Sentiment Analysis - Text Classification with Universal Embeddings

Textual data in spite of being highly unstructured, can be classified into two major types of documents.

- **Factual documents** which typically depict some form of statements or facts with no specific feelings or emotion attached to them. These are also known as objective documents.
- **Subjective documents** on the other hand have text which expresses feelings, mood, emotions and opinion.

Sentiment Analysis is also popularly known as opinion analysis or opinion mining. The key idea is to use techniques from text analytics, NLP, machine learning and linguistics to extract important information or data points from unstructured text. This in turn can help us derive the sentiment from text data

Here we will be looking at building supervised sentiment analysis classification models thanks to the advantage of labeled data! The dataset we will be working with is the IMDB Large Movie Review Dataset having 50000 reviews classified into positive and negative sentiment. I have provided a compressed version of the dataset in this repository itself for your benefit!

Do remember that the focus here is not sentiment analysis but text classification by leveraging universal sentence embeddings.

We will leverage the following sentence encoders here for demonstration from TensorFlow Hub (https://tfhub.dev/):

- **Neural-Net Language Model (nnlm-en-dim128)** (https://tfhub.dev/google/nnlm-en-dim128/1)
- **Universal Sentence Encoder (universal-sentence-encoder)** (https://tfhub.dev/google/universal-sentence-encoder/2)

Developed by Dipanjan (DJ) Sarkar (https://www.linkedin.com/in/dipanzan/)

Install Tensorflow Hub

In [1]:

```
!pip install tensorflow-hub
```

```
Requirement already satisfied: tensorflow-hub in c:\users\finan\appdata\loc
al\programs\python\python37\lib\site-packages (0.4.0)
Requirement already satisfied: six>=1.10.0 in c:\users\finan\appdata\local
\programs\python\python37\lib\site-packages (from tensorflow-hub) (1.12.0)
Requirement already satisfied: protobuf>=3.4.0 in c:\users\finan\appdata\lo
cal\programs\python\python37\lib\site-packages (from tensorflow-hub) (3.7.
0)
Requirement already satisfied: numpy>=1.12.0 in c:\users\finan\appdata\loca
l\programs\python\python37\lib\site-packages (from tensorflow-hub) (1.16.2)
Requirement already satisfied: setuptools in c:\users\finan\appdata\local\p
rograms\python\python37\lib\site-packages (from protobuf>=3.4.0->tensorflow
-hub) (41.0.0)

WARNING: You are using pip version 19.1, however version 19.1.1 is availabl
e.
You should consider upgrading via the 'python -m pip install --upgrade pip'
command.
```

Load up Dependencies

In [2]:

```
import tensorflow as tf
import tensorflow_hub as hub
import numpy as np
import pandas as pd
```

```
WARNING: Logging before flag parsing goes to stderr.
W0510 10:32:24.765622  9676 __init__.py:56] Some hub symbols are not availa
ble because TensorFlow version is less than 1.14
```

Check if GPU is available for use!

In [3]:

```
tf.test.is_gpu_available()
```

Out[3]:

```
False
```

In [4]:

```
tf.test.gpu_device_name()
```

Out[4]:

```
''
```

Load and View Dataset

In [5]:

```
dataset = pd.read_csv('movie_reviews.csv.bz2', compression='bz2')
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
review       50000 non-null object
sentiment    50000 non-null object
dtypes: object(2)
memory usage: 781.3+ KB
```

In [6]:

```
dataset['sentiment'] = [1 if sentiment == 'positive' else 0 for sentiment in
dataset.head()
```

Out[6]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. <br /><br />The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

Build train, validation and test datasets

In [7]:

```
reviews = dataset['review'].values
sentiments = dataset['sentiment'].values

train_reviews = reviews[:30000]
train_sentiments = sentiments[:30000]

val_reviews = reviews[30000:35000]
val_sentiments = sentiments[30000:35000]

test_reviews = reviews[35000:]
test_sentiments = sentiments[35000:]
train_reviews.shape, val_reviews.shape, test_reviews.shape
```

