Open in Colab

IMdb Sentiment Analysis Task

Importing usefull packages

In [1]:

```
# Import packages
import tensorflow as tf
# tf.enable_eager_execution()

from keras.datasets import imdb
from keras import preprocessing
from keras.models import Sequential
from keras.layers import Flatten, Dense
from keras.layers import Embedding
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras import losses
from keras import metrics
from keras import optimizers
from keras.layers import Dropout

import string
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

import nltk
nltk.download('punkt')
nltk.download('stopwords')

# Downgrate numpy to fix a problem
!pip install numpy==1.16.2
import numpy as np
print(np.__version__)

import matplotlib.pyplot as plt

import os

import numpy as np

import string
```

```python
!pip install ipdb
import ipdb # deb

from gensim.models.keyedvectors import KeyedVectors

# Spliting data
from sklearn.model_selection import train_test_split

from sklearn import metrics # For RUC

from nltk.stem import PorterStemmer

import tensorflow_hub as hub
import pandas as pd
import re
import seaborn as sns

# from google.colab import files

from IPython import display

import logging
logging.getLogger('googleapiclient.discovery_cache').setLevel(logging.ERROR)
```

```
Using TensorFlow backend.
[nltk_data] Downloading package punkt to /home/aims/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/aims/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
Collecting numpy==1.16.2
  Using cached https://files.pythonhosted.org/packages/91/e7/6c780e612d245cca62bc3ba
8e263038f7c144a96a54f877f3714a0e8427e/numpy-1.16.2-cp37-cp37m-manylinux1_x86_64.whl
Installing collected packages: numpy
  Found existing installation: numpy 1.17.2
    Uninstalling numpy-1.17.2:
      Successfully uninstalled numpy-1.17.2
Successfully installed numpy-1.16.2
1.17.2
Requirement already satisfied: ipdb in /home/aims/anaconda3/envs/aims-tf-1/lib/pytho
n3.7/site-packages (0.12.2)
Requirement already satisfied: setuptools in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from ipdb) (41.6.0.post20191030)
Requirement already satisfied: ipython>=5.1.0; python_version >= "3.4" in /home/aim
s/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages (from ipdb) (7.9.0)
Requirement already satisfied: pexpect; sys_platform != "win32" in /home/aims/anacon
da3/envs/aims-tf-1/lib/python3.7/site-packages (from ipython>=5.1.0; python_version
>= "3.4"->ipdb) (4.7.0)
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in /home/aims/anaconda3/
envs/aims-tf-1/lib/python3.7/site-packages (from ipython>=5.1.0; python_version >=
"3.4"->ipdb) (2.0.10)
Requirement already satisfied: pickleshare in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb) (0.7.
5)
Requirement already satisfied: decorator in /home/aims/anaconda3/envs/aims-tf-1/lib/
python3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb) (4.4.1)
Requirement already satisfied: jedi>=0.10 in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb) (0.1
5.1)
Requirement already satisfied: traitlets>=4.2 in /home/aims/anaconda3/envs/aims-tf-
1/lib/python3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb)
(4.3.3)
Requirement already satisfied: pygments in /home/aims/anaconda3/envs/aims-tf-1/lib/p
ython3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb) (2.4.2)
Requirement already satisfied: backcall in /home/aims/anaconda3/envs/aims-tf-1/lib/p
ython3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb) (0.1.0)
Requirement already satisfied: ptyprocess>=0.5 in /home/aims/anaconda3/envs/aims-tf-
1/lib/python3.7/site-packages (from pexpect; sys_platform != "win32"->ipython>=5.1.
0; python_version >= "3.4"->ipdb) (0.6.0)
Requirement already satisfied: wcwidth in /home/aims/anaconda3/envs/aims-tf-1/lib/py
thon3.7/site-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython>=5.1.0; python_ver
sion >= "3.4"->ipdb) (0.1.7)
Requirement already satisfied: six>=1.9.0 in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython>=5.1.0; python
_version >= "3.4"->ipdb) (1.12.0)
Requirement already satisfied: parso>=0.5.0 in /home/aims/anaconda3/envs/aims-tf-1/l
ib/python3.7/site-packages (from jedi>=0.10->ipython>=5.1.0; python_version >= "3.
4"->ipdb) (0.5.1)
Requirement already satisfied: ipython-genutils in /home/aims/anaconda3/envs/aims-tf
-1/lib/python3.7/site-packages (from traitlets>=4.2->ipython>=5.1.0; python_version
>= "3.4"->ipdb) (0.2.0)
```

In [ ]:

```
link = "https://drive.google.com/file/d/1smGRs2g2HoI6VSvonoZmWKzXOP6uPUaW/view?usp=

_, id_t = link.split('d/')
```

```
id = id_t.split('/')[0]
print (id) # Verify that you have everything after '='

# Install the PyDrive wrapper & import libraries.
# This only needs to be done once per notebook.
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authenticate and create the PyDrive client.
# This only needs to be done once per notebook.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)


file_id = id
downloaded = drive.CreateFile({'id':file_id})
downloaded.FetchMetadata(fetch_all=True)
downloaded.GetContentFile(downloaded.metadata['title'])
```

```
1smGRs2g2HoI6VSvonoZmWKzXOP6uPUaW
```

Loading data (Manual)

If you want to manually load the data from a tex file

In [ ]:

```
link = "https://drive.google.com/file/d/1smGRs2g2HoI6VSvonoZmWKzXOP6uPUaW/view?usp=

_, id_t = link.split('d/')

id = id_t.split('/')[0]
print (id) # Verify that you have everything after '='

# Install the PyDrive wrapper & import libraries.
# This only needs to be done once per notebook.
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authenticate and create the PyDrive client.
# This only needs to be done once per notebook.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)


file_id = id
downloaded = drive.CreateFile({'id':file_id})
```

```
downloaded.FetchMetadata(fetch_all=True)
downloaded.GetContentFile(downloaded.metadata['title'])
```

In [ ]:
```
!ls
```

```
aclImdb.zip   crawl-300d-2M-subword.bin   crawl-300d-2M-subword.zip
adc.json      crawl-300d-2M-subword.vec   sample_data
```

In [ ]:
```
!unzip -qq aclImdb.zip
```

In [ ]:
```
!ls
```

In [ ]:
```
# imdb_dir = './data/aclImdb'
imdb_dir = './aclImdb'

# Reading in the training folder
train_dir = os.path.join(imdb_dir, 'train')
texts_tr_ = []
labels_tr = []

for label_type in ['neg', 'pos']:
    dir_name = os.path.join(train_dir, label_type)
    for fname in os.listdir(dir_name):
        if fname[-4:] == '.txt':
            f = open(os.path.join(dir_name, fname), encoding="utf8")
            texts_tr_.append(f.read())
            f.close()
            if label_type == 'neg':
                labels_tr.append(0)
            else:
                labels_tr.append(1)

# Reading in the testing folder
train_dir = os.path.join(imdb_dir, 'test')
texts_tst_ = []
labels_tst = []

for label_type in ['neg', 'pos']:
    dir_name = os.path.join(train_dir, label_type)
    for fname in os.listdir(dir_name):
        if fname[-4:] == '.txt':
            f = open(os.path.join(dir_name, fname), encoding="utf8")
            texts_tst_.append(f.read())
            f.close()
            if label_type == 'neg':
                labels_tst.append(0)
            else:
                labels_tst.append(1)
```

In [ ]:
```
# Make sure that we have only 1 and 2 in the label
(np.unique(labels_tr), np.unique(labels_tst))
```

In [ ]:
```
len(labels_tr)
```

In [ ]:
```python
# Looking at 2 examples
print(texts_tr_[1])
print(labels_tr[1])

print(texts_tst_[-10])
print(labels_tst[-10])
```

Load pre-trained embedding

In [ ]:
```python
#################################################################################
########################################## Fasttext################################

# 2 million word vectors trained with subword information on Common Crawl (600B tok
# link = "https://drive.google.com/file/d/1Z_-E-gvsBP5BnJFKvX4sVnBVchdoUwsx/view?us

# 1 million word vectors trained with subword infomation on Wikipedia 2017,
# UMBC webbase corpus and statmt.org news dataset (16B tokens).
link = "https://drive.google.com/file/d/1lZhw_9cXME_eYbl1IwrMTyTEn6ONozgI/view?usp=

# crawl-300d-2M.vec.zip: 2 million word vectors trained on Common Crawl (600B token
# link = "https://drive.google.com/file/d/1jkJmSpRVZr_V2vGGkLmoEio6eY2Q2el-/view?us
#################################################################################
######################################### Word2vec ################################

# Developed by Tomas Mikolov at Google in 2013. Word2vec (https://code.google.com/a
# dimensions capture specific semantic properties
# link = "https://drive.google.com/file/d/13NmrtF-HoGmdv_m3Ld4eGVTbIekOcNi_/view?us
#################################################################################
########################################## Glove ##################################

# Go to https://nlp.stanford.edu/projects/glove, and download the precomputed embed
# from 2014 English Wikipedia. It's an 822 MB zip file called glove.6B.zip,
# containing 100-dimensional embedding vectors for 400,000 words (or nonword tokens
# link = "https://drive.google.com/file/d/1qlkC4-gpOhJVVvja8NEnnYPJFFzUu1CW/view?us
#################################################################################



#################################################################################

_, id_t = link.split('d/')


id = id_t.split('/')[0]
print (id) # Verify that you have everything after '='

# Install the PyDrive wrapper & import libraries.
# This only needs to be done once per notebook.
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
```

```
# Authenticate and create the PyDrive client.
# This only needs to be done once per notebook.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

file_id = id
downloaded = drive.CreateFile({'id':file_id})
downloaded.FetchMetadata(fetch_all=True)
downloaded.GetContentFile(downloaded.metadata['title'])
```

```
1lZhw_9cXME_eYbl1IwrMTyTEn6ONozgI
     |████████████████| 993kB 6.6MB/s
  Building wheel for PyDrive (setup.py) ... done
```

In [ ]:

```
!ls
```

```
adc.json  sample_data  wiki-news-300d-1M-subword.vec.zip
```

In [ ]:

```
# !unzip crawl-300d-2M.vec.zip
# !unzip crawl-300d-2M-subword.zip
!unzip wiki-news-300d-1M-subword.vec.zip
# !unzip glove.840B.300d.zip
```

```
Archive:  wiki-news-300d-1M-subword.vec.zip
  inflating: wiki-news-300d-1M-subword.vec
```

In [ ]:

```
# !rm crawl-300d-2M.vec.zip
```

In [ ]:

```
!ls
```

```
 adc.json                                       'Training and Validation loss.png'
 sample_data                                     wiki-news-300d-1M-subword.vec
'Training and validation accuracy.eps'           wiki-news-300d-1M-subword.vec.zip
```

In [ ]:

```
# gensim_w2v = KeyedVectors.load_word2vec_format('./crawl-300d-2M.vec')
gensim_w2v = KeyedVectors.load_word2vec_format('./wiki-news-300d-1M-subword.vec')
# gensim_w2v = KeyedVectors.load_word2vec_format('./GoogleNews-vectors-negative300.
# gensim_w2v = KeyedVectors.load_word2vec_format('./crawl-300d-2M-subword.bin')
```

```
W0521 18:59:08.737219 139952709396352 smart_open_lib.py:385] this function is deprec
ated, use smart_open.open instead
```

In [ ]:

```
gensim_w2v['I'].shape
```

Out[ ]:

```
(300,)
```

In [ ]:

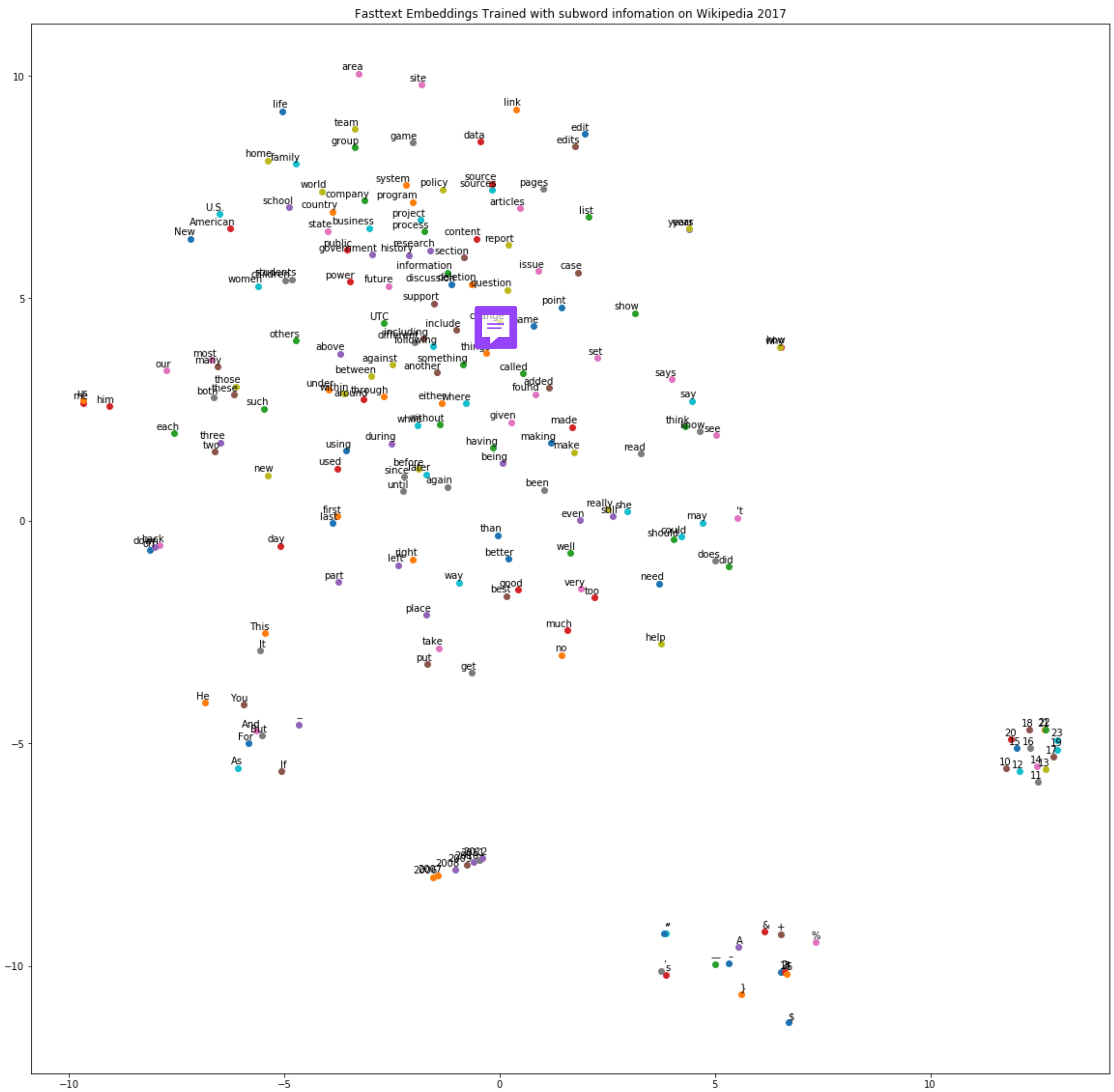
```
gensim_w2v["Hello"]
```

```
Out[ ]: array([-0.0038,  0.0601,  0.0188,  0.0288,  0.0031, -0.0114,  0.015 ,
        -0.0148, -0.004 , -0.0307,  0.015 ,  0.0423,  0.0133,  0.0243,
        -0.0038,  0.0272,  0.0356, -0.0024,  0.0257, -0.0343, -0.008 ,
         0.0032,  0.0065,  0.0043,  0.0186,  0.0173,  0.0047,  0.0351,
         0.0249, -0.0264, -0.0262,  0.0177,  0.0399,  0.0346, -0.0193,
         0.0078,  0.0046, -0.0115,  0.0021, -0.0317, -0.0078, -0.0675,
        -0.0009, -0.0058,  0.005 ,  0.0385,  0.0162, -0.0008, -0.0287,
         0.0565,  0.0094, -0.0034,  0.052 , -0.0209,  0.0455,  0.0119,
         0.008 , -0.0217, -0.0714, -0.0148,  0.0285, -0.0107, -0.0339,
         0.001 , -0.0323,  0.0292,  0.0139,  0.0141,  0.012 ,  0.0052,
        -0.0153,  0.0006,  0.0195, -0.0176,  0.0104, -0.017 ,  0.014 ,
        -0.0169,  0.0068,  0.0106, -0.0219, -0.025 ,  0.029 , -0.0596,
        -0.0245, -0.015 , -0.0285, -0.0399,  0.0048,  0.0084, -0.005 ,
         0.0104, -0.093 , -0.0481,  0.0094,  0.0111,  0.0026,  0.0017,
        -0.0146, -0.0191,  0.0015,  0.0279, -0.0163, -0.0197, -0.0904,
         0.0026, -0.0014,  0.0056,  0.0164, -0.0012, -0.0234,  0.0363,
         0.0214, -0.0242, -0.0097,  0.0062, -0.0147, -0.0073,  0.1028,
        -0.005 ,  0.0126, -0.0043,  0.0063, -0.0156, -0.0694, -0.0018,
         0.0622,  0.01  ,  0.0235,  0.0847, -0.028 ,  0.0356, -0.0216,
        -0.0182,  0.0408, -0.0136,  0.0231,  0.0385,  0.0118,  0.011 ,
        -0.0057,  0.0198, -0.0043,  0.0231,  0.025 , -0.006 ,  0.0144,
        -0.0033,  0.033 ,  0.0041, -0.0211, -0.028 ,  0.0143,  0.0381,
        -0.0297,  0.0048,  0.0247,  0.0127,  0.0058,  0.0233, -0.0179,
        -0.0144, -0.0348,  0.0093, -0.0391, -0.0735, -0.0159, -0.0102,
        -0.0056,  0.0319, -0.0032,  0.0241, -0.0269,  0.0022, -0.033 ,
         0.0459,  0.0079,  0.0103,  0.0116,  0.0144, -0.0065,  0.0151,
         0.0271, -0.0189, -0.0126,  0.0021, -0.0186,  0.1573,  0.0072,
         0.0091, -0.026 ,  0.0394, -0.0221,  0.029 , -0.0062,  0.0007,
         0.0184,  0.022 , -0.0113,  0.01  , -0.0463,  0.052 ,  0.0168,
         0.007 , -0.0231, -0.0032, -0.0165, -0.0193,  0.0079,  0.0517,
         0.0155,  0.0411, -0.003 , -0.0402,  0.0828, -0.0178, -0.0351,
        -0.0345, -0.0321, -0.0345, -0.013 , -0.0081,  0.0053,  0.0666,
         0.0305,  0.005 , -0.0311,  0.1185,  0.0576, -0.0697, -0.0086,
         0.0626,  0.0613, -0.0016,  0.046 , -0.0047,  0.0188, -0.0064,
        -0.0161, -0.0025, -0.0091,  0.0042,  0.0144, -0.0395,  0.0571,
        -0.0267,  0.0008,  0.0074, -0.0021, -0.0455,  0.06  ,  0.009 ,
         0.0112, -0.0302, -0.0185, -0.0247, -0.0137,  0.0084,  0.0029,
        -0.0384,  0.0055,  0.0096,  0.0357, -0.0146, -0.0107, -0.0267,
         0.0182,  0.0091,  0.0118, -0.0344,  0.0221,  0.0702,  0.0507,
         0.0059,  0.0493,  0.0018, -0.0043, -0.0182, -0.0266,  0.0116,
         0.0014,  0.0223,  0.0996, -0.0211, -0.0605, -0.0096,  0.0038,
        -0.0237,  0.0339, -0.0033, -0.0044, -0.0143, -0.0192,  0.025 ,
         0.036 ,  0.0064,  0.0324,  0.0071,  0.0049,  0.0229],
      dtype=float32)
```

