniqueWords.index).apply(lambda x:
PorterStemmer().stem(x)).values
            uniqueWords["mode"] = uniqueWords.stem.apply(lambda x: self.unique
Stems.loc[x, "word"] if x in self.uniqueStems.index else "")
            texts = texts.apply(lambda x: " ".join([uniqueWords.loc[y, "mode"]
for y in x.split()]))
        #if self.english: texts = self.words.apply(lambda x: " ".join([y for y
in x.split() if self.words.loc[y,"english"]]))
        print("Transformed.")
        return(texts)

    def fit_transform(self, texts):
        texts = self._doAlways(texts)
        self.fit(texts)
        texts = self.transform(texts)
        return(texts)

    def _doAlways(self, texts):
        # Remove parts between <>'s
        texts = texts.apply(lambda x: re.sub('<.*?>', ' ', x))
```

```
        # Keep letters and digits only.
        if self.alpha: texts = texts.apply(lambda x: re.sub('[^a-zA-Z0-9 ]+',
' ', x))
        # Set everything to lower case
        if self.lower: texts = texts.apply(lambda x: x.lower())
        return texts
```

In [9]:
```
train.head()
```

Out[9]:

	review	label	file
6011	One word can describe this movie and that is w...	1	4160_9.txt
9653	The Ancient Mariner is a truly classic piece o...	1	7439_9.txt
15040	The late 80's saw an inexplicable rash of supe...	0	12287_3.txt
6029	A delightful piece of cinema storytelling in a...	1	4177_9.txt
9729	Greetings again from the darkness. Mary Heron ...	1	7507_8.txt

In [10]:
```
preprocess = Preprocessor(alpha=True, lower=True, stemmer=True)
```

In [11]:
```
%%time
trainX = preprocess.fit_transform(train.review)
validX = preprocess.transform(valid.review)
```

```
Fitted.
Transformed.
Transformed.
Wall time: 2min 32s
```

In [12]:
```
trainX.head()
```

Out[12]:
```
6011     one word can describe this movie and that is w...
9653     the ancient marine is a truly classic piece of...
15040    the late 80 s saw an inexplicable rash of supe...
6029     a delightful piece of cinema storytelling in a...
9729     greetings again from the dark mary  is amassed...
Name: review, dtype: object
```

In [21]:
```
print(preprocess.uniqueWords.shape)
preprocess.uniqueWords[preprocess.uniqueWords.word.str.contains("disapp")]
```

```
(38123, 3)
```

Out[21]:

	word	count	stem
15094	disapproving	6	disapprov
15093	disapproves	5	disapprov
15091	disapproval	3	disapprov
15087	disappointingly	12	disappointingli
15085	disappointed	569	disappoint
15086	disappointing	260	disappoint
15088	disappointment	247	disappoint
15084	disappoint	58	disappoint
15090	disappoints	21	disappoint
15089	disappointments	13	disappoint
15081	disappeared	61	disappear
15078	disappear	58	disappear
15083	disappears	44	disappear
15079	disappearance	21	disappear
15082	disappearing	17	disappear
15080	disappearances	4	disappear

In [22]:
```
print(preprocess.uniqueStems.shape)
preprocess.uniqueStems[preprocess.uniqueStems.word.str.contains("disappoint")]
```

```
(25274, 2)
```

Out[22]:

	word	count
stem		
disappoint	disappointed	569
disappointingli	disappointingly	12

In [23]:
```
stop_words = text.ENGLISH_STOP_WORDS.union(["thats","weve","dont","lets","your
e","im","thi","ha",
    "wa","st","ask","want","like","thank","know","susan","ryan","say","got","o
ught","ive","theyre"])
tfidf = TfidfVectorizer(min_df=2, max_features=10000, stop_words=stop_words)
#, ngram_range=(1,3)
```

In [24]:
```
%%time
trainX = tfidf.fit_transform(trainX).toarray()
validX = tfidf.transform(validX).toarray()
```

```
Wall time: 10.9 s
```

In [25]:
```
print(trainX.shape)
print(validX.shape)
```

```
(15000, 10000)
(10000, 10000)
```

In [26]:
```
trainY = train.label
validY = valid.label
```

In [27]:
```
print(trainX.shape, trainY.shape)
print(validX.shape, validY.shape)
```

```
(15000, 10000) (15000,)
(10000, 10000) (10000,)
```

In [28]:
```
from scipy.stats.stats import pearsonr
```

In [29]:
```
getCorrelation = np.vectorize(lambda x: pearsonr(trainX[:,x], trainY)[0])
correlations = getCorrelation(np.arange(trainX.shape[1]))
print(correlations)
```