Out[7]:

```
((30000,), (5000,), (15000,))
```

Basic Text Wrangling

In [8]:

```
!pip install contractions
!pip install beautifulsoup4
```

```
Requirement already satisfied: contractions in c:\users\finan\appdata\local
\programs\python\python37\lib\site-packages (0.0.18)

WARNING: You are using pip version 19.1, however version 19.1.1 is availabl
e.
You should consider upgrading via the 'python -m pip install --upgrade pip'
command.

Requirement already satisfied: beautifulsoup4 in c:\users\finan\appdata\loc
al\programs\python\python37\lib\site-packages (4.7.1)
Requirement already satisfied: soupsieve>=1.2 in c:\users\finan\appdata\loc
al\programs\python\python37\lib\site-packages (from beautifulsoup4) (1.8)

WARNING: You are using pip version 19.1, however version 19.1.1 is availabl
e.
You should consider upgrading via the 'python -m pip install --upgrade pip'
command.
```

In [ ]:

```
import contractions

from bs4 import BeautifulSoup
import unicodedata
import re


def strip_html_tags(text):
    soup = BeautifulSoup(text, "html.parser")
    [s.extract() for s in soup(['iframe', 'script'])]
    stripped_text = soup.get_text()
    stripped_text = re.sub(r'[\r|\n|\r\n]+', '\n', stripped_text)
    return stripped_text


def remove_accented_chars(text):
    text = unicodedata.normalize('NFKD', text).encode('ascii', 'ignore').deco
    return text


def expand_contractions(text):
    return contractions.fix(text)



def remove_special_characters(text, remove_digits=False):
    pattern = r'[^a-zA-Z0-9\s]' if not remove_digits else r'[^a-zA-Z\s]'
    text = re.sub(pattern, '', text)
    return text


def pre_process_document(document):

    # strip HTML
    document = strip_html_tags(document)

    # lower case
    document = document.lower()

    # remove extra newlines (often might be present in really noisy text)
    document = document.translate(document.maketrans("\n\t\r", "   "))

    # remove accented characters
    document = remove_accented_chars(document)

    # expand contractions
    document = expand_contractions(document)

    # remove special characters and\or digits
    # insert spaces between special characters to isolate them
    special_char_pattern = re.compile(r'([{.(-)!}])')
    document = special_char_pattern.sub(" \\1 ", document)
    document = remove_special_characters(document, remove_digits=True)

    # remove extra whitespace
    document = re.sub(' +', ' ', document)
    document = document.strip()

    return document


pre_process_corpus = np.vectorize(pre_process_document)
```

In [ ]:

```
train_reviews = pre_process_corpus(train_reviews)
val_reviews = pre_process_corpus(val_reviews)
test_reviews = pre_process_corpus(test_reviews)
```

Build Data Ingestion Functions

In [11]:

```
# Training input on the whole training set with no limit on training epochs.
train_input_fn = tf.estimator.inputs.numpy_input_fn(
    {'sentence': train_reviews}, train_sentiments,
    batch_size=256, num_epochs=None, shuffle=True)
```

In [12]:

```
# Prediction on the whole training set.
predict_train_input_fn = tf.estimator.inputs.numpy_input_fn(
    {'sentence': train_reviews}, train_sentiments, shuffle=False)
```

In [13]:

```
# Prediction on the whole validation set.
predict_val_input_fn = tf.estimator.inputs.numpy_input_fn(
    {'sentence': val_reviews}, val_sentiments, shuffle=False)
```

In [14]:

```
# Prediction on the test set.
predict_test_input_fn = tf.estimator.inputs.numpy_input_fn(
    {'sentence': test_reviews}, test_sentiments, shuffle=False)
```

Build Deep Learning Model with Universal Sentence Encoder

In [15]:

```
embedding_feature = hub.text_embedding_column(
    key='sentence',
    module_spec="https://tfhub.dev/google/universal-sentence-encoder/2",
    trainable=False)
```