In [ ]:

```
from sklearn.manifold import TSNE
from matplotlib import pylab

words = [word for word in gensim_w2v.index2word[200:300]]
embeddings = [gensim_w2v[word] for word in words]
words_embedded = TSNE(n_components=2).fit_transform(embeddings)

pylab.figure(figsize=(20, 20))
for i, label in enumerate(words):
  x, y = words_embedded[i, :]
  pylab.scatter(x, y)
```

```
    pylab.annotate(label, xy=(x, y), xytext=(5, 2), textcoords='offset points',
                   ha='right', va='bottom')

pylab.title('Fasttext Embeddings Trained with subword infomation on Wikipedia 2017'
pylab.savefig("Fasttest300-dembeddings.eps", format='eps', dpi=1200)
# pylab.axes(False)
pylab.show()
```




```
In [ ]: !ls
```

```
 adc.json                        sample_data
'Fasttest 300-d embeddings'      wiki-news-300d-1M-subword.vec
'Fasttest 300-d embeddings.eps'  wiki-news-300d-1M-subword.vec.zip
 Fasttest300-dembeddings.eps
```

```
In [ ]: files.download("Fasttest300-dembeddings.eps")
```

Approach (TF-Hub)

In [2]:
```
# Install TF-Hub.
!pip install tensorflow-hub
!pip install seaborn

# Install Ipdb
!pip install ipdb

from tensorflow.python.data import Dataset
```

```
Requirement already satisfied: tensorflow-hub in /home/aims/anaconda3/envs/aims-tf-
1/lib/python3.7/site-packages (0.6.0)
Requirement already satisfied: numpy>=1.12.0 in /home/aims/anaconda3/envs/aims-tf-1/
lib/python3.7/site-packages (from tensorflow-hub) (1.16.2)
Requirement already satisfied: six>=1.10.0 in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from tensorflow-hub) (1.12.0)
Requirement already satisfied: protobuf>=3.4.0 in /home/aims/anaconda3/envs/aims-tf-
1/lib/python3.7/site-packages (from tensorflow-hub) (3.10.0)
Requirement already satisfied: setuptools in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from protobuf>=3.4.0->tensorflow-hub) (41.6.0.post2019103
0)
Requirement already satisfied: seaborn in /home/aims/anaconda3/envs/aims-tf-1/lib/py
thon3.7/site-packages (0.9.0)
Requirement already satisfied: numpy>=1.9.3 in /home/aims/anaconda3/envs/aims-tf-1/l
ib/python3.7/site-packages (from seaborn) (1.16.2)
Requirement already satisfied: pandas>=0.15.2 in /home/aims/.local/lib/python3.7/sit
e-packages (from seaborn) (0.25.2)
Requirement already satisfied: scipy>=0.14.0 in /home/aims/anaconda3/envs/aims-tf-1/
lib/python3.7/site-packages (from seaborn) (1.3.1)
Requirement already satisfied: matplotlib>=1.4.3 in /home/aims/anaconda3/envs/aims-t
f-1/lib/python3.7/site-packages (from seaborn) (3.1.1)
Requirement already satisfied: python-dateutil>=2.6.1 in /home/aims/anaconda3/envs/a
ims-tf-1/lib/python3.7/site-packages (from pandas>=0.15.2->seaborn) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in /home/aims/anaconda3/envs/aims-tf-1/l
ib/python3.7/site-packages (from pandas>=0.15.2->seaborn) (2019.3)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /home/aim
s/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages (from matplotlib>=1.4.3->seab
orn) (2.4.2)
Requirement already satisfied: cycler>=0.10 in /home/aims/anaconda3/envs/aims-tf-1/l
ib/python3.7/site-packages (from matplotlib>=1.4.3->seaborn) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /home/aims/anaconda3/envs/aims-t
f-1/lib/python3.7/site-packages (from matplotlib>=1.4.3->seaborn) (1.1.0)
Requirement already satisfied: six>=1.5 in /home/aims/anaconda3/envs/aims-tf-1/lib/p
ython3.7/site-packages (from python-dateutil>=2.6.1->pandas>=0.15.2->seaborn) (1.12.
0)
Requirement already satisfied: setuptools in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from kiwisolver>=1.0.1->matplotlib>=1.4.3->seaborn) (41.
6.0.post20191030)
Requirement already satisfied: ipdb in /home/aims/anaconda3/envs/aims-tf-1/lib/pytho
n3.7/site-packages (0.12.2)
Requirement already satisfied: ipython>=5.1.0; python_version >= "3.4" in /home/aim
s/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages (from ipdb) (7.9.0)
Requirement already satisfied: setuptools in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from ipdb) (41.6.0.post20191030)
Requirement already satisfied: decorator in /home/aims/anaconda3/envs/aims-tf-1/lib/
python3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb) (4.4.1)
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in /home/aims/anaconda3/
envs/aims-tf-1/lib/python3.7/site-packages (from ipython>=5.1.0; python_version >=
"3.4"->ipdb) (2.0.10)
Requirement already satisfied: pickleshare in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb) (0.7.
5)
Requirement already satisfied: pexpect; sys_platform != "win32" in /home/aims/anacon
da3/envs/aims-tf-1/lib/python3.7/site-packages (from ipython>=5.1.0; python_version
>= "3.4"->ipdb) (4.7.0)
Requirement already satisfied: pygments in /home/aims/anaconda3/envs/aims-tf-1/lib/p
```

```
ython3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb) (2.4.2)
Requirement already satisfied: jedi>=0.10 in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb) (0.1
5.1)
Requirement already satisfied: backcall in /home/aims/anaconda3/envs/aims-tf-1/lib/p
ython3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb) (0.1.0)
Requirement already satisfied: traitlets>=4.2 in /home/aims/anaconda3/envs/aims-tf-
1/lib/python3.7/site-packages (from ipython>=5.1.0; python_version >= "3.4"->ipdb)
(4.3.3)
Requirement already satisfied: six>=1.9.0 in /home/aims/anaconda3/envs/aims-tf-1/li
b/python3.7/site-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython>=5.1.0; python
_version >= "3.4"->ipdb) (1.12.0)
Requirement already satisfied: wcwidth in /home/aims/anaconda3/envs/aims-tf-1/lib/py
thon3.7/site-packages (from prompt-toolkit<2.1.0,>=2.0.0->ipython>=5.1.0; python_ver
sion >= "3.4"->ipdb) (0.1.7)
Requirement already satisfied: ptyprocess>=0.5 in /home/aims/anaconda3/envs/aims-tf-
1/lib/python3.7/site-packages (from pexpect; sys_platform != "win32"->ipython>=5.1.
0; python_version >= "3.4"->ipdb) (0.6.0)
Requirement already satisfied: parso>=0.5.0 in /home/aims/anaconda3/envs/aims-tf-1/l
ib/python3.7/site-packages (from jedi>=0.10->ipython>=5.1.0; python_version >= "3.
4"->ipdb) (0.5.1)
Requirement already satisfied: ipython-genutils in /home/aims/anaconda3/envs/aims-tf
-1/lib/python3.7/site-packages (from traitlets>=4.2->ipython>=5.1.0; python_version
>= "3.4"->ipdb) (0.2.0)
```

More detailed information about installing Tensorflow can be found at https://www.tensorflow.org/install/.

Getting started

Data

We will try to solve the Large Movie Review Dataset v1.0 task from Mass et al. The dataset consists of IMDB movie reviews labeled by positivity from 1 to 10. The task is to label the reviews as **negative** or **positive**.

In [3]:
```python
# Load all files from a directory in a DataFrame.
def load_directory_data(directory):
  data = {}
  data["sentence"] = []
#   data["sentiment"] = []
  for file_path in os.listdir(directory):
    # changed tf.gfile by tf.io.gfile tf 2.0
    with tf.gfile.GFile(os.path.join(directory, file_path), "r") as f:
      data["sentence"].append(f.read())
#       data["sentiment"].append(re.match("\d+_(\d+)\.txt", file_path).group(1))#;
# ipdb.set_trace()
  return pd.DataFrame.from_dict(data)

# Merge positive and negative examples, add a polarity column and shuffle.
def load_dataset(directory):
  pos_df = load_directory_data(os.path.join(directory, "pos"))
  neg_df = load_directory_data(os.path.join(directory, "neg"))
  pos_df["sentiment"] = 1
```

```
  neg_df["sentiment"] = 0
  return pd.concat([pos_df, neg_df]).sample(frac=1).reset_index(drop=True)

# Download and process the dataset files.
def download_and_load_datasets(force_download=False):
  dataset = tf.keras.utils.get_file(
      fname="aclImdb.tar.gz",
      origin="http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz",
      extract=True)

  train_df = load_dataset(os.path.join(os.path.dirname(dataset),
                                       "aclImdb", "train"))
  test_df = load_dataset(os.path.join(os.path.dirname(dataset),
                                      "aclImdb", "test"))

  return train_df, test_df

# Reduce logging output.
tf.logging.set_verbosity(tf.logging.ERROR)

train_df, test_df = download_and_load_datasets()
```

In [8]:
```
# pd.set_option('display.max_colwidth', 200)
```

In [4]:
```
train_df.head(10)
```

Out[4]:

	sentence	sentiment
0	Kenny Doughty as Jed Willis is sexier in this ...	1
1	Having endured this inaccurate movie I will ad...	0
2	A friend of mine recommended this movie, citin...	1
3	The Matador is better upon reflection because ...	1
4	The progression of the plot is enough to "rope...	0
5	Those 2 points are dedicated the reasonable pe...	0
6	This is one of the most calming, relaxing, and...	1
7	WWE's last PPV of 2006, proved to be a hit wit...	1
8	What a gem of a movie, so good that they made ...	1
9	Let's set one thing straight: this movie does ...	1

In [5]:
```
test_df.tail(10)
```

Out[5]:

	sentence	sentiment
24990	yeesh,talk about craptastic.this thing is brut...	0
24991	I decided to watch this ultra-low budget film ...	0
24992	It didn't take too long after Halloween had ki...	0
24993	I thought this movie was absolutely hilarious....	1
24994	I say Ben Johnson and my fellow Canadians say,...	0
24995	Wow, I think the overall average rating of thi...	1
24996	I just got back from the GLBT Film Festival at...	1
24997	044: The Big Trail (1930) - released 10/24/193...	1
24998	** HERE BE SPOILERS ** Recap: Macl...	1
24999	I watched this movie three times at different ...	1

In [ ]:
```
test_df
```

Model

Input functions

Estimator framework provides input functions that wrap Pandas dataframes.

In [ ]:
```
# Training input on the whole training set with no limit on training epochs.
# # Changed estimator.inputs.pandas_input_fn by tf.compat.v1.estimator.inputs.panda
# train_input_fn = tf.compat.v1.estimator.inputs.pandas_input_fn(
#     train_df, train_df["sentiment"], num_echs=None, shuffle=True)
# # .head(10)
# # Prediction on the whole training set.
# predict_train_input_fn = tf.compat.v1.estimator.inputs.pandas_input_fn(
#     train_df, train_df["sentiment"], shuffle=False)
# # Prediction on the test set.
# predict_test_input_fn = tf.compat.v1.estimator.inputs.pandas_input_fn(
#     test_df, test_df["sentiment"], shuffle=False)
```

In [6]:
```
# Training input on the whole training set with no limit on training epochs.
train_input_fn = tf.estimator.inputs.pandas_input_fn(
    train_df, train_df["sentiment"], num_epochs=None, shuffle=True)
# .head(10)
# Prediction on the whole training set.
predict_train_input_fn = tf.estimator.inputs.pandas_input_fn(
    train_df, train_df["sentiment"], shuffle=False)
# Prediction on the test set.
predict_test_input_fn = tf.estimator.inputs.pandas_input_fn(
    test_df, test_df["sentiment"], shuffle=False)
```

Feature columns

TF-Hub provides a feature column that applies a module on the given text feature and passes further the outputs of the module. In this tutorial we will be using the nnlm-en-dim128 module. For the purpose of this tutorial, the most important facts are:

- The module takes **a batch of sentences in a 1-D tensor of strings** as input.
- The module is responsible for **preprocessing of sentences** (e.g. removal of punctuation and splitting on spaces).
- The module works with any input (e.g. **nnlm-en-dim128** hashes words not present in vocabulary into ~20.000 buckets).

In [7]:
```
embedded_text_feature_column = hub.text_embedding_column(
    key="sentence",
    module_spec="https://tfhub.dev/google/Wiki-words-250-with-normalization/1", tra
```

Estimator

For classification we can use a DNN Classifier (note further remarks about different modelling of the label function at the end of the tutorial).

In [9]:
```
estimator = tf.estimator.DNNClassifier(
    hidden_units=[512, 511, 512],
    feature_columns=[embedded_text_feature_column],
    n_classes=2,
    optimizer=tf.train.AdamOptimizer(learning_rate=0.003))
```

Training

Train the estimator for a reasonable amount of steps.