```
[-0.01133404 -0.02084958  0.01552458 ...  0.02185732  0.00800117
 -0.00226902]
```

In [30]:
```
allIndeces = np.argsort(-correlations)
bestIndeces = allIndeces[np.concatenate([np.arange(1000), np.arange(-1000, 0
)])]
```

In [31]:
```
vocabulary = np.array(tfidf.get_feature_names())
print(vocabulary[bestIndeces][:10])
print(vocabulary[bestIndeces][-10:])
```

```
['great' 'love' 'excellent' 'best' 'beautiful' 'perfect' 'favorite'
 'enjoy' 'amazing' 'performance']
['minutes' 'stupid' 'horrible' 'terrible' 'boring' 'worse' 'awful' 'waste'
 'worst' 'bad']
```

In [32]:
```
trainX = trainX[:,bestIndeces]
validX = validX[:,bestIndeces]
```

In [33]:
```
print(trainX.shape, trainY.shape)
print(validX.shape, validY.shape)
```

```
(15000, 2000) (15000,)
(10000, 2000) (10000,)
```

In [34]:
```
DROPOUT = 0.5
ACTIVATION = "tanh"

model = Sequential([
    Dense(int(trainX.shape[1]/2), activation=ACTIVATION, input_dim=trainX.shape[1]),
    Dropout(DROPOUT),
    Dense(int(trainX.shape[1]/2), activation=ACTIVATION, input_dim=trainX.shape[1]),
    Dropout(DROPOUT),
    Dense(int(trainX.shape[1]/4), activation=ACTIVATION),
    Dropout(DROPOUT),
    Dense(100, activation=ACTIVATION),
    Dropout(DROPOUT),
    Dense(20, activation=ACTIVATION),
    Dropout(DROPOUT),
    Dense(5, activation=ACTIVATION),
    Dropout(DROPOUT),
    Dense(1, activation='sigmoid'),
])
```

```
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
```

In [35]:
```
model.compile(optimizer=optimizers.Adam(0.00005), loss='binary_crossentropy',
metrics=['accuracy'])
model.summary()
```