In [16]:

```
dnn = tf.estimator.DNNClassifier(
          hidden_units=[512, 128],
          feature_columns=[embedding_feature],
          n_classes=2,
          activation_fn=tf.nn.relu,
          dropout=0.1,
          optimizer=tf.train.AdagradOptimizer(learning_rate=0.005))
```

```
INFO:tensorflow:Using default config.

I0510 10:34:30.163194  9676 estimator.py:1739] Using default config.

WARNING:tensorflow:Using temporary folder as model directory: C:\Users\fina
n\AppData\Local\Temp\tmpzoownwot

W0510 10:34:30.165180  9676 estimator.py:1760] Using temporary folder as mo
del directory: C:\Users\finan\AppData\Local\Temp\tmpzoownwot

INFO:tensorflow:Using config: {'_model_dir': 'C:\\Users\\finan\\AppData\\Lo
cal\\Temp\\tmpzoownwot', '_tf_random_seed': None, '_save_summary_steps': 10
0, '_save_checkpoints_steps': None, '_save_checkpoints_secs': 600, '_sessio
n_config': allow_soft_placement: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log
_step_count_steps': 100, '_train_distribute': None, '_device_fn': None, '_p
rotocol': None, '_eval_distribute': None, '_experimental_distribute': None,
'_service': None, '_cluster_spec': <tensorflow.python.training.server_lib.C
lusterSpec object at 0x000002B2903DE278>, '_task_type': 'worker', '_task_i
d': 0, '_global_id_in_cluster': 0, '_master': '', '_evaluation_master': '',
'_is_chief': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}

I0510 10:34:30.167162  9676 estimator.py:201] Using config: {'_model_dir':
'C:\\Users\\finan\\AppData\\Local\\Temp\\tmpzoownwot', '_tf_random_seed': N
one, '_save_summary_steps': 100, '_save_checkpoints_steps': None, '_save_ch
eckpoints_secs': 600, '_session_config': allow_soft_placement: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log
_step_count_steps': 100, '_train_distribute': None, '_device_fn': None, '_p
rotocol': None, '_eval_distribute': None, '_experimental_distribute': None,
'_service': None, '_cluster_spec': <tensorflow.python.training.server_lib.C
lusterSpec object at 0x000002B2903DE278>, '_task_type': 'worker', '_task_i
d': 0, '_global_id_in_cluster': 0, '_master': '', '_evaluation_master': '',
'_is_chief': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}
```

Train for approx 12 epochs

In [17]:

```
256*1500 / 30000
```

Out[17]:

```
12.8
```

Model Training

In [18]:

```
tf.logging.set_verbosity(tf.logging.ERROR)
import time

TOTAL_STEPS = 1500
STEP_SIZE = 100
for step in range(0, TOTAL_STEPS+1, STEP_SIZE):
    print()
    print('-'*100)
    print('Training for step =', step)
    start_time = time.time()
    dnn.train(input_fn=train_input_fn, steps=STEP_SIZE)
    elapsed_time = time.time() - start_time
    print('Train Time (s):', elapsed_time)
    print('Eval Metrics (Train):', dnn.evaluate(input_fn=predict_train_input_
    print('Eval Metrics (Validation):', dnn.evaluate(input_fn=predict_val_inp
```

```
9, 'prediction/mean': 0.5271177, 'recall': 0.8813187, 'global_step': 300}
Eval Metrics (Validation): {'accuracy': 0.845, 'accuracy_baseline': 0.50
5, 'auc': 0.9274982, 'auc_precision_recall': 0.9234167, 'average_loss':
0.3489759, 'label/mean': 0.495, 'loss': 43.621986, 'precision': 0.818590
7, 'prediction/mean': 0.532409, 'recall': 0.88242424, 'global_step': 300}

--------------------------------------------------------------------------
--------------------------
Training for step = 300
Train Time (s): 150.79049634933472
Eval Metrics (Train): {'accuracy': 0.8579, 'accuracy_baseline': 0.5005,
'auc': 0.93498445, 'auc_precision_recall': 0.93492436, 'average_loss': 0.
32765296, 'label/mean': 0.5005, 'loss': 41.828037, 'precision': 0.845190
7, 'prediction/mean': 0.52012026, 'recall': 0.87665665, 'global_step': 40
0}
Eval Metrics (Validation): {'accuracy': 0.849, 'accuracy_baseline': 0.50
5, 'auc': 0.9290105, 'auc_precision_recall': 0.9242683, 'average_loss':
0.34316924, 'label/mean': 0.495, 'loss': 42.896156, 'precision': 0.825264
75, 'prediction/mean': 0.52481365, 'recall': 0.8816162, 'global_step': 40
0}
```