In [10]:
```
# Training for 20,000 steps means 128,000 training examples with the default
# batch size. This is roughly equivalent to 100 epochs since the training dataset
# contains 25,000 examples.
estimator.train(input_fn=train_input_fn, steps=20000);
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_estimator/python/e
stimator/estimator.py in _train_with_estimator_spec(self, estimator_spec, worker_hoo
ks, hooks, global_step_tensor, saving_listeners)
   1493       while not mon_sess.should_stop():
-> 1494         _, loss = mon_sess.run([estimator_spec.train_op, estimator_spec.los
s])
   1495         any_step_done = True

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/monitored_session.py in run(self, fetches, feed_dict, options, run_metadata)
    753         options=options,
--> 754         run_metadata=run_metadata)
    755

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/monitored_session.py in run(self, fetches, feed_dict, options, run_metadata)
   1258             options=options,
-> 1259             run_metadata=run_metadata)
   1260       except _PREEMPTION_ERRORS as e:

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/monitored_session.py in run(self, *args, **kwargs)
   1344     try:
-> 1345       return self._sess.run(*args, **kwargs)
   1346     except _PREEMPTION_ERRORS:

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/monitored_session.py in run(self, fetches, feed_dict, options, run_metadata)
   1425               options=options,
-> 1426               run_metadata=run_metadata))
   1427     self._should_stop = self._should_stop or run_context.stop_requested

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/basic_session_run_hooks.py in after_run(self, run_context, run_values)
    593         self._timer.update_last_triggered_step(global_step)
--> 594         if self._save(run_context.session, global_step):
    595           run_context.request_stop()

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/basic_session_run_hooks.py in _save(self, session, step)
    610
--> 611     self._get_saver().save(session, self._save_path, global_step=step)
    612     self._summary_writer.add_session_log(

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/saver.py in save(self, sess, save_path, global_step, latest_filename, meta_graph_
suffix, write_meta_graph, write_state, strip_default_attrs, save_debug_info)
   1175               self.saver_def.save_tensor_name,
-> 1176               {self.saver_def.filename_tensor_name: checkpoint_file})
   1177

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/clien
t/session.py in run(self, fetches, feed_dict, options, run_metadata)
    955       result = self._run(None, fetches, feed_dict, options_ptr,
```

```
--> 956                         run_metadata_ptr)
    957       if run_metadata:

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/clien
t/session.py in _run(self, handle, fetches, feed_dict, options, run_metadata)
   1179       results = self._do_run(handle, final_targets, final_fetches,
-> 1180                              feed_dict_tensor, options, run_metadata)
   1181     else:

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/clien
t/session.py in _do_run(self, handle, target_list, fetch_list, feed_dict, options, r
un_metadata)
   1358       return self._do_call(_run_fn, feeds, fetches, targets, options,
-> 1359                            run_metadata)
   1360     else:

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/clien
t/session.py in _do_call(self, fn, *args)
   1364     try:
-> 1365       return fn(*args)
   1366     except errors.OpError as e:

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/clien
t/session.py in _run_fn(feed_dict, fetch_list, target_list, options, run_metadata)
   1349       return self._call_tf_sessionrun(options, feed_dict, fetch_list,
-> 1350                                       target_list, run_metadata)
   1351 

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/clien
t/session.py in _call_tf_sessionrun(self, options, feed_dict, fetch_list, target_lis
t, run_metadata)
   1442                                               fetch_list, target_list,
-> 1443                                               run_metadata)
   1444 

KeyboardInterrupt: 

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-10-53b21840fd50> in <module>
      2 # batch size. This is roughly equivalent to 100 epochs since the training da
taset
      3 # contains 25,000 examples.
----> 4 estimator.train(input_fn=train_input_fn, steps=20000);

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_estimator/python/e
stimator/estimator.py in train(self, input_fn, hooks, steps, max_steps, saving_liste
ners)
    368 
    369       saving_listeners = _check_listeners_type(saving_listeners)
--> 370       loss = self._train_model(input_fn, hooks, saving_listeners)
    371       logging.info('Loss for final step: %s.', loss)
    372       return self

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_estimator/python/e
```

```
stimator/estimator.py in _train_model(self, input_fn, hooks, saving_listeners)
   1159       return self._train_model_distributed(input_fn, hooks, saving_listener
s)
   1160     else:
-> 1161       return self._train_model_default(input_fn, hooks, saving_listeners)
   1162
   1163   def _train_model_default(self, input_fn, hooks, saving_listeners):

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_estimator/python/e
stimator/estimator.py in _train_model_default(self, input_fn, hooks, saving_listener
s)
   1193       return self._train_with_estimator_spec(estimator_spec, worker_hooks,
   1194                                              hooks, global_step_tensor,
-> 1195                                              saving_listeners)
   1196
   1197   def _train_model_distributed(self, input_fn, hooks, saving_listeners):

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_estimator/python/e
stimator/estimator.py in _train_with_estimator_spec(self, estimator_spec, worker_hoo
ks, hooks, global_step_tensor, saving_listeners)
   1493       while not mon_sess.should_stop():
   1494         _, loss = mon_sess.run([estimator_spec.train_op, estimator_spec.los
s])
-> 1495         any_step_done = True
   1496     if not any_step_done:
   1497       logging.warning('Training with estimator made no steps. '

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/monitored_session.py in __exit__(self, exception_type, exception_value, tracebac
k)
    859     if exception_type in [errors.OutOfRangeError, StopIteration]:
    860       exception_type = None
--> 861     self._close_internal(exception_type)
    862     # __exit__ should return True to suppress an exception.
    863     return exception_type is None

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/monitored_session.py in _close_internal(self, exception_type)
    897         if self._sess is None:
    898           raise RuntimeError('Session is already closed.')
--> 899         self._sess.close()
    900       finally:
    901         self._sess = None

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/monitored_session.py in close(self)
   1164     if self._sess:
   1165       try:
-> 1166         self._sess.close()
   1167       except _PREEMPTION_ERRORS as e:
   1168         logging.warning(

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/monitored_session.py in close(self)
   1335     finally:
   1336       try:
```

```
-> 1337             _WrappedSession.close(self)
   1338         except Exception:  # pylint: disable=broad-except
   1339           # We intentionally suppress exceptions from the close() here since

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/monitored_session.py in close(self)
   1164     if self._sess:
   1165       try:
-> 1166         self._sess.close()
   1167       except _PREEMPTION_ERRORS as e:
   1168         logging.warning(

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/traini
ng/monitored_session.py in close(self)
   1164     if self._sess:
   1165       try:
-> 1166         self._sess.close()
   1167       except _PREEMPTION_ERRORS as e:
   1168         logging.warning(

~/anaconda3/envs/aims-tf-1/lib/python3.7/site-packages/tensorflow_core/python/clien
t/session.py in close(self)
    751     if self._session and not self._closed:
    752       self._closed = True
--> 753       tf_session.TF_CloseSession(self._session)
    754 
    755   def __del__(self):

KeyboardInterrupt:
```

In [ ]:

```
# 100*1000/5
```

Out[ ]:

```
20000.0
```

Prediction

Run predictions for both training and test set.

In [21]:

```
train_eval_result = estimator.evaluate(input_fn=predict_train_input_fn)
test_eval_result = estimator.evaluate(input_fn=predict_test_input_fn)

print(f"Training set accuracy: {train_eval_result['accuracy']*100:.1f} %")
print(f"Test set accuracy: {test_eval_result['accuracy']*100:.1f} %")
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Saver not created because there are no variables in the graph to res
tore
INFO:tensorflow:Saver not created because there are no variables in the graph to res
tore
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2019-12-24T11:22:28Z
INFO:tensorflow:Starting evaluation at 2019-12-24T11:22:28Z
```

```
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpuhnu7j4x/model.ckpt-196
INFO:tensorflow:Restoring parameters from /tmp/tmpuhnu7j4x/model.ckpt-196
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2019-12-24-11:23:08
INFO:tensorflow:Finished evaluation at 2019-12-24-11:23:08
INFO:tensorflow:Saving dict for global step 196: accuracy = 0.6654, accuracy_baseline = 0.5, auc = 0.7386924, auc_precision_recall = 0.73729104, average_loss = 0.6553721, global_step = 196, label/mean = 0.5, loss = 0.65532506, precision = 0.6355116, prediction/mean = 0.51865435, recall = 0.77568
INFO:tensorflow:Saving dict for global step 196: accuracy = 0.6654, accuracy_baseline = 0.5, auc = 0.7386924, auc_precision_recall = 0.73729104, average_loss = 0.6553721, global_step = 196, label/mean = 0.5, loss = 0.65532506, precision = 0.6355116, prediction/mean = 0.51865435, recall = 0.77568
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 196: /tmp/tmpuhnu7j4x/model.ckpt-196
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 196: /tmp/tmpuhnu7j4x/model.ckpt-196
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2019-12-24T11:23:19Z
INFO:tensorflow:Starting evaluation at 2019-12-24T11:23:19Z
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpuhnu7j4x/model.ckpt-196
INFO:tensorflow:Restoring parameters from /tmp/tmpuhnu7j4x/model.ckpt-196
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2019-12-24-11:24:00
INFO:tensorflow:Finished evaluation at 2019-12-24-11:24:00
INFO:tensorflow:Saving dict for global step 196: accuracy = 0.6574, accuracy_baseline = 0.5, auc = 0.7281414, auc_precision_recall = 0.7253007, average_loss = 0.6574305, global_step = 196, label/mean = 0.5, loss = 0.65742254, precision = 0.63006544, prediction/mean = 0.5172086, recall = 0.76248
INFO:tensorflow:Saving dict for global step 196: accuracy = 0.6574, accuracy_baseline = 0.5, auc = 0.7281414, auc_precision_recall = 0.7253007, average_loss = 0.6574305, global_step = 196, label/mean = 0.5, loss = 0.65742254, precision = 0.63006544, prediction/mean = 0.5172086, recall = 0.76248
```

```
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 196: /tmp/tmpuhnu7j
4x/model.ckpt-196
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 196: /tmp/tmpuhnu7j
4x/model.ckpt-196
Training set accuracy: 66.5 %
Test set accuracy: 65.7 %
```

In [ ]:
```
train_eval_result = estimator.evaluate(input_fn=predict_train_input_fn)
test_eval_result = estimator.evaluate(input_fn=predict_test_input_fn)

print(f"Training set accuracy: {train_eval_result['accuracy']*100:.1f} %")
print(f"Test set accuracy: {test_eval_result['accuracy']*100:.1f} %")
```

```
Training set accuracy: 100.0 %
Test set accuracy: 86.6 %
```

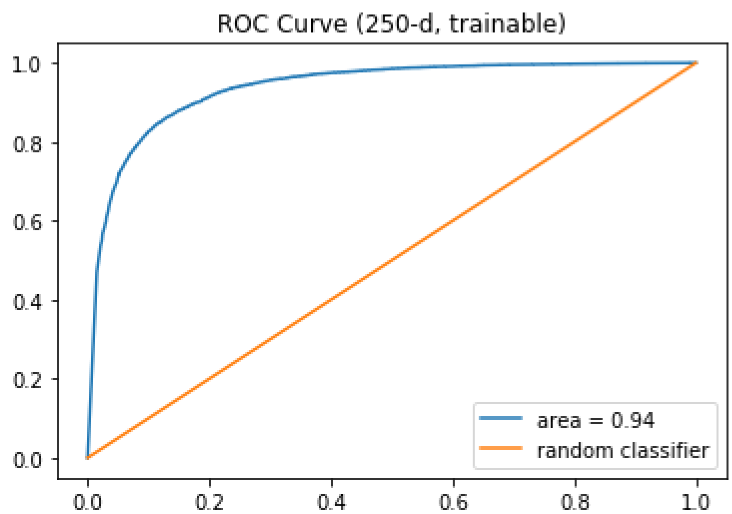
In [ ]:
```
print("Training set metrics:")
for m in train_eval_result:
  print(m, train_eval_result[m])

print("\n-----------------------------------\n")

print("Testing set metrics:")
for m in test_eval_result:
  print(m, test_eval_result[m])
# print("---")
```

```
Training set metrics:
accuracy 1.0
accuracy_baseline 0.5
auc 1.0
auc_precision_recall 1.0
average_loss 4.2935313e-05
label/mean 0.5
loss 0.005476443
precision 1.0
prediction/mean 0.50000554
recall 1.0
global_step 20000

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

Testing set metrics:
accuracy 0.86576
accuracy_baseline 0.5
auc 0.90123457
auc_precision_recall 0.91897255
average_loss 1.144381
label/mean 0.5
loss 145.96698
precision 0.8721309
prediction/mean 0.4916909
recall 0.8572
global_step 20000
```

Evaluation

In [ ]:
```
from sklearn.metrics import auc
validation_probabilities = estimator.predict(input_fn=predict_test_input_fn)
# Get just the probabilities for the positive class.
validation_probabilities = np.array([item['probabilities'][1] for item in validatio

false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(
    test_df["sentiment"], validation_probabilities)


roc_auc = auc(false_positive_rate,true_positive_rate)
plt.plot(false_positive_rate,true_positive_rate,label='area = %.2f' %roc_auc)
plt.plot([0, 1], [0, 1], label="random classifier")
plt.title("ROC Curve (250-d, trainable)")
plt.legend(loc=0)
plt.savefig("ROC curve (250).eps", format='eps', dpi=1200)
plt.show
```

Out[ ]: `<function matplotlib.pyplot.show>`



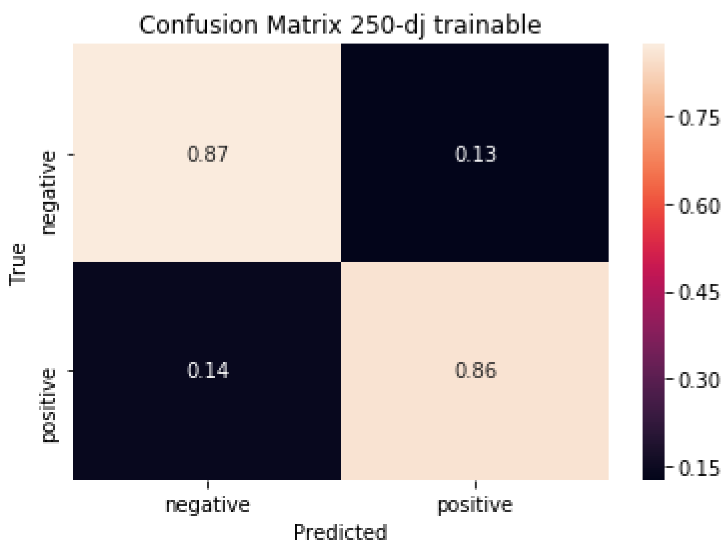
Confusion matrix

We can visually check the confusion matrix to understand the distribution of misclassifications.