```
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow
\python\ops\nn_impl.py:180: add_dispatch_support.<locals>.wrapper (from tenso
rflow.python.ops.array_ops) is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1000)              2001000
_________________________________________________________________
dropout (Dropout)            (None, 1000)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dropout_1 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               500500
_________________________________________________________________
dropout_2 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_3 (Dense)              (None, 100)               50100
_________________________________________________________________
dropout_3 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_4 (Dense)              (None, 20)                2020
_________________________________________________________________
dropout_4 (Dropout)          (None, 20)                0
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 105
_________________________________________________________________
dropout_5 (Dropout)          (None, 5)                 0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 6
=================================================================
Total params: 3,554,731
Trainable params: 3,554,731
Non-trainable params: 0
_________________________________________________________________
```

In [36]:
```
EPOCHS = 30
BATCHSIZE = 1500
```

In [37]:
```
%%time
model.fit(trainX, trainY, epochs=EPOCHS, batch_size=BATCHSIZE, validation_data
=(validX, validY))
```

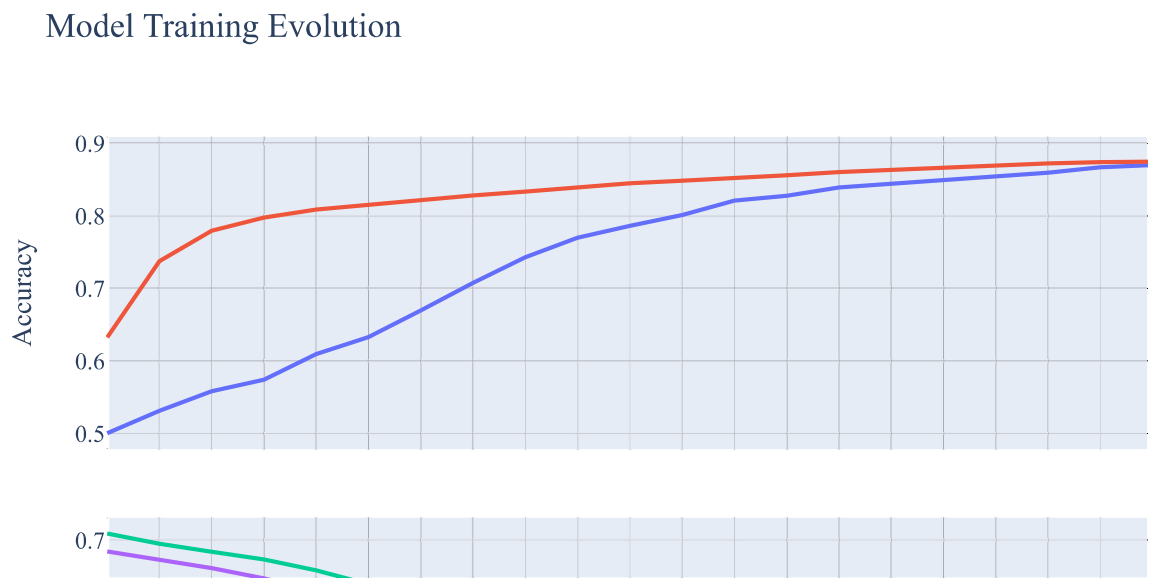
```
Train on 15000 samples, validate on 10000 samples
Epoch 1/30
15000/15000 [==============================] - 11s 756us/sample - loss: 0.709
0 - acc: 0.5003 - val_loss: 0.6843 - val_acc: 0.6322
Epoch 2/30
15000/15000 [==============================] - 9s 607us/sample - loss: 0.6950
- acc: 0.5312 - val_loss: 0.6733 - val_acc: 0.7375
Epoch 3/30
15000/15000 [==============================] - 9s 581us/sample - loss: 0.6837
- acc: 0.5579 - val_loss: 0.6611 - val_acc: 0.7797
Epoch 4/30
15000/15000 [==============================] - 9s 607us/sample - loss: 0.6730
- acc: 0.5743 - val_loss: 0.6469 - val_acc: 0.7977
Epoch 5/30
15000/15000 [==============================] - 9s 576us/sample - loss: 0.6578
- acc: 0.6095 - val_loss: 0.6287 - val_acc: 0.8085
Epoch 6/30
15000/15000 [==============================] - 10s 675us/sample - loss: 0.638
8 - acc: 0.6328 - val_loss: 0.6056 - val_acc: 0.8156
Epoch 7/30
15000/15000 [==============================] - 10s 659us/sample - loss: 0.616
7 - acc: 0.6695 - val_loss: 0.5772 - val_acc: 0.8223
Epoch 8/30
15000/15000 [==============================] - 8s 566us/sample - loss: 0.5875
- acc: 0.7076 - val_loss: 0.5442 - val_acc: 0.8280
Epoch 9/30
15000/15000 [==============================] - 9s 618us/sample - loss: 0.5588
- acc: 0.7429 - val_loss: 0.5089 - val_acc: 0.8332
Epoch 10/30
15000/15000 [==============================] - 9s 584us/sample - loss: 0.5312
- acc: 0.7697 - val_loss: 0.4748 - val_acc: 0.8389
Epoch 11/30
15000/15000 [==============================] - 9s 574us/sample - loss: 0.5012
- acc: 0.7859 - val_loss: 0.4445 - val_acc: 0.8448
Epoch 12/30
15000/15000 [==============================] - 9s 584us/sample - loss: 0.4837
- acc: 0.8013 - val_loss: 0.4190 - val_acc: 0.8482
Epoch 13/30
15000/15000 [==============================] - 9s 581us/sample - loss: 0.4549
- acc: 0.8211 - val_loss: 0.3989 - val_acc: 0.8515
Epoch 14/30
15000/15000 [==============================] - 9s 592us/sample - loss: 0.4414
- acc: 0.8275 - val_loss: 0.3823 - val_acc: 0.8559
Epoch 15/30
15000/15000 [==============================] - 9s 575us/sample - loss: 0.4256
- acc: 0.8392 - val_loss: 0.3692 - val_acc: 0.8603
Epoch 16/30
15000/15000 [==============================] - 9s 620us/sample - loss: 0.4164
- acc: 0.8439 - val_loss: 0.3599 - val_acc: 0.8632
Epoch 17/30
15000/15000 [==============================] - 9s 597us/sample - loss: 0.4052
- acc: 0.8485 - val_loss: 0.3505 - val_acc: 0.8664
Epoch 18/30
15000/15000 [==============================] - 9s 590us/sample - loss: 0.3931
- acc: 0.8538 - val_loss: 0.3440 - val_acc: 0.8706
Epoch 19/30
15000/15000 [==============================] - 9s 574us/sample - loss: 0.3896
```