Model Evaluation

In [19]:

```
dnn.evaluate(input_fn=predict_train_input_fn)
```

Out[19]:

```
{'accuracy': 0.88023335,
 'accuracy_baseline': 0.5005,
 'auc': 0.951243,
 'auc_precision_recall': 0.95073926,
 'average_loss': 0.2838174,
 'label/mean': 0.5005,
 'loss': 36.23201,
 'precision': 0.8745409,
 'prediction/mean': 0.5085198,
 'recall': 0.8881119,
 'global_step': 1600}
```

In [20]:

```
dnn.evaluate(input_fn=predict_test_input_fn)
```

Out[20]:

```
{'accuracy': 0.8622,
 'accuracy_baseline': 0.5006667,
 'auc': 0.93942887,
 'auc_precision_recall': 0.93853045,
 'average_loss': 0.31512484,
 'label/mean': 0.5006667,
 'loss': 40.058243,
 'precision': 0.8570117,
 'prediction/mean': 0.50911707,
 'recall': 0.8699068,
 'global_step': 1600}
```

Build a Generic Model Trainer on any Input Sentence Encoder

In [21]:

```
import time

TOTAL_STEPS = 1500
STEP_SIZE = 500

my_checkpointing_config = tf.estimator.RunConfig(
    keep_checkpoint_max = 2,       # Retain the 2 most recent checkpoints.
)

def train_and_evaluate_with_sentence_encoder(hub_module, train_module=False,
    embedding_feature = hub.text_embedding_column(
        key='sentence', module_spec=hub_module, trainable=train_module)

    print()
    print('='*100)
    print('Training with', hub_module)
    print('Trainable is:', train_module)
    print('='*100)

    dnn = tf.estimator.DNNClassifier(
            hidden_units=[512, 128],
            feature_columns=[embedding_feature],
            n_classes=2,
            activation_fn=tf.nn.relu,
            dropout=0.1,
            optimizer=tf.train.AdagradOptimizer(learning_rate=0.005),
            model_dir=path,
            config=my_checkpointing_config)

    for step in range(0, TOTAL_STEPS+1, STEP_SIZE):
        print('-'*100)
        print('Training for step =', step)
        start_time = time.time()
        dnn.train(input_fn=train_input_fn, steps=STEP_SIZE)
        elapsed_time = time.time() - start_time
        print('Train Time (s):', elapsed_time)
        print('Eval Metrics (Train):', dnn.evaluate(input_fn=predict_train_in
        print('Eval Metrics (Validation):', dnn.evaluate(input_fn=predict_val

    train_eval_result = dnn.evaluate(input_fn=predict_train_input_fn)
    test_eval_result = dnn.evaluate(input_fn=predict_test_input_fn)

    return {
      "Model Dir": dnn.model_dir,
      "Training Accuracy": train_eval_result["accuracy"],
      "Test Accuracy": test_eval_result["accuracy"],
      "Training AUC": train_eval_result["auc"],
      "Test AUC": test_eval_result["auc"],
      "Training Precision": train_eval_result["precision"],
      "Test Precision": test_eval_result["precision"],
      "Training Recall": train_eval_result["recall"],
      "Test Recall": test_eval_result["recall"]
    }
```