In [ ]:
```
def get_predictions(estimator, input_fn):
  return [x["class_ids"][0] for x in estimator.predict(input_fn=input_fn)]

LABELS = [
    "negative", "positive"
]

# Create a confusion matrix on training data.
with tf.Graph().as_default():
  cm = tf.confusion_matrix(test_df["sentiment"],
                           get_predictions(estimator, predict_test_input_fn))
```

```
  with tf.Session() as session:
    cm_out = session.run(cm)

# Normalize the confusion matrix so that each row sums to 1.
cm_out = cm_out.astype(float) / cm_out.sum(axis=1)[:, np.newaxis]

sns.heatmap(cm_out, annot=True, xticklabels=LABELS, yticklabels=LABELS);
plt.xlabel("Predicted");
plt.ylabel("True");
plt.title("Confusion Matrix 250-dj trainable")
plt.savefig("Confusion matrix (250).eps", format='eps', dpi=1000)
plt.plot()
```

Out[ ]: []



In [ ]:
```
!ls
```

```
'Confusion matrix (250).eps'  'ROC curve (250).png'
'ROC curve (250).eps'          sample_data
```

In [ ]:
```
# files.download("Confusion matrix (250).eps")
files.download("ROC curve (250).eps")
# files.download("ROC curve (Bow+Fast_text).eps")
```

In [ ]:
```
def my_input_fn(features, targets, batch_size=1, shuffle=True, num_epochs=None):
    """Trains a linear regression model.

    Args:
      features: pandas DataFrame of features
      targets: pandas DataFrame of targets
      batch_size: Size of batches to be passed to the model
      shuffle: True or False. Whether to shuffle the data.
      num_epochs: Number of epochs for which data should be repeated. None = repeat
    Returns:
      Tuple of (features, labels) for next data batch
    """

    # Convert pandas data into a dict of np arrays.
    features = {key:np.array(value) for key,value in dict(features).items()}
```

```
    # Construct a dataset, and configure batching/repeating.
    ds = Dataset.from_tensor_slices((features,targets)) # warning: 2GB limit
    ds = ds.batch(batch_size).repeat(num_epochs)

    # Shuffle the data, if specified.
    if shuffle:
      ds = ds.shuffle(10000)

    # Return the next batch of data.
    features, labels = ds.make_one_shot_iterator().get_next()
    return features, labels

def preprocess_targets(dataframe_):
  """Prepares target features (i.e., labels) from California housing data set.

  Args:
    california_housing_dataframe: A Pandas DataFrame expected to contain data
      from the California housing data set.
  Returns:
    A DataFrame that contains the target feature.
  """
  output_targets = pd.DataFrame()
  # Create a boolean categorical feature representing whether the
  # median_house_value is above a set threshold.
  output_targets["sentiment"] = (dataframe_["sentiment"]).astype(float)
  return output_targets
```

In [ ]:
```
def train_DNN_classifier_model(
    learning_rate,
    steps,
    batch_size,
    training_examples,
    training_targets,
    validation_examples,
    validation_targets):
  """Trains a neural network classifier model.

  In addition to training, this function also prints training progress information,
  as well as a plot of the training and validation loss over time.

  Args:
    my_optimizer: An instance of `tf.train.Optimizer`, the optimizer to use.
    steps: A non-zero `int`, the total number of training steps. A training step
      consists of a forward and backward pass using a single batch.
    batch_size: A non-zero `int`, the batch size.
    hidden_units: A `list` of int values, specifying the number of neurons in each
    training_examples: A `DataFrame` containing one or more columns from
      `california_housing_dataframe` to use as input features for training.
    training_targets: A `DataFrame` containing exactly one column from
      `california_housing_dataframe` to use as target for training.
    validation_examples: A `DataFrame` containing one or more columns from
      `california_housing_dataframe` to use as input features for validation.
    validation_targets: A `DataFrame` containing exactly one column from
      `california_housing_dataframe` to use as target for validation.
```

```
Returns:
  A tuple `(estimator, training_losses, validation_losses)`:
    estimator: the trained `DNNRegressor` object.
    training_losses: a `list` containing the training loss values taken during tr
    validation_losses: a `list` containing the validation loss values taken durin
"""

periods = 10
steps_per_period = steps / periods

estimator = tf.estimator.DNNClassifier(
  hidden_units=[512],
  feature_columns=[embedded_text_feature_column],
  n_classes=2,
  optimizer=tf.train.AdagradOptimizer(learning_rate=learning_rate))

# Create input functions.
training_input_fn = lambda: my_input_fn(training_examples,
                                        training_targets["sentiment"],
                                        batch_size=batch_size)
predict_training_input_fn = lambda: my_input_fn(training_examples,
                                                training_targets["sentiment"],
                                                num_epochs=1,
                                                shuffle=False)
predict_validation_input_fn = lambda: my_input_fn(validation_examples,
                                                  validation_targets["sentiment"]
                                                  num_epochs=1,
                                                  shuffle=False)

# Train the model, but do so inside a loop so that we can periodically assess
# loss metrics.
print("Training model...")
print("LogLoss (on training data):")

training_log_losses = []
validation_log_losses = []

training_acc = []
validation_acc = []

for period in range (0, periods):
  # Train the model, starting from the prior state.
  estimator.train(
      input_fn=training_input_fn,
      steps=steps_per_period
  )

  # Take a break and compute predictions.
  training_probabilities_acc = estimator.predict(input_fn=predict_training_input_
  training_probabilities_acc = np.array([item['probabilities'][1] for item in tra

  validation_probabilities_acc = estimator.predict(input_fn=predict_validation_in
  validation_probabilities_acc = np.array([item['probabilities'][1] for item in v


  training_acc_ = metrics.accuracy_score(training_targets["sentiment"], training_
```

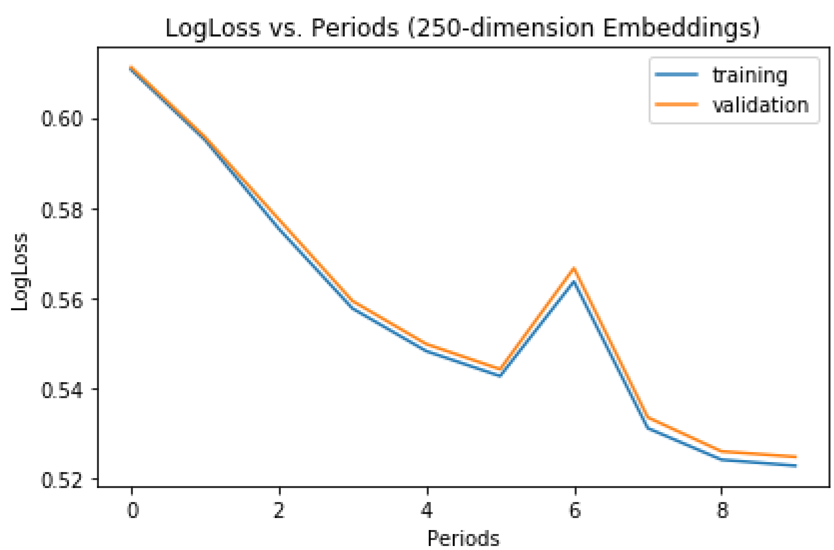
```
  validation_acc_ = metrics.accuracy_score(validation_targets["sentiment"], valid

  # Take a break and compute predictions.
  training_probabilities = estimator.predict(input_fn=predict_training_input_fn)
  training_probabilities = np.array([item['probabilities'] for item in training_p

  validation_probabilities = estimator.predict(input_fn=predict_validation_input_
  validation_probabilities = np.array([item['probabilities'] for item in validati

  training_log_loss = metrics.log_loss(training_targets, training_probabilities)
  validation_log_loss = metrics.log_loss(validation_targets, validation_probabili
  # Occasionally print the current loss.
  print("  period %02d : %0.2f" % (period, training_log_loss))
  # Add the loss metrics from this period to our list.
  training_log_losses.append(training_log_loss)
  validation_log_losses.append(validation_log_loss)

  training_acc.append(training_acc_)
  validation_acc.append(validation_acc_)

print("Model training finished.")

# Output a graph of loss metrics over periods.
plt.ylabel("LogLoss")
plt.xlabel("Periods")
plt.title("LogLoss vs. Periods (250-dimension Embeddings)")
plt.tight_layout()
plt.plot(training_log_losses, label="training")
plt.plot(validation_log_losses, label="validation")
plt.legend()
plt.show()

# Output a graph of loss metrics over periods.
plt.ylabel("Accuracy")
plt.xlabel("Periods")
plt.title("Accuracy vs. Periods (250-dimension Embeddings)")
plt.tight_layout()
plt.plot(training_acc, label="training")
plt.plot(validation_acc, label="validation")
plt.legend()
plt.show()

validation_probabilities_f = estimator.predict(input_fn=predict_test_input_fn)
# Get just the probabilities for the positive class.
validation_probabilities_f = np.array([item['probabilities'][1] for item in valid

false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(
    test_df["sentiment"], validation_probabilities_f)
plt.plot(false_positive_rate, true_positive_rate, label="our model")
plt.plot([0, 1], [0, 1], label="random classifier")
_ = plt.legend(loc=0)
plt.title("ROC curve (250-dimension Embeddings)")
plt.show()

train_eval_result = estimator.evaluate(input_fn=predict_train_input_fn)
test_eval_result = estimator.evaluate(input_fn=predict_test_input_fn)
```

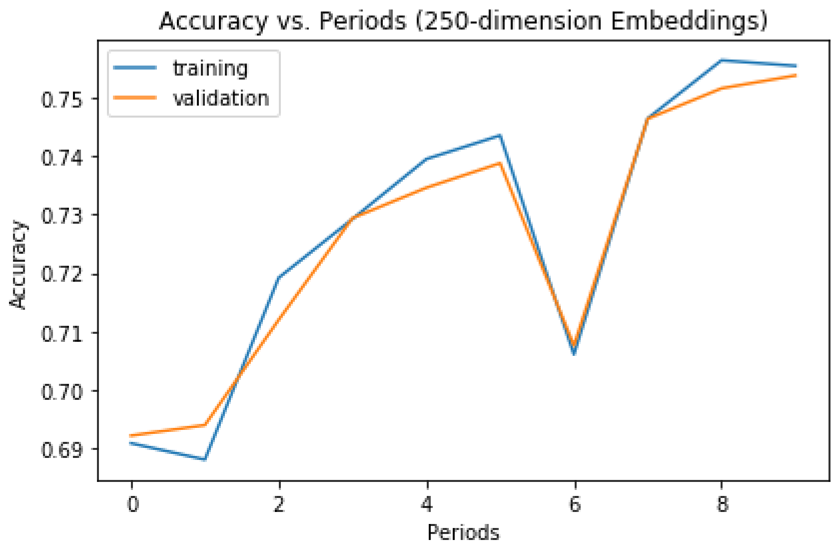
```python
    print(f"Training set accuracy: {train_eval_result['accuracy']*100:.1f} %")
    print(f"Test set accuracy: {test_eval_result['accuracy']*100:.1f} %")

    return estimator
```

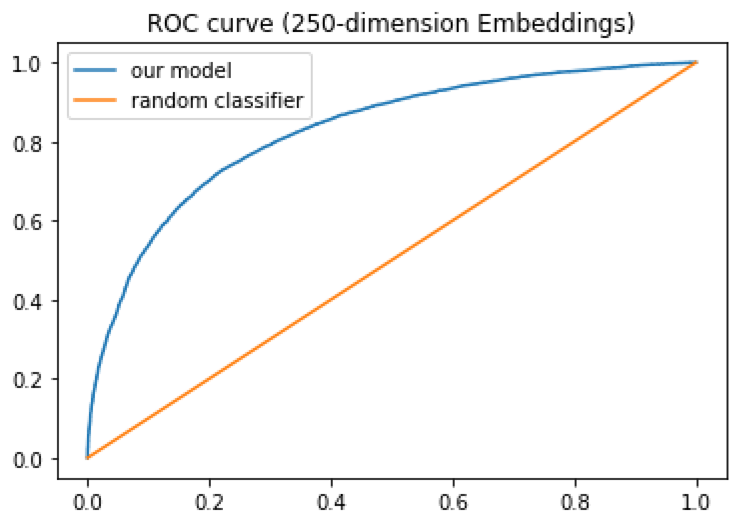
In [ ]:
```python
# Choose the first 12000 (out of 17000) examples for training.
training_examples = train_df.head(20000)
training_targets = preprocess_targets(train_df.head(20000))

# Choose the last 5000 (out of 17000) examples for validation.
validation_examples = train_df.tail(5000)
validation_targets = preprocess_targets(train_df.tail(5000))
```

In [ ]:
```python
DNN_classifie = train_DNN_classifier_model(
    learning_rate=0.003,
    steps=1000,
    batch_size=32,
    training_examples=training_examples,
    training_targets=training_targets,
    validation_examples=validation_examples,
    validation_targets=validation_targets)
```

```
Training model...
LogLoss (on training data):
  period 00 : 0.61
  period 01 : 0.60
  period 02 : 0.58
  period 03 : 0.56
  period 04 : 0.55
  period 05 : 0.54
  period 06 : 0.56
  period 07 : 0.53
  period 08 : 0.52
  period 09 : 0.52
Model training finished.
```








```
Training set accuracy: 75.5 %
Test set accuracy: 74.6 %
```

My Aproach

In [ ]:

```
# Install TF-Hub.
!pip install tensorflow-hub
!pip install seaborn
```

```
Requirement already satisfied: tensorflow-hub in /usr/local/lib/python3.6/dist-packa
ges (0.4.0)
Requirement already satisfied: protobuf>=3.4.0 in /usr/local/lib/python3.6/dist-pack
ages (from tensorflow-hub) (3.7.1)
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.6/dist-packag
es (from tensorflow-hub) (1.16.2)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages
(from tensorflow-hub) (1.12.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages
(from protobuf>=3.4.0->tensorflow-hub) (41.0.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.6/dist-packages (0.
9.0)
Requirement already satisfied: scipy>=0.14.0 in /usr/local/lib/python3.6/dist-packag
es (from seaborn) (1.2.1)
Requirement already satisfied: numpy>=1.9.3 in /usr/local/lib/python3.6/dist-package
s (from seaborn) (1.16.2)
Requirement already satisfied: pandas>=0.15.2 in /usr/local/lib/python3.6/dist-packa
ges (from seaborn) (0.24.2)
Requirement already satisfied: matplotlib>=1.4.3 in /usr/local/lib/python3.6/dist-pa
ckages (from seaborn) (3.0.3)
Requirement already satisfied: python-dateutil>=2.5.0 in /usr/local/lib/python3.6/di
st-packages (from pandas>=0.15.2->seaborn) (2.5.3)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages
(from pandas>=0.15.2->seaborn) (2018.9)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/loca
l/lib/python3.6/dist-packages (from matplotlib>=1.4.3->seaborn) (2.4.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-package
s (from matplotlib>=1.4.3->seaborn) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-pa
ckages (from matplotlib>=1.4.3->seaborn) (1.1.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (f
rom python-dateutil>=2.5.0->pandas>=0.15.2->seaborn) (1.12.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages
(from kiwisolver>=1.0.1->matplotlib>=1.4.3->seaborn) (41.0.1)
```

Getting started

Data

We will try to solve the Large Movie Review Dataset v1.0 task from Mass et al. The dataset consists of IMDB movie reviews labeled by positivity from 1 to 10. The task is to label the reviews as **negative** or **positive**.