```
- acc: 0.8596 - val_loss: 0.3386 - val_acc: 0.8721
Epoch 20/30
15000/15000 [==============================] - 9s 580us/sample - loss: 0.3761
- acc: 0.8668 - val_loss: 0.3345 - val_acc: 0.8740
Epoch 21/30
15000/15000 [==============================] - 9s 591us/sample - loss: 0.3730
- acc: 0.8703 - val_loss: 0.3311 - val_acc: 0.8748
Epoch 22/30
15000/15000 [==============================] - 9s 597us/sample - loss: 0.3688
- acc: 0.8709 - val_loss: 0.3282 - val_acc: 0.8763
Epoch 23/30
15000/15000 [==============================] - 9s 579us/sample - loss: 0.3622
- acc: 0.8743 - val_loss: 0.3264 - val_acc: 0.8763
Epoch 24/30
15000/15000 [==============================] - 9s 579us/sample - loss: 0.3594
- acc: 0.8773 - val_loss: 0.3240 - val_acc: 0.8774
Epoch 25/30
15000/15000 [==============================] - 9s 569us/sample - loss: 0.3556
- acc: 0.8768 - val_loss: 0.3222 - val_acc: 0.8781
Epoch 26/30
15000/15000 [==============================] - 9s 592us/sample - loss: 0.3507
- acc: 0.8797 - val_loss: 0.3211 - val_acc: 0.8779
Epoch 27/30
15000/15000 [==============================] - 9s 588us/sample - loss: 0.3485
- acc: 0.8834 - val_loss: 0.3202 - val_acc: 0.8769
Epoch 28/30
15000/15000 [==============================] - 8s 564us/sample - loss: 0.3475
- acc: 0.8803 - val_loss: 0.3197 - val_acc: 0.8777
Epoch 29/30
15000/15000 [==============================] - 9s 614us/sample - loss: 0.3480
- acc: 0.8795 - val_loss: 0.3191 - val_acc: 0.8768
Epoch 30/30
15000/15000 [==============================] - 9s 597us/sample - loss: 0.3415
- acc: 0.8870 - val_loss: 0.3188 - val_acc: 0.8767
Wall time: 4min 32s
```

Out[37]: <tensorflow.python.keras.callbacks.History at 0x1e71f6350f0>

In [38]:
```
x = np.arange(EPOCHS)
history = model.history.history

data = [
    go.Scatter(x=x, y=history["acc"], name="Train Accuracy", marker=dict(size=
5), yaxis='y2'),
    go.Scatter(x=x, y=history["val_acc"], name="Valid Accuracy", marker=dict(s
ize=5), yaxis='y2'),
    go.Scatter(x=x, y=history["loss"], name="Train Loss", marker=dict(size=5
)),
    go.Scatter(x=x, y=history["val_loss"], name="Valid Loss", marker=dict(size
=5))
]
layout = go.Layout(
    title="Model Training Evolution", font=dict(family='Palatino'), xaxis=dict
(title='Epoch', dtick=1),
    yaxis1=dict(title="Loss", domain=[0, 0.45]), yaxis2=dict(title="Accuracy",
domain=[0.55, 1]),
)
py.iplot(go.Figure(data=data, layout=layout), show_link=False)
```



In [39]:
```
train["probability"] = model.predict(trainX)
train["prediction"] = train.probability-0.5>0
train["truth"] = train.label==1
train.tail()
```

Out[39]:

	review	label	file	probability	prediction	truth
2455	Origins of the Care Bears & their Cousins. If ...	1	1220_9.txt	0.907206	True	True
1043	I always enjoy this movie when it shows up on ...	1	1093_8.txt	0.906141	True	True
17166	I watched this movie when Joe Bob Briggs hoste...	0	2950_1.txt	0.112409	False	False
24532	Former brat pack actor and all round pretty bo...	0	9580_3.txt	0.101270	False	False
23170	Any movie in which Brooke Shields out-acts a F...	0	8354_2.txt	0.095695	False	False

In [40]:
```
print(model.evaluate(trainX, trainY))
print((train.truth==train.prediction).mean())
```

```
15000/15000 [==============================] - 5s 329us/sample - loss: 0.2515
- acc: 0.9220
[0.2514871084690094, 0.922]
0.922
```

In [41]:
```
valid["probability"] = model.predict(validX)
valid["prediction"] = valid.probability-0.5>0
valid["truth"] = valid.label==1
valid.tail()
```

Out[41]:

	review	label	file	probability	prediction	truth
8087	There aren't too many times when I see a film ...	1	6029_7.txt	0.211895	False	True
11702	In Iran women are prohibited from attending li...	1	9283_8.txt	0.889475	True	True
1056	Big S isn't playing with taboos or forcing an ...	1	10951_8.txt	0.894298	True	True
13081	For the first forty minutes, Empire really sha...	0	10523_2.txt	0.096872	False	False
3811	Louise Brooks gives a wonderful performance in...	1	2180_8.txt	0.905861	True	True

In [42]:
```
print(model.evaluate(validX, validY))
print((valid.truth==valid.prediction).mean())
```