Train Deep Learning Models on difference Sentence Encoders

- NNLM - pre-trained and fine-tuning
- USE - pre-trained and fine-tuning

In [22]:

```
tf.logging.set_verbosity(tf.logging.ERROR)

results = {}

results["nnlm-en-dim128"] = train_and_evaluate_with_sentence_encoder(
    "https://tfhub.dev/google/nnlm-en-dim128/1", path='/storage/models/nnlm-e

results["nnlm-en-dim128-with-training"] = train_and_evaluate_with_sentence_en
    "https://tfhub.dev/google/nnlm-en-dim128/1", train_module=True, path='/st

results["use-512"] = train_and_evaluate_with_sentence_encoder(
    "https://tfhub.dev/google/universal-sentence-encoder/2", path='/storage/m

results["use-512-with-training"] = train_and_evaluate_with_sentence_encoder(
    "https://tfhub.dev/google/universal-sentence-encoder/2", train_module=Tru
```

```
============================================================================
=========================
Training with https://tfhub.dev/google/nnlm-en-dim128/1 (https://tfhub.dev/
google/nnlm-en-dim128/1)
Trainable is: False
============================================================================
=========================
----------------------------------------------------------------------------
-------------------------
Training for step = 0
Train Time (s): 51.826966524124146
Eval Metrics (Train): {'accuracy': 0.80146664, 'accuracy_baseline': 0.5005,
'auc': 0.8842025, 'auc_precision_recall': 0.8850704, 'average_loss': 0.4282
3544, 'label/mean': 0.5005, 'loss': 54.668354, 'precision': 0.79789543, 'pr
ediction/mean': 0.5084689, 'recall': 0.807992, 'global_step': 500}
Eval Metrics (Validation): {'accuracy': 0.8008, 'accuracy_baseline': 0.505,
'auc': 0.8791525, 'auc_precision_recall': 0.8745084, 'average_loss': 0.4373
9897, 'label/mean': 0.495, 'loss': 54.674873, 'precision': 0.7912564, 'pred
iction/mean': 0.50845456, 'recall': 0.81171715, 'global_step': 500}
----------------------------------------------------------------------------
-------------------------
Training for step = 500
Train Time (s): 50.849717140197754
Eval Metrics (Train): {'accuracy': 0.80916667, 'accuracy_baseline': 0.5005,
'auc': 0.892072, 'auc_precision_recall': 0.8928739, 'average_loss': 0.41402
233, 'label/mean': 0.5005, 'loss': 52.853916, 'precision': 0.8115359, 'pred
iction/mean': 0.50149363, 'recall': 0.8058608, 'global_step': 1000}
Eval Metrics (Validation): {'accuracy': 0.7998, 'accuracy_baseline': 0.505,
'auc': 0.8818506, 'auc_precision_recall': 0.87735856, 'average_loss': 0.432
13424, 'label/mean': 0.495, 'loss': 54.016777, 'precision': 0.79409415, 'pr
ediction/mean': 0.50118107, 'recall': 0.80404043, 'global_step': 1000}
----------------------------------------------------------------------------
-------------------------
Training for step = 1000
Train Time (s): 50.59104132652283
Eval Metrics (Train): {'accuracy': 0.8148, 'accuracy_baseline': 0.5005, 'au
c': 0.8995086, 'auc_precision_recall': 0.9002718, 'average_loss': 0.4025783
2, 'label/mean': 0.5005, 'loss': 51.39298, 'precision': 0.8035818, 'predict
ion/mean': 0.5192918, 'recall': 0.8337662, 'global_step': 1500}
```

```
Eval Metrics (Validation): {'accuracy': 0.8008, 'accuracy_baseline': 0.505,
'auc': 0.88397235, 'auc_precision_recall': 0.8808539, 'average_loss': 0.429
79467, 'label/mean': 0.495, 'loss': 53.724335, 'precision': 0.78475165, 'pr
ediction/mean': 0.5190896, 'recall': 0.82343435, 'global_step': 1500}
---------------------------------------------------------------------------
-------------------------
Training for step = 1500
Train Time (s): 50.57778525352478
Eval Metrics (Train): {'accuracy': 0.82266665, 'accuracy_baseline': 0.5005,
'auc': 0.9059942, 'auc_precision_recall': 0.9066139, 'average_loss': 0.3885
194, 'label/mean': 0.5005, 'loss': 49.59822, 'precision': 0.82951534, 'pred
iction/mean': 0.4958161, 'recall': 0.8127206, 'global_step': 2000}