In [ ]:
```
# Load all files from a directory in a DataFrame.
def load_directory_data(directory):
  data = {}
  data["sentence"] = []
#   data["sentiment"] = []
  for file_path in os.listdir(directory):
    with tf.gfile.GFile(os.path.join(directory, file_path), "r") as f:
      data["sentence"].append(f.read())
#       data["sentiment"].append(re.match("\d+_(\d+)\.txt", file_path).group(1))#;
  return pd.DataFrame.from_dict(data)
```

```python
# Merge positive and negative examples, add a polarity column and shuffle.
def load_dataset(directory):
  pos_df = load_directory_data(os.path.join(directory, "pos"))
  neg_df = load_directory_data(os.path.join(directory, "neg"))
  pos_df["sentiment"] = 1
  neg_df["sentiment"] = 0
  return pd.concat([pos_df, neg_df]).sample(frac=1).reset_index(drop=True)

# Download and process the dataset files.
def download_and_load_datasets(force_download=False):
  dataset = tf.keras.utils.get_file(
      fname="aclImdb.tar.gz",
      origin="http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz",
      extract=True)

  train_df = load_dataset(os.path.join(os.path.dirname(dataset),
                                       "aclImdb", "train"))
  test_df = load_dataset(os.path.join(os.path.dirname(dataset),
                                      "aclImdb", "test"))

  return train_df, test_df

# Reduce logging output.
tf.logging.set_verbosity(tf.logging.ERROR)

train_df, test_df = download_and_load_datasets()
```

In [ ]:

```python
# Double-check that we've done the right thing.
print("Training examples summary:")
display.display(train_df.describe())
display.display(train_df.head())
print("\n###############################\n")
print("Validation examples summary:")
display.display(test_df.describe())
display.display(test_df.head())
```

Training examples summary:

	sentiment
count	25000.00000
mean	0.50000
std	0.50001
min	0.00000
25%	0.00000
50%	0.50000
75%	1.00000
max	1.00000

	sentence	sentiment
0	I have always said that some plays by their ve...	0
1	There's nothing amazing about 'The Amazing Mr ...	0
2	This is according to me a quite bizarre movie ...	1
3	Having first watched the movie at 14, I rememb...	1
4	Many days after seeing Conceiving Ada, I am st...	0

```
#############################

Validation examples summary:
```

	sentiment
count	25000.00000
mean	0.50000
std	0.50001
min	0.00000
25%	0.00000
50%	0.50000
75%	1.00000
max	1.00000

	sentence	sentiment
0	This is how I interpreted the movie: First thi...	1
1	Michael Bassett's film 'Solomon Kane' (based o...	0
2	This movie cannot be serious because it has a ...	0
3	Being a "Wallace and Gromit-fan", I was lookin...	1
4	The plot: Four people are caught in an elevato...	0

In [ ]:

```
# Not really used
def clean_doc(text_1):
  ps = PorterStemmer()
  text_clean = []
#   ipdb.set_trace()
  for text in text_1:
    # split into tokens by white space
    tokens = word_tokenize(text)

    # convert to lower case
    tokens = [w.lower() for w in tokens]

    # remove punctuation from each token
```

```
    table = str.maketrans('', '', string.punctuation)
    stripped = [w.translate(table) for w in tokens]

    # remove remaining tokens that are not alphabetic
    tokens = [word for word in stripped if word.isalpha()]

    # filter out stop words
    stop_words = set(stopwords.words('english'))
    tokens = [w for w in tokens if not w in stop_words]

    # filter out short tokens
    tokens = [word for word in tokens if len(word) > 2]

    tokens = [ps.stem(w) for w in tokens]

    text_clean.append(tokens)

  return text_clean
```

In [ ]:

```
t=["him and I mamana", "Maman l'j#24 how are you", "marie", "python","pythoner","py
```

In [ ]:

```
clean_doc(t)
```

Out[ ]:

```
[['mamana'],
 ['maman'],
 ['mari'],
 ['python'],
 ['python'],
 ['python'],
 ['python'],
 ['pythonli']]
```

In [ ]:

```
texts_tr = train_df.sentence
texts_tst = test_df.sentence

# texts_tr = clean_doc(train_df.sentence)
# texts_tst = clean_doc(test_df.sentence)

# Training 92916
# Considers only the top
# 20,000 words in the dataset
max_words = 20000

tokenizer_tr = Tokenizer(num_words=max_words)
# tokenizer_tr = Tokenizer()
tokenizer_tr.fit_on_texts(texts_tr)
data_tr = tokenizer_tr.texts_to_sequences(texts_tr)
tfidf_train = tokenizer_tr.texts_to_matrix(texts_tr, mode='tfidf')
# bin_tr = tokenizer_tr.texts_to_matrix(texts_tr, mode='binary')


# Testing
tokenizer_tst = Tokenizer(num_words=max_words)
# tokenizer_tst = Tokenizer()
tokenizer_tst.fit_on_texts(texts_tst)
```

```
data_tst = tokenizer_tst.texts_to_sequences(texts_tst)
tfidf_tst = tokenizer_tr.texts_to_matrix(texts_tst, mode='tfidf')
# bin_tst = tokenizer_tr.texts_to_matrix(texts_tst, mode='binary')


word_index = tokenizer_tr.word_index
print('Found %s unique tokens.' % len(set(word_index))) #88582
```

```
Found 88582 unique tokens.
```

In [ ]:

```
%%time
bin_tr.shape
```

In [ ]:

```
len(data_tr)
```

Out[ ]:

```
25000
```

In [ ]:

```
tokenizer_tr
```

Out[ ]:

```
<keras_preprocessing.text.Tokenizer at 0x7f00136247b8>
```

In [ ]:

```
# max_len = 500 # depending on the size of the testing data 2332

# Cuts off reviews after 100 words
# max_len = 1239
max_len = 2500

# Train
x_train = pad_sequences(data_tr, maxlen = max_len)
# x_train = pad_sequences(data_tr)
y_train = np.asarray(train_df.sentiment)

# Suffle data
indices = np.arange(x_train.shape[0])
np.random.shuffle(indices)
x_train = x_train[indices]
y_train = y_train[indices]


# Testing
x_test = pad_sequences(data_tst, maxlen = max_len)
# x_test = pad_sequences(data_tst)

y_test = np.asarray(test_df.sentiment)

# indices = np.arange(x_test.shape[0])
# np.random.shuffle(indices)
# x_test = x_test[indices]
# y_test = y_test[indices]


# y = np.asarray(test_df.polarity)

# x_test, x_val, y_test, y_val = train_test_split(X, y, test_size=0.2, random_state
```

In [ ]:
```
x_train.shape
```
Out[ ]:
```
(25000, 2500)
```
In [ ]:
```
x_test.shape
```
Out[ ]:
```
(25000, 2500)
```
In [ ]:
```
# bin_tr.shape
```
In [ ]:
```
bin_tst.shape
```
Out[ ]:
```
(25000, 20000)
```
In [ ]:
```
(np.unique(y_test[:12500]), np.unique(y_train[:12500]))
```
Out[ ]:
```
(array([0, 1]), array([0, 1]))
```
In [ ]:
```
len(word_index)
```
Out[ ]:
```
88582
```
In [ ]:
```
word_index
```
In [ ]:
```
embedding_dim = gensim_w2v['hello'].shape[0]


number_invoc = 0
number_outvoc  = 0
oov = {}
embedding_matrix = np.zeros((max_words, embedding_dim))

for word, i in word_index.items():
    if i < max_words:
#          ipdb.set_trace()
        if word not in gensim_w2v.vocab:
            number_outvoc+=1
            oov[i]=word
            pass;
        else :
            number_invoc+=1
            embedding_vector = gensim_w2v[word]
            embedding_matrix[i] = embedding_vector
```
In [ ]:
```
len(oov)
```
Out[ ]:
```
2
```
In [ ]:
```
print(f"Number in vocabulary : {number_invoc}\nNumber out vocabulary : {number_outv
```
```
Number in vocabulary : 18116
Number out vocabulary : 1883
Sum : 19999
```

Version 01 (word embeddings to sentence embeddings)

```
x_train_ = np.zeros((x_train.shape[0], gensim_w2v['hello'].shape[0]))
for i, x in enumerate(x_train[:,:]):
#   ipdb.set_trace()
  for x_ in x:
    x_train_[i]+=embedding_matrix[x_]

print(f"Training processig finish\n")

x_test_ = np.zeros((x_test.shape[0], gensim_w2v['hello'].shape[0]))
for i, x in enumerate(x_test[:,:]):
  for x_ in x:
    x_test_[i]+=embedding_matrix[x_]

print(f"\n################################\nTesting Processing Done\n")


# x_val_ = np.zeros((x_val.shape[0], gensim_w2v['hello'].shape[0]))
# for i, x in enumerate(x_val[:,:]):
#   for x_ in x:
#     x_val_[i]+=embedding_matrix[x_]
```

Version 02 (word embeddings to sentence embeddings)

$$CBOW\left(f_1,\ldots,f_k\right)=\frac{1}{k}\sum_{i=1}^{k}v\left(f_i\right)$$

```
x_train_ = np.zeros((x_train.shape[0], gensim_w2v['hello'].shape[0]))
for i, x in enumerate(x_train[:,:]):
#   ipdb.set_trace()
  count_ = 0
  for x_ in x:
    count_+=1
    x_train_[i]+=embedding_matrix[x_]
  x_train_[i]/=count_

print(f"Training processig finish")

x_test_ = np.zeros((x_test.shape[0], gensim_w2v['hello'].shape[0]))
for i, x in enumerate(x_test[:,:]):
  for x_ in x:
    x_test_[i]+=embedding_matrix[x_]
  x_test_[i]/=count_

print(f"################################\nTesting Processing Done\n")

# x_val_ = np.zeros((x_val.shape[0], gensim_w2v['hello'].shape[0]))
# for i, x in enumerate(x_val[:,:]):
#   for x_ in x:
```

```
#     x_val_[i]+=embedding_matrix[x_]
#   x_val_[i]/=count_
```

In [ ]:
```
count_
```

In [ ]:
```
indices = np.arange(x_train.shape[0])
np.random.shuffle(indices)
x_train = x_train[indices]
y_train = y_train[indices]
```

Version 03 (word embeddings to sentence embeddings) the one used in the esssay

$$WCBOW\left(f_1,...,f_k\right)=\frac{1}{\sum_{i=1}^{k}a_i}\sum_{i=1}^{k}a_i v\left(f_i\right)$$

In [ ]:
```
x_train_ = np.zeros((x_train.shape[0], gensim_w2v['hello'].shape[0]))

print(f"Starting Training processing ...")
for i, x in enumerate(x_train[:,:]):
#   ipdb.set_trace()
  for x_ in x:
    x_train_[i]+=(embedding_matrix[x_]*tfidf_train[i,x_])
  x_train_[i]/=(np.sum(tfidf_train[i,:]))

print(f"##############################\nTraining processig finish\n")


x_test_ = np.zeros((x_test.shape[0], gensim_w2v['hello'].shape[0]))

print(f"Starting Testing processing ...")
for i, x in enumerate(x_test[:,:]):
  for x_ in x:
    x_test_[i]+=(embedding_matrix[x_]*tfidf_tst[i,x_])
  x_test_[i]/=(np.sum(tfidf_tst[i,:]))

print(f"##############################\nTesting Processing Done")


# x_val_ = np.zeros((x_val.shape[0], gensim_w2v['hello'].shape[0]))
# for i, x in enumerate(x_val[:,:]):
#   for x_ in x:
#     x_val_[i]+=(embedding_matrix[x_]*tfidf_train[i,x_])
#   x_val_[i]/=(np.sum(tfidf_train[i,:]))
```

```
Starting Training processing ...
##############################
Training processig finish

Starting Testing processing ...
##############################
Testing Processing Done
```

In [ ]:
```
np.unique(y_train[:12500])
```

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

Version 04 (word embeddings to sentence embeddings)

$$WCBOW\left(f_1, \ldots, f_k\right) = \frac{1}{\sqrt{\sum_{i=1}^{k} a_i^2}} \sum_{i=1}^{k} a_i v\left(f_i\right)$$

In [ ]:
```
x_train_ = np.zeros((x_train.shape[0], gensim_w2v['hello'].shape[0]))

print(f"Starting Training processing ...")
for i, x in enumerate(x_train[:,:]):
#   ipdb.set_trace()
  for x_ in x:
    x_train_[i]+=(embedding_matrix[x_]*tfidf_train[i,x_])
  x_train_[i]/=(np.sqrt(np.sum([tfidf_train[i, j]**2 for j in range(tfidf_train.sha

print(f"##############################\nTraining processig finish\n")


x_test_ = np.zeros((x_test.shape[0], gensim_w2v['hello'].shape[0]))

print(f"Starting Testing processing ...")
for i, x in enumerate(x_test[:,:]):
  for x_ in x:
    x_test_[i]+=(embedding_matrix[x_]*tfidf_tst[i,x_])
  x_test_[i]/=(np.sqrt(np.sum([tfidf_tst[i, j]**2 for j in range(tfidf_tst.shape[1]

print(f"##############################\nTesting Processing Done")


# x_val_ = np.zeros((x_val.shape[0], gensim_w2v['hello'].shape[0]))
# for i, x in enumerate(x_val[:,:]):
#   for x_ in x:
#     x_val_[i]+=(embedding_matrix[x_]*tfidf_train[i,x_])
#   x_val_[i]/=(np.sum(tfidf_train[i,:]))
```

```
Starting Training processing ...
##############################
Training processig finish

Starting Testing processing ...
##############################
Testing Processing Done
```