```
10000/10000 [==============================] - 3s 325us/sample - loss: 0.3188
- acc: 0.8767
[0.3188201421737671, 0.8767]
0.8767
```

In [43]:
```
trainCross = train.groupby(["prediction", "truth"]).size().unstack()
trainCross
```

Out[43]:

prediction	truth False	truth True
False	6919	559
True	611	6911

In [44]:
```
validCross = valid.groupby(["prediction", "truth"]).size().unstack()
validCross
```

Out[44]:

prediction	truth False	truth True
False	4339	602
True	631	4428

In [45]:
```
truepositives = valid[(valid.truth==True)&(valid.truth==valid.prediction)]
print(len(truepositives), "true positives.")
truepositives.sort_values("probability", ascending=False).head(3)
```

```
4428 true positives.
```

Out[45]:

	review	label	file	probability	prediction	truth
917	I have rarely emerged from viewing a film with...	1	10826_10.txt	0.908975	True	True
2339	Antonio Margheriti's "Danza Macabra" aka. "Cas...	1	12105_10.txt	0.908895	True	True
11228	Director Sidney Lumet has made some masterpiec...	1	8857_10.txt	0.908892	True	True

In [46]:
```
truenegatives = valid[(valid.truth==False)&(valid.truth==valid.prediction)]
print(len(truenegatives), "true negatives.")
truenegatives.sort_values("probability", ascending=True).head(3)
```

```
4339 true negatives.
```

Out[46]:

	review	label	file	probability	prediction	truth
19935	Aside from the horrendous acting and the ridic...	0	5442_3.txt	0.090714	False	False
20213	I can't believe the positive reviews of this m...	0	5693_1.txt	0.090723	False	False
23196	This movie is most possibly the worst movie I ...	0	8378_1.txt	0.090728	False	False

In [47]:
```
falsepositives = valid[(valid.truth==True)&(valid.truth!=valid.prediction)]
print(len(falsepositives), "false positives.")
falsepositives.sort_values("probability", ascending=True).head(3)
```

```
602 false positives.
```

Out[47]:

	review	label	file	probability	prediction	truth
12340	I wouldn't go so far as to not recommend this ...	1	9858_7.txt	0.091938	False	True
6038	This movie has everything that makes a bad mov...	1	4185_8.txt	0.092445	False	True
7982	After seeing the terrible, terrible, terrible ...	1	5935_9.txt	0.093050	False	True

In [48]:
```
falsenegatives = valid[(valid.truth==False)&(valid.truth!=valid.prediction)]
print(len(falsenegatives), "false negatives.")
falsenegatives.sort_values("probability", ascending=False).head(3)
```

```
631 false negatives.
```

Out[48]:

	review	label	file	probability	prediction	truth
16505	I've almost forever been against the inclusion...	0	2355_3.txt	0.907444	True	False
18569	The beginning of this movie is excellent with ...	0	4212_4.txt	0.906162	True	False
18682	I firmly believe that the best Oscar ceremony ...	0	4314_4.txt	0.906133	True	False

In [49]:
```
HTML(valid.loc[22148].review)
```

Out[49]: If this is based on the true-life relationship, as purported, between Ms. Curtin and Mr. Levinson, I'm thrilled I do not know them personally. This is painfully slow, and both characters take stupid pills liberally throughout the movie while the theme song gets played into the ground. Many stupid scenes with people acting stupid does not make for a comedy.

In [79]:
```
unseen = pd.Series('this movie is very good')
print(probability)
print("Positive!") if probability > 0.5 else print("Negative!")
```

```
0.095343485
Negative!
```

In [80]:
```
unseen = preprocess.transform(unseen)       # Text preprocessing
unseen = tfidf.transform(unseen).toarray()  # Feature engineering
unseen = unseen[:,bestIndeces]              # Feature selection
probability = model.predict(unseen)[0,0]
```

```
Transformed.
```

In [81]:
```
print(probability)
print("Positive!") if probability > 0.5 else print("Negative!")
```

```
0.6559083
Positive!
```

In [ ]:
```

```