Eval Metrics (Validation): {'accuracy': 0.8012, 'accuracy_baseline': 0.505,
'auc': 0.88518846, 'auc_precision_recall': 0.88195205, 'average_loss': 0.42
649552, 'label/mean': 0.495, 'loss': 53.31194, 'precision': 0.8013838, 'pre
diction/mean': 0.4948716, 'recall': 0.79555553, 'global_step': 2000}

===========================================================================
=========================
Training with https://tfhub.dev/google/nnlm-en-dim128/1 (https://tfhub.dev/
google/nnlm-en-dim128/1)
Trainable is: True
===========================================================================
=========================
---------------------------------------------------------------------------
-------------------------
Training for step = 0
Train Time (s): 64.90571737289429
Eval Metrics (Train): {'accuracy': 0.9589, 'accuracy_baseline': 0.5005, 'au
c': 0.98980254, 'auc_precision_recall': 0.9897486, 'average_loss': 0.130429
73, 'label/mean': 0.5005, 'loss': 16.650604, 'precision': 0.97134066, 'pred
iction/mean': 0.48233554, 'recall': 0.94578755, 'global_step': 500}
Eval Metrics (Validation): {'accuracy': 0.875, 'accuracy_baseline': 0.505,
 'auc': 0.9476086, 'auc_precision_recall': 0.9472497, 'average_loss': 0.312
72167, 'label/mean': 0.495, 'loss': 39.09021, 'precision': 0.88381743, 'pre
diction/mean': 0.48048687, 'recall': 0.8606061, 'global_step': 500}
---------------------------------------------------------------------------
-------------------------
Training for step = 500
Train Time (s): 63.79932928085327
Eval Metrics (Train): {'accuracy': 0.99476665, 'accuracy_baseline': 0.5005,
'auc': 0.9982084, 'auc_precision_recall': 0.99844116, 'average_loss': 0.035
162635, 'label/mean': 0.5005, 'loss': 4.488847, 'precision': 0.9949367, 'pr
ediction/mean': 0.50185156, 'recall': 0.9946054, 'global_step': 1000}
Eval Metrics (Validation): {'accuracy': 0.8718, 'accuracy_baseline': 0.505,
'auc': 0.9367766, 'auc_precision_recall': 0.93749404, 'average_loss': 0.443
9645, 'label/mean': 0.495, 'loss': 55.495564, 'precision': 0.8636003, 'pred
iction/mean': 0.50576794, 'recall': 0.88, 'global_step': 1000}
---------------------------------------------------------------------------
-------------------------
Training for step = 1000
Train Time (s): 64.02167415618896
Eval Metrics (Train): {'accuracy': 0.99913335, 'accuracy_baseline': 0.5005,
'auc': 0.99956274, 'auc_precision_recall': 0.9997183, 'average_loss': 0.009
2969285, 'label/mean': 0.5005, 'loss': 1.186842, 'precision': 0.99960005,
 'prediction/mean': 0.49952236, 'recall': 0.998668, 'global_step': 1500}
Eval Metrics (Validation): {'accuracy': 0.8648, 'accuracy_baseline': 0.505,
'auc': 0.92224383, 'auc_precision_recall': 0.92588776, 'average_loss': 0.60
62944, 'label/mean': 0.495, 'loss': 75.7868, 'precision': 0.86168075, 'pred
iction/mean': 0.49812028, 'recall': 0.8658586, 'global_step': 1500}
---------------------------------------------------------------------------
-------------------------
Training for step = 1500
Train Time (s): 70.62114715576172
Eval Metrics (Train): {'accuracy': 0.99976665, 'accuracy_baseline': 0.5005,
'auc': 0.9998468, 'auc_precision_recall': 0.99991536, 'average_loss': 0.003
5679955, 'label/mean': 0.5005, 'loss': 0.45548877, 'precision': 1.0, 'predi
ction/mean': 0.5001454, 'recall': 0.9995338, 'global_step': 2000}
Eval Metrics (Validation): {'accuracy': 0.8632, 'accuracy_baseline': 0.505,
'auc': 0.9133162, 'auc_precision_recall': 0.9201185, 'average_loss': 0.7219
939, 'label/mean': 0.495, 'loss': 90.249245, 'precision': 0.8580568, 'predi
ction/mean': 0.49972987, 'recall': 0.86707073, 'global_step': 2000}

===========================================================================
=========================
Training with https://tfhub.dev/google/universal-sentence-encoder/2 (http