Model

In [ ]:
```
%%time
```

```
model = tf.keras.Sequential()
# model.add(Embedding(max_words, embedding_dim, input_length=maxlen))
# model.add(tf.keras.layers.Embedding(max_words,
#                                      embedding_dim,
#                                      input_length=x_train.shape[1],
#                                      embeddings_initializer=tf.keras.initializers.
#                                          embedding_matrix),
#                                      trainable = False
#                                     ))
# model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(512, input_shape=(x_train_.shape[1], ), activation=
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
model.summary()

class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('val_acc')>0.66):
      print("\nReached 90% val_acc so cancelling training!")
      self.model.stop_training = True

callbacks = myCallback()

monitor = tf.keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=1e-3, pati

model.compile(loss='binary_crossentropy', optimizer=tf.keras.optimizers.Adamax(0.00

# fit network
model.fit(x_train_, y_train, epochs=100,
          batch_size=64, verbose=2, validation_split=0.2)#,
#           validation_split=0.01, callbacks=[callbacks])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_8 (Dense)              (None, 512)               154112    
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_9 (Dense)              (None, 512)               262656    
_________________________________________________________________
dense_10 (Dense)             (None, 512)               262656    
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 513       
=================================================================
Total params: 679,937
Trainable params: 679,937
Non-trainable params: 0
_________________________________________________________________
Train on 20000 samples, validate on 5000 samples
Epoch 1/100
 - 2s - loss: 0.6899 - acc: 0.5321 - val_loss: 0.6861 - val_acc: 0.5442
Epoch 2/100
 - 2s - loss: 0.6841 - acc: 0.5559 - val_loss: 0.6784 - val_acc: 0.5688
Epoch 3/100
 - 2s - loss: 0.6775 - acc: 0.5709 - val_loss: 0.6765 - val_acc: 0.5664
Epoch 4/100
 - 2s - loss: 0.6726 - acc: 0.5867 - val_loss: 0.6775 - val_acc: 0.5614
Epoch 5/100
 - 2s - loss: 0.6688 - acc: 0.5926 - val_loss: 0.6693 - val_acc: 0.5892
Epoch 6/100
 - 2s - loss: 0.6658 - acc: 0.5915 - val_loss: 0.6701 - val_acc: 0.5866
Epoch 7/100
 - 2s - loss: 0.6651 - acc: 0.5980 - val_loss: 0.6658 - val_acc: 0.5974
Epoch 8/100
 - 2s - loss: 0.6625 - acc: 0.6011 - val_loss: 0.6650 - val_acc: 0.5960
Epoch 9/100
 - 2s - loss: 0.6624 - acc: 0.5993 - val_loss: 0.6623 - val_acc: 0.5992
Epoch 10/100
 - 2s - loss: 0.6592 - acc: 0.6063 - val_loss: 0.6619 - val_acc: 0.6032
Epoch 11/100
 - 2s - loss: 0.6577 - acc: 0.6078 - val_loss: 0.6589 - val_acc: 0.6038
Epoch 12/100
 - 2s - loss: 0.6563 - acc: 0.6076 - val_loss: 0.6645 - val_acc: 0.5960
Epoch 13/100
 - 2s - loss: 0.6550 - acc: 0.6145 - val_loss: 0.6603 - val_acc: 0.5958
Epoch 14/100
 - 2s - loss: 0.6556 - acc: 0.6109 - val_loss: 0.6637 - val_acc: 0.5922
Epoch 15/100
 - 2s - loss: 0.6510 - acc: 0.6173 - val_loss: 0.6725 - val_acc: 0.5890
Epoch 16/100
 - 2s - loss: 0.6532 - acc: 0.6144 - val_loss: 0.6558 - val_acc: 0.6066
Epoch 17/100
 - 2s - loss: 0.6515 - acc: 0.6212 - val_loss: 0.6551 - val_acc: 0.6098
Epoch 18/100
 - 2s - loss: 0.6505 - acc: 0.6121 - val_loss: 0.6574 - val_acc: 0.6100
Epoch 19/100
 - 2s - loss: 0.6500 - acc: 0.6215 - val_loss: 0.6559 - val_acc: 0.6040
```

```
Epoch 20/100
 - 2s - loss: 0.6490 - acc: 0.6231 - val_loss: 0.6536 - val_acc: 0.6136
Epoch 21/100
 - 2s - loss: 0.6508 - acc: 0.6188 - val_loss: 0.6560 - val_acc: 0.6062
Epoch 22/100
 - 2s - loss: 0.6500 - acc: 0.6203 - val_loss: 0.6562 - val_acc: 0.6134
Epoch 23/100
 - 2s - loss: 0.6479 - acc: 0.6256 - val_loss: 0.6621 - val_acc: 0.5932
Epoch 24/100
 - 2s - loss: 0.6478 - acc: 0.6234 - val_loss: 0.6524 - val_acc: 0.6162
Epoch 25/100
 - 2s - loss: 0.6471 - acc: 0.6206 - val_loss: 0.6521 - val_acc: 0.6182
Epoch 26/100
 - 2s - loss: 0.6460 - acc: 0.6270 - val_loss: 0.6614 - val_acc: 0.5964
Epoch 27/100
 - 2s - loss: 0.6467 - acc: 0.6230 - val_loss: 0.6564 - val_acc: 0.6122
Epoch 28/100
 - 2s - loss: 0.6460 - acc: 0.6249 - val_loss: 0.6521 - val_acc: 0.6126
Epoch 29/100
 - 2s - loss: 0.6443 - acc: 0.6258 - val_loss: 0.6506 - val_acc: 0.6148
Epoch 30/100
 - 2s - loss: 0.6426 - acc: 0.6281 - val_loss: 0.6562 - val_acc: 0.6058
Epoch 31/100
 - 2s - loss: 0.6450 - acc: 0.6255 - val_loss: 0.6503 - val_acc: 0.6172
Epoch 32/100
 - 2s - loss: 0.6439 - acc: 0.6267 - val_loss: 0.6538 - val_acc: 0.6112
Epoch 33/100
 - 2s - loss: 0.6429 - acc: 0.6324 - val_loss: 0.6499 - val_acc: 0.6184
Epoch 34/100
 - 2s - loss: 0.6435 - acc: 0.6321 - val_loss: 0.6497 - val_acc: 0.6198
Epoch 35/100
 - 2s - loss: 0.6432 - acc: 0.6275 - val_loss: 0.6633 - val_acc: 0.5902
Epoch 36/100
 - 2s - loss: 0.6420 - acc: 0.6299 - val_loss: 0.6502 - val_acc: 0.6184
Epoch 37/100
 - 2s - loss: 0.6432 - acc: 0.6289 - val_loss: 0.6526 - val_acc: 0.6128
Epoch 38/100
 - 2s - loss: 0.6407 - acc: 0.6348 - val_loss: 0.6523 - val_acc: 0.6152
Epoch 39/100
 - 2s - loss: 0.6415 - acc: 0.6308 - val_loss: 0.6502 - val_acc: 0.6160
Epoch 40/100
 - 2s - loss: 0.6416 - acc: 0.6277 - val_loss: 0.6490 - val_acc: 0.6210
Epoch 41/100
 - 2s - loss: 0.6413 - acc: 0.6298 - val_loss: 0.6484 - val_acc: 0.6180
Epoch 42/100
 - 2s - loss: 0.6396 - acc: 0.6307 - val_loss: 0.6571 - val_acc: 0.6024
Epoch 43/100
 - 2s - loss: 0.6407 - acc: 0.6298 - val_loss: 0.6491 - val_acc: 0.6176
Epoch 44/100
 - 2s - loss: 0.6404 - acc: 0.6329 - val_loss: 0.6521 - val_acc: 0.6112
Epoch 45/100
 - 2s - loss: 0.6405 - acc: 0.6292 - val_loss: 0.6620 - val_acc: 0.5972
Epoch 46/100
 - 2s - loss: 0.6406 - acc: 0.6337 - val_loss: 0.6494 - val_acc: 0.6224
Epoch 47/100
 - 2s - loss: 0.6379 - acc: 0.6320 - val_loss: 0.6496 - val_acc: 0.6164
```

```
Epoch 48/100
 - 2s - loss: 0.6396 - acc: 0.6327 - val_loss: 0.6488 - val_acc: 0.6192
Epoch 49/100
 - 2s - loss: 0.6382 - acc: 0.6325 - val_loss: 0.6517 - val_acc: 0.6160
Epoch 50/100
 - 2s - loss: 0.6383 - acc: 0.6342 - val_loss: 0.6496 - val_acc: 0.6148
Epoch 51/100
 - 2s - loss: 0.6379 - acc: 0.6363 - val_loss: 0.6528 - val_acc: 0.6168
Epoch 52/100
 - 2s - loss: 0.6369 - acc: 0.6392 - val_loss: 0.6480 - val_acc: 0.6180
Epoch 53/100
 - 2s - loss: 0.6382 - acc: 0.6341 - val_loss: 0.6456 - val_acc: 0.6246
Epoch 54/100
 - 2s - loss: 0.6369 - acc: 0.6353 - val_loss: 0.6524 - val_acc: 0.6136
Epoch 55/100
 - 2s - loss: 0.6363 - acc: 0.6371 - val_loss: 0.6488 - val_acc: 0.6238
Epoch 56/100
 - 2s - loss: 0.6339 - acc: 0.6382 - val_loss: 0.6491 - val_acc: 0.6242
Epoch 57/100
 - 2s - loss: 0.6372 - acc: 0.6349 - val_loss: 0.6471 - val_acc: 0.6238
Epoch 58/100
 - 2s - loss: 0.6362 - acc: 0.6360 - val_loss: 0.6464 - val_acc: 0.6262
Epoch 59/100
 - 2s - loss: 0.6354 - acc: 0.6374 - val_loss: 0.6478 - val_acc: 0.6202
Epoch 60/100
 - 2s - loss: 0.6352 - acc: 0.6401 - val_loss: 0.6500 - val_acc: 0.6150
Epoch 61/100
 - 2s - loss: 0.6342 - acc: 0.6391 - val_loss: 0.6530 - val_acc: 0.6110
Epoch 62/100
 - 2s - loss: 0.6345 - acc: 0.6400 - val_loss: 0.6465 - val_acc: 0.6204
Epoch 63/100
 - 2s - loss: 0.6357 - acc: 0.6353 - val_loss: 0.6458 - val_acc: 0.6254
Epoch 64/100
 - 2s - loss: 0.6337 - acc: 0.6405 - val_loss: 0.6504 - val_acc: 0.6172
Epoch 65/100
 - 2s - loss: 0.6322 - acc: 0.6407 - val_loss: 0.6481 - val_acc: 0.6194
Epoch 66/100
 - 2s - loss: 0.6346 - acc: 0.6380 - val_loss: 0.6457 - val_acc: 0.6232
Epoch 67/100
 - 2s - loss: 0.6324 - acc: 0.6405 - val_loss: 0.6465 - val_acc: 0.6250
Epoch 68/100
 - 2s - loss: 0.6321 - acc: 0.6409 - val_loss: 0.6482 - val_acc: 0.6174
Epoch 69/100
 - 2s - loss: 0.6332 - acc: 0.6385 - val_loss: 0.6452 - val_acc: 0.6220
Epoch 70/100
 - 2s - loss: 0.6326 - acc: 0.6366 - val_loss: 0.6454 - val_acc: 0.6290
Epoch 71/100
 - 2s - loss: 0.6329 - acc: 0.6378 - val_loss: 0.6459 - val_acc: 0.6276
Epoch 72/100
 - 2s - loss: 0.6328 - acc: 0.6388 - val_loss: 0.6485 - val_acc: 0.6254
Epoch 73/100
 - 2s - loss: 0.6320 - acc: 0.6392 - val_loss: 0.6512 - val_acc: 0.6132
Epoch 74/100
 - 2s - loss: 0.6309 - acc: 0.6417 - val_loss: 0.6470 - val_acc: 0.6208
Epoch 75/100
 - 2s - loss: 0.6325 - acc: 0.6402 - val_loss: 0.6462 - val_acc: 0.6266
```

```
Epoch 76/100
 - 2s - loss: 0.6313 - acc: 0.6395 - val_loss: 0.6507 - val_acc: 0.6218
Epoch 77/100
 - 2s - loss: 0.6294 - acc: 0.6413 - val_loss: 0.6484 - val_acc: 0.6120
Epoch 78/100
 - 2s - loss: 0.6308 - acc: 0.6418 - val_loss: 0.6494 - val_acc: 0.6210
Epoch 79/100
 - 2s - loss: 0.6298 - acc: 0.6434 - val_loss: 0.6478 - val_acc: 0.6202
Epoch 80/100
 - 2s - loss: 0.6306 - acc: 0.6416 - val_loss: 0.6460 - val_acc: 0.6264
Epoch 81/100
 - 2s - loss: 0.6291 - acc: 0.6439 - val_loss: 0.6447 - val_acc: 0.6280
Epoch 82/100
 - 2s - loss: 0.6303 - acc: 0.6431 - val_loss: 0.6452 - val_acc: 0.6244
Epoch 83/100
 - 2s - loss: 0.6296 - acc: 0.6454 - val_loss: 0.6459 - val_acc: 0.6258
Epoch 84/100
 - 2s - loss: 0.6293 - acc: 0.6397 - val_loss: 0.6475 - val_acc: 0.6196
Epoch 85/100
 - 2s - loss: 0.6298 - acc: 0.6434 - val_loss: 0.6448 - val_acc: 0.6208
Epoch 86/100
 - 2s - loss: 0.6275 - acc: 0.6454 - val_loss: 0.6474 - val_acc: 0.6244
Epoch 87/100
 - 2s - loss: 0.6278 - acc: 0.6425 - val_loss: 0.6503 - val_acc: 0.6170
Epoch 88/100
 - 2s - loss: 0.6285 - acc: 0.6421 - val_loss: 0.6443 - val_acc: 0.6234
Epoch 89/100
 - 2s - loss: 0.6282 - acc: 0.6436 - val_loss: 0.6461 - val_acc: 0.6270
Epoch 90/100
 - 2s - loss: 0.6275 - acc: 0.6461 - val_loss: 0.6461 - val_acc: 0.6232
Epoch 91/100
 - 2s - loss: 0.6258 - acc: 0.6469 - val_loss: 0.6690 - val_acc: 0.6030
Epoch 92/100
 - 2s - loss: 0.6271 - acc: 0.6445 - val_loss: 0.6466 - val_acc: 0.6206
Epoch 93/100
 - 2s - loss: 0.6294 - acc: 0.6428 - val_loss: 0.6448 - val_acc: 0.6262
Epoch 94/100
 - 2s - loss: 0.6269 - acc: 0.6446 - val_loss: 0.6463 - val_acc: 0.6320
Epoch 95/100
 - 2s - loss: 0.6268 - acc: 0.6434 - val_loss: 0.6493 - val_acc: 0.6244
Epoch 96/100
 - 2s - loss: 0.6272 - acc: 0.6452 - val_loss: 0.6487 - val_acc: 0.6196
Epoch 97/100
 - 2s - loss: 0.6274 - acc: 0.6453 - val_loss: 0.6524 - val_acc: 0.6160
Epoch 98/100
 - 2s - loss: 0.6266 - acc: 0.6461 - val_loss: 0.6554 - val_acc: 0.6098
Epoch 99/100
 - 2s - loss: 0.6254 - acc: 0.6475 - val_loss: 0.6476 - val_acc: 0.6196
Epoch 100/100
 - 2s - loss: 0.6253 - acc: 0.6496 - val_loss: 0.6456 - val_acc: 0.6256
CPU times: user 3min 47s, sys: 25.9 s, total: 4min 13s
Wall time: 3min 11s
```

In [ ]:

```
loss, acc = model.evaluate(x_test_, y_test, verbose=0)
print('Test Accuracy: %f' % (acc*100))
```

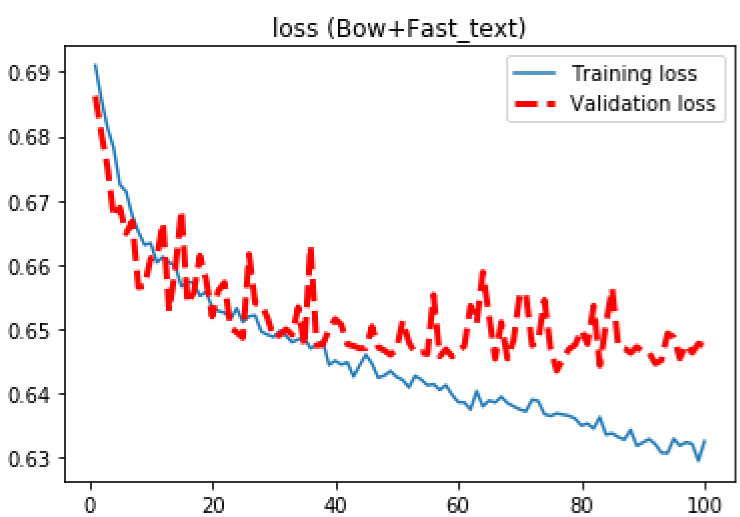
Test Accuracy: 66.039997

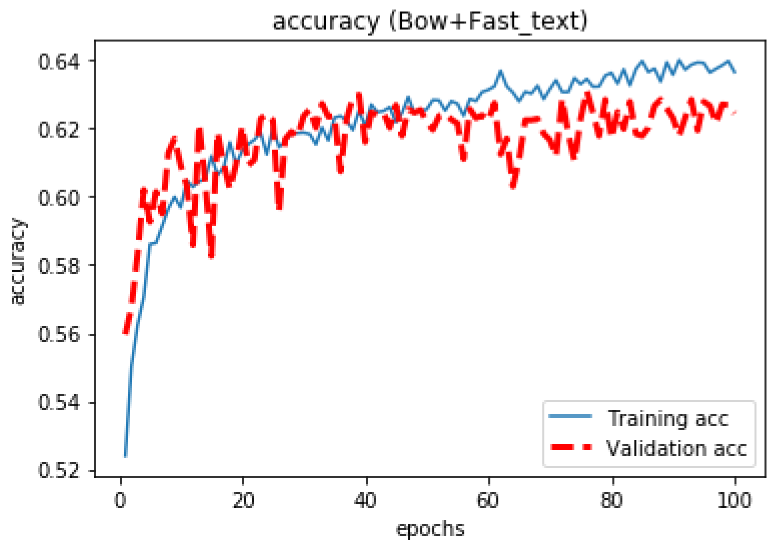
In [ ]:
```
loss, acc = model.evaluate(x_train_, y_train, verbose=0)
print('Test Accuracy: %f' % (acc*100))
```

Test Accuracy: 65.008003

Plots

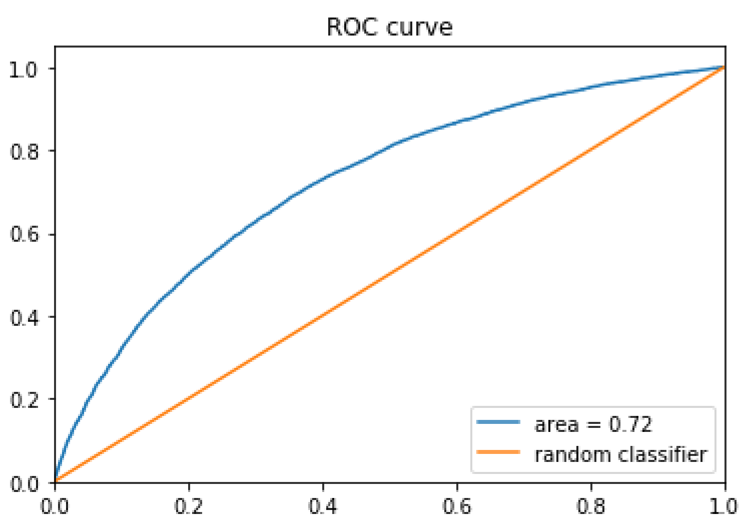
In [ ]:
```
acc = model.history.history['acc']
val_acc = model.history.history['val_acc']
loss = model.history.history['loss']
val_loss = model.history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.figure()
plt.plot(epochs, loss, label='Training loss')
plt.plot(epochs, val_loss, 'r--', label='Validation loss', linewidth=3)
plt.title('loss (Bow+Fast_text)')
plt.legend()
plt.savefig("Training and Validation loss (Bow+Fast_text).eps", format='eps', dpi=1

plt.show()

plt.figure()
plt.plot(epochs, acc, label='Training acc')
plt.plot(epochs, val_acc,'r--', label='Validation acc', linewidth=3)
plt.title('accuracy (Bow+Fast_text)')
plt.legend()
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.savefig("Training and validation accuracy (Bow+Fast_text).eps", format='eps', d
plt.show()
```





In [ ]:

```
#Create ROC curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
pred_probas = model.predict_proba(x_test_)[:,0]
fpr,tpr,_ = roc_curve(y_test, pred_probas)
roc_auc = auc(fpr,tpr)
plt.plot(fpr,tpr,label='area = %.2f' %roc_auc)
plt.plot([0, 1], [0, 1], label="random classifier")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.legend(loc='lower right')
plt.title("ROC curve")
plt.savefig("ROC curve (Bow+Fast_text).eps", format='eps', dpi=1200)
plt.show()
```



In [ ]:

```
!ls
```

```
 adc.json
'Confusion Matrix (tf_idf).eps'
 model_fasttext.h5
 model_tf-idf.h5
'ROC curve (Bow+Fast_text).eps'
'ROC curve.eps'
'ROC curve (tf_idf).eps'
 sample_data
'Training and validation accuracy (Bow+Fast_text).eps'
'Training and validation accuracy (tf-df).eps'
'Training and Validation loss (Bow+Fast_text).eps'
'Training and Validation loss (tf-df).eps'
 wiki-news-300d-1M-subword.vec
 wiki-news-300d-1M-subword.vec.zip
```

In [ ]:
```
files.download("Training and validation accuracy (Bow+Fast_text).eps")
files.download("Training and Validation loss (Bow+Fast_text).eps")
files.download("ROC curve (Bow+Fast_text).eps")
```

In [ ]:
```

```

Confusion matrix

We can visually check the confusion matrix to understand the distribution of misclassifications.

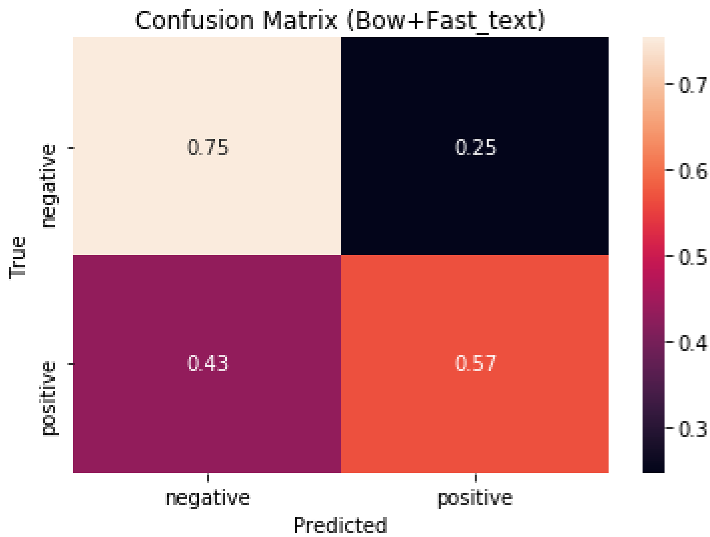
In [ ]:
```
res=model.predict(x_test_)
res=res.reshape(1, -1)[0]
res_ = [int(np.round(x)) for x in res]
```

In [ ]:
```
counter = np.count_nonzero(y_pred==y_test)
counter/len(y_test)
```

In [ ]:
```
y_pred = np.round(model.predict(x_test_)); y_pred

LABELS = [
    "negative", "positive"
]

# Create a confusion matrix on training data.
with tf.Graph().as_default():
  cm = tf.confusion_matrix(y_test, y_pred)
  with tf.Session() as session:
    cm_out = session.run(cm)

# Normalize the confusion matrix so that each row sums to 1.
cm_out = cm_out.astype(float) / cm_out.sum(axis=1)[:, np.newaxis]

sns.heatmap(cm_out, annot=True, xticklabels=LABELS, yticklabels=LABELS);
plt.xlabel("Predicted");
plt.ylabel("True")
plt.title("Confusion Matrix (Bow+Fast_text)")
```

```
plt.savefig("Confusion Matrix (Bow+Fast_text).eps", format='eps', dpi=1200)
plt.plot()
```

Out[ ]: []



In [ ]:
```
files.download("Confusion Matrix (Bow+Fast_text).eps")
```

Saving the entire model and save in the drive

In [ ]:
```
# Save entire model to a HDF5 file
model.save('model_fasttext.h5')
```

In [ ]:
```
!ls
```

```
'Confusion Matrix (tf_idf).eps'    sample_data
 model_tf-idf.h5                  'Training and validation accuracy (tf-df).eps'
'ROC curve.eps'                   'Training and Validation loss (tf-df).eps'
'ROC curve (tf_idf).eps'
```

In [ ]:
```
files.download( "model_fasttext.h5" )
```

In [ ]:
```
!ls
```

```
adc.json   sample_data                    wiki-news-300d-1M-subword.vec.zip
gdrive     wiki-news-300d-1M-subword.vec
```

TF_IDF

In [ ]:
```
%%time

model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(512, input_shape=(tfidf_train.shape[1], ), activati
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

```
model.summary()

class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('val_acc')>0.66):
      print("\nReached 90% val_acc so cancelling training!")
      self.model.stop_training = True

callbacks = myCallback()

monitor = tf.keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=1e-3, pati

model.compile(loss='binary_crossentropy', optimizer=tf.keras.optimizers.Adamax(0.00

# fit network
model.fit(tfidf_train, train_df.sentiment,
          epochs=100,
          batch_size=64, verbose=2,
          validation_split=0.2)#, callbacks=[monitor])



#           epochs=2000,
#           batch_size=32, verbose=2,
#           validation_split=0.2, callbacks=[monitor])

loss, acc = model.evaluate(tfidf_tst, test_df.sentiment, verbose=0)
print('Test Accuracy: %f' % (acc*100))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 512)               10240512
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 10,766,337
Trainable params: 10,766,337
Non-trainable params: 0
_________________________________________________________________
Train on 20000 samples, validate on 5000 samples
Epoch 1/100
 - 8s - loss: 0.3369 - acc: 0.8618 - val_loss: 0.2672 - val_acc: 0.8960
Epoch 2/100
 - 7s - loss: 0.1137 - acc: 0.9589 - val_loss: 0.3398 - val_acc: 0.8978
Epoch 3/100
 - 7s - loss: 0.0330 - acc: 0.9889 - val_loss: 0.5157 - val_acc: 0.8966
Epoch 4/100
 - 7s - loss: 0.0093 - acc: 0.9974 - val_loss: 0.6161 - val_acc: 0.8952
Epoch 5/100
 - 7s - loss: 0.0060 - acc: 0.9987 - val_loss: 0.6852 - val_acc: 0.8960
Epoch 6/100
 - 7s - loss: 0.0018 - acc: 0.9994 - val_loss: 0.7886 - val_acc: 0.8922
Epoch 7/100
 - 7s - loss: 0.0030 - acc: 0.9991 - val_loss: 0.7765 - val_acc: 0.8984
Epoch 8/100
 - 7s - loss: 0.0026 - acc: 0.9993 - val_loss: 0.7780 - val_acc: 0.8942
Epoch 9/100
 - 7s - loss: 0.0013 - acc: 0.9998 - val_loss: 0.8012 - val_acc: 0.8946
Epoch 10/100
 - 7s - loss: 0.0012 - acc: 0.9997 - val_loss: 0.8453 - val_acc: 0.8948
Epoch 11/100
 - 7s - loss: 0.0011 - acc: 0.9998 - val_loss: 0.8181 - val_acc: 0.8942
Epoch 12/100
 - 7s - loss: 7.9785e-04 - acc: 0.9998 - val_loss: 0.8214 - val_acc: 0.8984
Epoch 13/100
 - 7s - loss: 2.3640e-04 - acc: 0.9998 - val_loss: 0.8624 - val_acc: 0.8980
Epoch 14/100
 - 7s - loss: 0.0017 - acc: 0.9996 - val_loss: 0.9277 - val_acc: 0.8892
Epoch 15/100
 - 7s - loss: 5.9793e-04 - acc: 0.9998 - val_loss: 0.8994 - val_acc: 0.8960
Epoch 16/100
 - 7s - loss: 0.0011 - acc: 0.9999 - val_loss: 0.8839 - val_acc: 0.8956
Epoch 17/100
 - 7s - loss: 3.1154e-04 - acc: 0.9999 - val_loss: 0.9185 - val_acc: 0.8990
Epoch 18/100
 - 7s - loss: 8.8152e-04 - acc: 0.9998 - val_loss: 0.9511 - val_acc: 0.8968
Epoch 19/100
 - 7s - loss: 0.0035 - acc: 0.9994 - val_loss: 0.8385 - val_acc: 0.8980
```

```
Epoch 20/100
 - 6s - loss: 2.9463e-04 - acc: 0.9999 - val_loss: 0.8554 - val_acc: 0.9004
Epoch 21/100
 - 6s - loss: 1.6112e-04 - acc: 0.9999 - val_loss: 0.9021 - val_acc: 0.8966
Epoch 22/100
 - 6s - loss: 2.2117e-05 - acc: 1.0000 - val_loss: 0.9318 - val_acc: 0.8990
Epoch 23/100
 - 7s - loss: 7.0926e-04 - acc: 0.9998 - val_loss: 0.9512 - val_acc: 0.8998
Epoch 24/100
 - 7s - loss: 0.0013 - acc: 0.9999 - val_loss: 0.9834 - val_acc: 0.8938
Epoch 25/100
 - 7s - loss: 0.0012 - acc: 0.9998 - val_loss: 0.9983 - val_acc: 0.8970
Epoch 26/100
 - 7s - loss: 0.0021 - acc: 0.9997 - val_loss: 0.9302 - val_acc: 0.8958
Epoch 27/100
 - 7s - loss: 1.7721e-04 - acc: 0.9999 - val_loss: 0.9671 - val_acc: 0.8938
Epoch 28/100
 - 7s - loss: 3.0822e-04 - acc: 0.9998 - val_loss: 1.1473 - val_acc: 0.8834
Epoch 29/100
 - 7s - loss: 0.0011 - acc: 0.9998 - val_loss: 0.9919 - val_acc: 0.8944
Epoch 30/100
 - 7s - loss: 0.0014 - acc: 0.9998 - val_loss: 0.9746 - val_acc: 0.8918
Epoch 31/100
 - 7s - loss: 8.6304e-04 - acc: 0.9998 - val_loss: 0.9593 - val_acc: 0.8942
Epoch 32/100
 - 7s - loss: 1.7316e-04 - acc: 0.9999 - val_loss: 0.9736 - val_acc: 0.8942
Epoch 33/100
Epoch 34/100
 - 7s - loss: 5.1193e-05 - acc: 1.0000 - val_loss: 1.0183 - val_acc: 0.8930
Epoch 35/100
 - 7s - loss: 6.3139e-04 - acc: 0.9999 - val_loss: 1.0127 - val_acc: 0.8914
Epoch 36/100
 - 7s - loss: 0.0022 - acc: 0.9997 - val_loss: 1.0349 - val_acc: 0.8902
Epoch 37/100
 - 7s - loss: 0.0015 - acc: 0.9995 - val_loss: 1.0221 - val_acc: 0.8890
Epoch 38/100
 - 7s - loss: 1.4370e-04 - acc: 0.9999 - val_loss: 1.0375 - val_acc: 0.8892
Epoch 39/100
 - 7s - loss: 5.6880e-04 - acc: 0.9998 - val_loss: 1.1447 - val_acc: 0.8802
Epoch 40/100
 - 7s - loss: 0.0014 - acc: 0.9998 - val_loss: 1.0963 - val_acc: 0.8788
Epoch 41/100
 - 7s - loss: 0.0017 - acc: 0.9998 - val_loss: 1.1475 - val_acc: 0.8672
Epoch 42/100
 - 7s - loss: 0.0065 - acc: 0.9989 - val_loss: 0.9781 - val_acc: 0.8804
Epoch 43/100
 - 7s - loss: 0.0048 - acc: 0.9986 - val_loss: 0.9061 - val_acc: 0.8802
Epoch 44/100
 - 7s - loss: 0.0058 - acc: 0.9989 - val_loss: 0.9336 - val_acc: 0.8788
Epoch 45/100
 - 7s - loss: 0.0037 - acc: 0.9994 - val_loss: 0.9883 - val_acc: 0.8780
Epoch 46/100
 - 7s - loss: 0.0019 - acc: 0.9995 - val_loss: 1.0249 - val_acc: 0.8778
Epoch 47/100
 - 6s - loss: 0.0032 - acc: 0.9997 - val_loss: 1.1694 - val_acc: 0.8752
Epoch 48/100
```

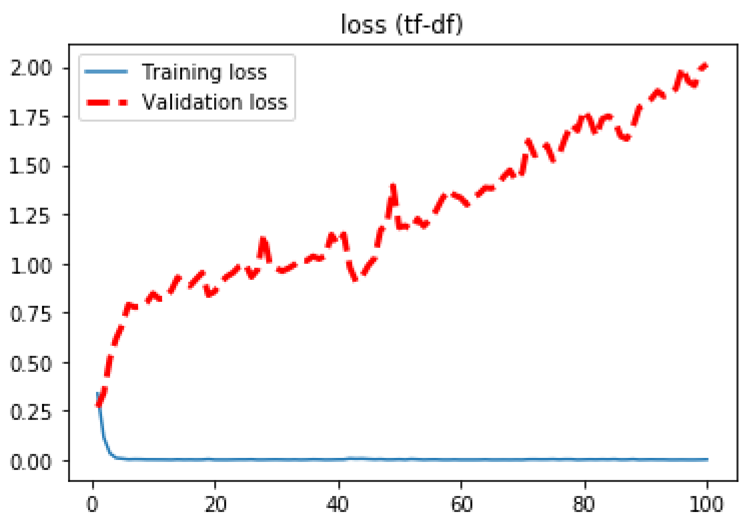
```
 - 7s - loss: 8.3272e-04 - acc: 0.9998 - val_loss: 1.1949 - val_acc: 0.8696
Epoch 49/100
 - 7s - loss: 0.0011 - acc: 0.9998 - val_loss: 1.3930 - val_acc: 0.8434
Epoch 50/100
 - 6s - loss: 0.0025 - acc: 0.9995 - val_loss: 1.1824 - val_acc: 0.8756
Epoch 51/100
 - 6s - loss: 7.2852e-04 - acc: 0.9999 - val_loss: 1.1900 - val_acc: 0.8760
Epoch 52/100
 - 7s - loss: 0.0038 - acc: 0.9995 - val_loss: 1.1710 - val_acc: 0.8762
Epoch 53/100
 - 7s - loss: 0.0021 - acc: 0.9994 - val_loss: 1.2258 - val_acc: 0.8698
Epoch 54/100
 - 6s - loss: 3.2281e-04 - acc: 0.9999 - val_loss: 1.1885 - val_acc: 0.8730
Epoch 55/100
 - 6s - loss: 0.0013 - acc: 0.9998 - val_loss: 1.2211 - val_acc: 0.8718
Epoch 56/100
 - 6s - loss: 0.0016 - acc: 0.9998 - val_loss: 1.2672 - val_acc: 0.8626
Epoch 57/100
 - 6s - loss: 1.0581e-04 - acc: 1.0000 - val_loss: 1.3278 - val_acc: 0.8614
Epoch 58/100
 - 6s - loss: 2.4040e-05 - acc: 1.0000 - val_loss: 1.3712 - val_acc: 0.8590
Epoch 59/100
 - 6s - loss: 8.9596e-04 - acc: 0.9998 - val_loss: 1.3455 - val_acc: 0.8580
Epoch 60/100
 - 6s - loss: 3.1944e-05 - acc: 1.0000 - val_loss: 1.3329 - val_acc: 0.8580
Epoch 61/100
 - 6s - loss: 5.7467e-05 - acc: 0.9999 - val_loss: 1.2981 - val_acc: 0.8702
Epoch 62/100
 - 6s - loss: 7.5426e-06 - acc: 1.0000 - val_loss: 1.3322 - val_acc: 0.8662
Epoch 63/100
 - 6s - loss: 2.1823e-04 - acc: 0.9999 - val_loss: 1.3525 - val_acc: 0.8638
Epoch 64/100
 - 6s - loss: 0.0013 - acc: 0.9998 - val_loss: 1.3840 - val_acc: 0.8672
Epoch 65/100
 - 6s - loss: 1.0146e-06 - acc: 1.0000 - val_loss: 1.3805 - val_acc: 0.8682
Epoch 66/100
 - 6s - loss: 1.2118e-06 - acc: 1.0000 - val_loss: 1.3948 - val_acc: 0.8680
Epoch 67/100
 - 6s - loss: 2.9641e-04 - acc: 0.9999 - val_loss: 1.4402 - val_acc: 0.8656
Epoch 68/100
 - 6s - loss: 5.9403e-05 - acc: 0.9999 - val_loss: 1.4727 - val_acc: 0.8648
Epoch 69/100
 - 7s - loss: 5.2386e-04 - acc: 0.9999 - val_loss: 1.4214 - val_acc: 0.8650
Epoch 70/100
 - 7s - loss: 1.9603e-04 - acc: 0.9999 - val_loss: 1.4617 - val_acc: 0.8662
Epoch 71/100
 - 7s - loss: 0.0022 - acc: 0.9998 - val_loss: 1.6225 - val_acc: 0.8558
Epoch 72/100
 - 7s - loss: 0.0024 - acc: 0.9998 - val_loss: 1.5492 - val_acc: 0.8636
Epoch 73/100
 - 7s - loss: 0.0018 - acc: 0.9998 - val_loss: 1.5602 - val_acc: 0.8618
Epoch 74/100
 - 7s - loss: 0.0020 - acc: 0.9998 - val_loss: 1.6006 - val_acc: 0.8494
Epoch 75/100
 - 7s - loss: 0.0031 - acc: 0.9995 - val_loss: 1.5246 - val_acc: 0.8610
Epoch 76/100
```

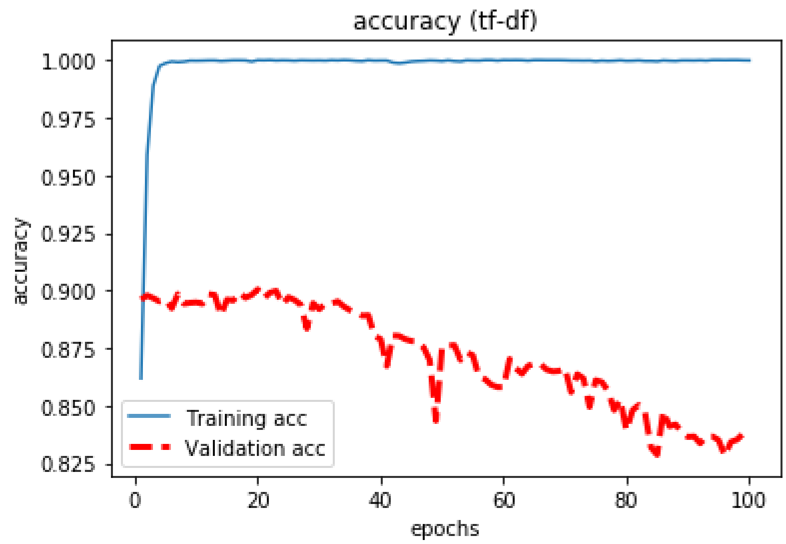
```
 - 7s - loss: 0.0014 - acc: 0.9998 - val_loss: 1.5526 - val_acc: 0.8602
Epoch 77/100
 - 6s - loss: 0.0027 - acc: 0.9996 - val_loss: 1.6377 - val_acc: 0.8564
Epoch 78/100
 - 6s - loss: 0.0025 - acc: 0.9997 - val_loss: 1.7071 - val_acc: 0.8480
Epoch 79/100
 - 7s - loss: 2.7805e-04 - acc: 0.9999 - val_loss: 1.6750 - val_acc: 0.8526
Epoch 80/100
 - 7s - loss: 0.0011 - acc: 0.9997 - val_loss: 1.7770 - val_acc: 0.8392
Epoch 81/100
 - 7s - loss: 0.0021 - acc: 0.9997 - val_loss: 1.7334 - val_acc: 0.8480
Epoch 82/100
 - 7s - loss: 0.0014 - acc: 0.9998 - val_loss: 1.6461 - val_acc: 0.8502
Epoch 83/100
 - 7s - loss: 0.0024 - acc: 0.9996 - val_loss: 1.7337 - val_acc: 0.8472
Epoch 84/100
 - 6s - loss: 0.0017 - acc: 0.9996 - val_loss: 1.7479 - val_acc: 0.8318
Epoch 85/100
 - 7s - loss: 0.0030 - acc: 0.9995 - val_loss: 1.7283 - val_acc: 0.8286
Epoch 86/100
 - 7s - loss: 7.0453e-04 - acc: 0.9998 - val_loss: 1.6466 - val_acc: 0.8476
Epoch 87/100
 - 7s - loss: 0.0010 - acc: 0.9997 - val_loss: 1.6321 - val_acc: 0.8410
Epoch 88/100
 - 7s - loss: 0.0030 - acc: 0.9996 - val_loss: 1.6919 - val_acc: 0.8418
Epoch 89/100
 - 7s - loss: 4.4214e-04 - acc: 0.9998 - val_loss: 1.7913 - val_acc: 0.8384
Epoch 90/100
 - 7s - loss: 9.6882e-04 - acc: 0.9998 - val_loss: 1.8123 - val_acc: 0.8364
Epoch 91/100
 - 7s - loss: 0.0016 - acc: 0.9998 - val_loss: 1.8367 - val_acc: 0.8366
Epoch 92/100
 - 7s - loss: 9.9077e-04 - acc: 0.9999 - val_loss: 1.8755 - val_acc: 0.8336
Epoch 93/100
 - 7s - loss: 0.0010 - acc: 0.9998 - val_loss: 1.8494 - val_acc: 0.8366
Epoch 94/100
 - 7s - loss: 8.7694e-06 - acc: 1.0000 - val_loss: 1.8602 - val_acc: 0.8366
Epoch 95/100
 - 7s - loss: 6.3414e-06 - acc: 1.0000 - val_loss: 1.8890 - val_acc: 0.8350
Epoch 96/100
 - 7s - loss: 3.8481e-04 - acc: 0.9999 - val_loss: 1.9959 - val_acc: 0.8282
Epoch 97/100
 - 7s - loss: 5.6698e-06 - acc: 1.0000 - val_loss: 1.9246 - val_acc: 0.8342
Epoch 98/100
 - 7s - loss: 2.8885e-05 - acc: 1.0000 - val_loss: 1.9042 - val_acc: 0.8352
Epoch 99/100
 - 7s - loss: 5.2961e-04 - acc: 0.9999 - val_loss: 1.9869 - val_acc: 0.8378
Epoch 100/100
 - 7s - loss: 0.0016 - acc: 0.9998 - val_loss: 2.0131 - val_acc: 0.8366
Test Accuracy: 79.824001
CPU times: user 9min 56s, sys: 2min 16s, total: 12min 12s
Wall time: 11min 32s
```

Plots

In [ ]:

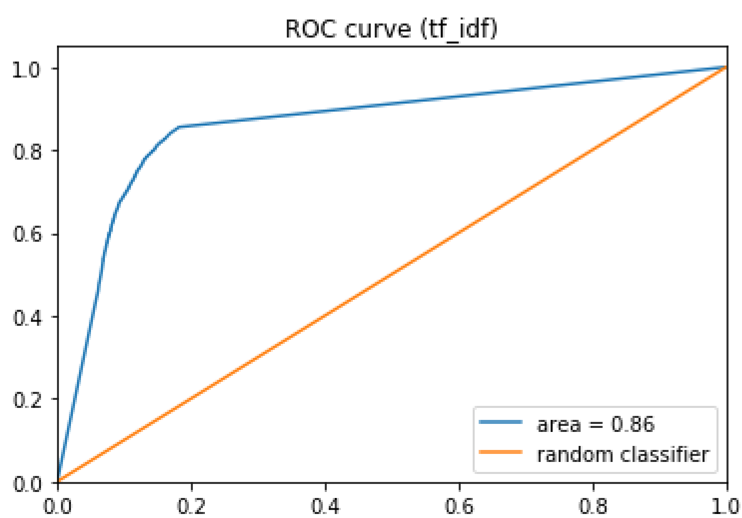
```
acc = model.history.history['acc']
val_acc = model.history.history['val_acc']
loss = model.history.history['loss']
val_loss = model.history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.figure()
plt.plot(epochs, loss, label='Training loss')
plt.plot(epochs, val_loss, 'r--',label='Validation loss',  linewidth=3)
plt.title('loss (tf-df)')
plt.legend()
plt.savefig("Training and Validation loss (tf-df).eps", format='eps', dpi=1200)
plt.show()

plt.figure()
plt.plot(epochs, acc, label='Training acc')
plt.plot(epochs, val_acc,'r--', label='Validation acc', linewidth=3)
plt.title('accuracy (tf-df)')
plt.legend()
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.savefig("Training and validation accuracy (tf-df).eps", format='eps', dpi=1200)
plt.show()
```





In [ ]:

```
#Create ROC curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
pred_probas = model.predict_proba(tfidf_tst)[:,0]
fpr,tpr,_ = roc_curve(test_df.sentiment, pred_probas)
roc_auc = auc(fpr,tpr)
plt.plot(fpr,tpr,label='area = %.2f' %roc_auc)
plt.plot([0, 1], [0, 1], label="random classifier")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.legend(loc='lower right')
plt.title("ROC curve (tf_idf)")
plt.savefig("ROC curve (tf_idf).eps", format='eps', dpi=1200)
plt.show()
```

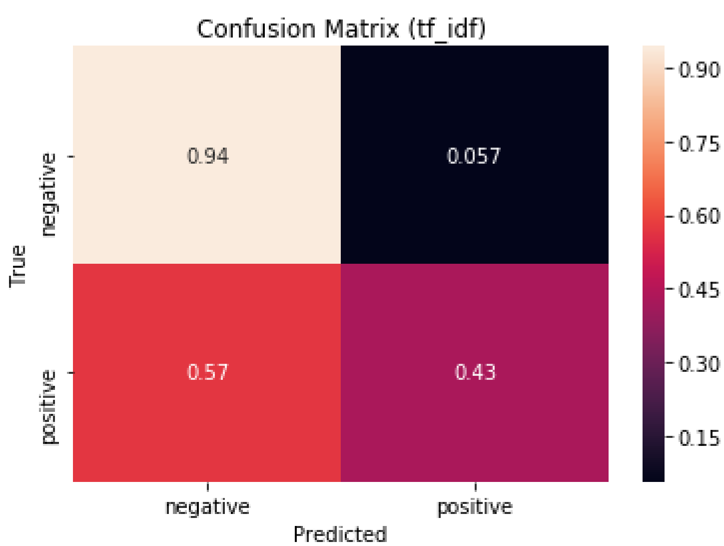


In [ ]:

```
model.predict(tfidf_tst[:,:])
```

Out[ ]:
```
array([[9.9999994e-01],
       [2.9802322e-08],
       [0.0000000e+00],
       ...,
       [0.0000000e+00],
       [3.5142303e-03],
       [0.0000000e+00]], dtype=float32)
```

In [ ]:
```python
LABELS = [
    "negative", "positive"
]

# Create a confusion matrix on training data.
with tf.Graph().as_default():
  cm = tf.confusion_matrix(test_df.sentiment,
                           model.predict(tfidf_tst[:,:]))
  with tf.Session() as session:
    cm_out = session.run(cm)

# Normalize the confusion matrix so that each row sums to 1.
cm_out = cm_out.astype(float) / cm_out.sum(axis=1)[:, np.newaxis]

sns.heatmap(cm_out, annot=True, xticklabels=LABELS, yticklabels=LABELS);
plt.title("Confusion Matrix (tf_idf)")
plt.xlabel("Predicted");
plt.ylabel("True");
plt.savefig("Confusion Matrix (tf_idf).eps", format='eps', dpi=1200)
plt.show()
```



In [ ]:
```python
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
```

In [ ]:
```python
(0.95+0.28)/(0.95+0.28+0.049+0.72)
```

Out[ ]:
```
0.6153076538269134
```

In [ ]:
```python
# accuracy = accuracy_score(tfidf_tst, test_df.sentiment)
# print('Accuracy: %f' % accuracy)
```

```
# precision tp / (tp + fp)
# precision = precision_score(tfidf_tst, test_df.sentiment)
# print('Precision: %f' % precision)

# recall: tp / (tp + fn)
recall = recall_score(tfidf_tst, test_df.sentiment)
print('Recall: %f' % recall)
```

In [ ]:
```
!ls
```

```
'Confusion Matrix (tf_idf).eps'   sample_data
 model_tf-idf.h5                 'Training and validation accuracy (tf-df).eps'
'ROC curve.eps'                  'Training and Validation loss (tf-df).eps'
'ROC curve (tf_idf).eps'
```

In [ ]:
```
files.download('Training and validation accuracy (tf-df).eps')
files.download("Training and Validation loss (tf-df).eps")
files.download("ROC curve (tf_idf).eps")
files.download("Confusion Matrix (tf_idf).eps")

files.download('model_tf-idf.h5')
```

Small Example

A small example for understanding of how Tokenize work

Word level

In [ ]:
```
samples = ['The cat sat on the mat.', 'The dog ate my homework.']
samples_essay= ['Salomon likes to learn new things. Djeff likes that too',
            'Salomon also likes to learn Karateka']
sample_wiki = ["John likes to watch movies. Mary likes movies too.",
               "John also likes to watch football games."]

sample_presentation =["That which we call a rose by any other name would smell as w
                      "Good night sweet prince.",
                      "Parting is such sweet sorrow."]


tokenizer = Tokenizer()
tokenizer.fit_on_texts(sample_presentation)
sequences = tokenizer.texts_to_sequences(sample_presentation)
bag_of_word_results = tokenizer.texts_to_matrix(sample_presentation, mode='freq')
tf_idf_results = tokenizer.texts_to_matrix(sample_presentation, mode='tfidf')

word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

```
Found 22 unique tokens.
```

In [ ]:
```
sequences
```

Out[ ]:
```
[[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15],
 [16, 17, 1, 18],
 [19, 20, 21, 1, 22]]
```

In [ ]:
```
bag_of_word_results
```

Out[ ]:
```
array([[0.        , 0.        , 0.07142857, 0.07142857, 0.07142857,
        0.07142857, 0.07142857, 0.07142857, 0.07142857, 0.07142857,
        0.07142857, 0.07142857, 0.07142857, 0.07142857, 0.07142857,
        0.07142857, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.25      , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.25      , 0.25      , 0.25      , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.2       , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.2       ,
        0.2       , 0.2       , 0.2       ]])
```

In [ ]:
```
tf_idf_results
```

Out[ ]:
```
array([[0.        , 0.        , 0.91629073, 0.91629073, 0.91629073,
        0.91629073, 0.91629073, 0.91629073, 0.91629073, 0.91629073,
        0.91629073, 0.91629073, 0.91629073, 0.91629073, 0.91629073,
        0.91629073, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.69314718, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.91629073, 0.91629073, 0.91629073, 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.69314718, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.91629073,
        0.91629073, 0.91629073, 0.91629073]])
```

In [ ]:
```
len(word_index)
```

Out[ ]:
```
22
```

In [ ]:
```
bag_of_word_results.shape
```

Out[ ]:
```
(2, 12)
```

In [ ]:
```
# results
```

Caracter level

In [ ]:
```
samples = ['The cat sat on the mat.', 'The dog ate my homework.']
```

```python
tokenizer = Tokenizer(char_level=True)
tokenizer.fit_on_texts(samples)
sequences = tokenizer.texts_to_sequences(samples)
one_hot_results = tokenizer.texts_to_matrix(samples, mode='binary')
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

```
Found 17 unique tokens.
```

In [ ]:
```python
sequences
```

Out[ ]:
```
[[2, 4, 3, 1, 9, 5, 2, 1, 10, 5, 2, 1, 6, 11, 1, 2, 4, 3, 1, 7, 5, 2, 8],
 [2,
  4,
  3,
  1,
  12,
  6,
  13,
  1,
  5,
  2,
  3,
  1,
  7,
  14,
  1,
  4,
  6,
  7,
  3,
  15,
  6,
  16,
  17,
  8]]
```

In [ ]:
```python
one_hot_results
```

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

In [ ]:
```python
word_index
```

```
Out[ ]: {' ': 1,
 't': 2,
 'e': 3,
 'h': 4,
 'a': 5,
 'o': 6,
 'm': 7,
 '.': 8,
 'c': 9,
 's': 10,
 'n': 11,
 'd': 12,
 'g': 13,
 'y': 14,
 'w': 15,
 'r': 16,
 'k': 17}
```

hashing trick

```
In [ ]: samples = ['The cat sat on the mat.', 'The dog ate my homework.']

dimensionality = 1000
max_length = 10

results = np.zeros((len(samples), max_length, dimensionality))
for i, sample in enumerate(samples):
    for j, word in list(enumerate(sample.split()))[:max_length]:
#         Hashes the word into a
#         random integer index
#         between 0 and 1,000
        index = abs(hash(word)) % dimensionality
        results[i, j, index] = 1.
```

About the Authors:

Salomon Kabongo KABENAMUALU, Master degree student at the African Institute for mathematical SCiences (AIMS South Africa) his research focused on the use machine learning technique in the field of Natural Language Processing.

References : How to Develop a Deep Learning Bag-of-Words Model for Predicting Movie Review Sentiment