s://tfhub.dev/google/universal-sentence-encoder/2)
Trainable is: False
===========================================================================
=========================
---------------------------------------------------------------------------
-------------------------
Training for step = 0
Train Time (s): 302.2654027938843
Eval Metrics (Train): {'accuracy': 0.85943335, 'accuracy_baseline': 0.5005,
'auc': 0.936284, 'auc_precision_recall': 0.9362018, 'average_loss': 0.32272
59, 'label/mean': 0.5005, 'loss': 41.199047, 'precision': 0.8643542, 'predi
ction/mean': 0.49555996, 'recall': 0.85301363, 'global_step': 500}
Eval Metrics (Validation): {'accuracy': 0.8536, 'accuracy_baseline': 0.505,
'auc': 0.9300823, 'auc_precision_recall': 0.9257036, 'average_loss': 0.3369
7137, 'label/mean': 0.495, 'loss': 42.12142, 'precision': 0.8484606, 'predi
ction/mean': 0.4998482, 'recall': 0.8573737, 'global_step': 500}
---------------------------------------------------------------------------
-------------------------
Training for step = 500
Train Time (s): 305.69727897644043
Eval Metrics (Train): {'accuracy': 0.8656, 'accuracy_baseline': 0.5005, 'au
c': 0.94388026, 'auc_precision_recall': 0.94367194, 'average_loss': 0.30855
84, 'label/mean': 0.5005, 'loss': 39.390434, 'precision': 0.891942, 'predic
tion/mean': 0.4698603, 'recall': 0.832301, 'global_step': 1000}
Eval Metrics (Validation): {'accuracy': 0.8508, 'accuracy_baseline': 0.505,
'auc': 0.9339114, 'auc_precision_recall': 0.92933506, 'average_loss': 0.330
38926, 'label/mean': 0.495, 'loss': 41.298656, 'precision': 0.8649219, 'pre
diction/mean': 0.4724691, 'recall': 0.8278788, 'global_step': 1000}
---------------------------------------------------------------------------
-------------------------
Training for step = 1000

Train Time (s): 316.0776982307434
Eval Metrics (Train): {'accuracy': 0.88, 'accuracy_baseline': 0.5005, 'au
c': 0.9505916, 'auc_precision_recall': 0.9502528, 'average_loss': 0.28600
14, 'label/mean': 0.5005, 'loss': 36.51082, 'precision': 0.8793115, 'pred
iction/mean': 0.50179684, 'recall': 0.8811855, 'global_step': 1500}
Eval Metrics (Validation): {'accuracy': 0.8576, 'accuracy_baseline': 0.50
```

```
5, 'auc': 0.93670297, 'auc_precision_recall': 0.9329308, 'average_loss':
0.3213528, 'label/mean': 0.495, 'loss': 40.1691, 'precision': 0.84636545,
'prediction/mean': 0.5047452, 'recall': 0.87030303, 'global_step': 1500}
------------------------------------------------------------------------
--------------------------
Training for step = 1500
Train Time (s): 327.14475274086
Eval Metrics (Train): {'accuracy': 0.88566667, 'accuracy_baseline': 0.500
5, 'auc': 0.95566285, 'auc_precision_recall': 0.9551946, 'average_loss':
0.27226546, 'label/mean': 0.5005, 'loss': 34.757294, 'precision': 0.89962
053, 'prediction/mean': 0.48445228, 'recall': 0.8684649, 'global_step': 2
000}
Eval Metrics (Validation): {'accuracy': 0.8594, 'accuracy_baseline': 0.50
5, 'auc': 0.93832415, 'auc_precision_recall': 0.9346671, 'average_loss':
0.31797662, 'label/mean': 0.495, 'loss': 39.747078, 'precision': 0.862224
04, 'prediction/mean': 0.48683268, 'recall': 0.85212123, 'global_step': 2
000}

========================================================================
==========================
Training with https://tfhub.dev/google/universal-sentence-encoder/2 (http
s://tfhub.dev/google/universal-sentence-encoder/2)
Trainable is: True
========================================================================
==========================
------------------------------------------------------------------------
--------------------------
Training for step = 0
Train Time (s): 491.92156052589417
Eval Metrics (Train): {'accuracy': 0.9992333, 'accuracy_baseline': 0.500
5, 'auc': 0.9996782, 'auc_precision_recall': 0.99972016, 'average_loss':
 0.004658407, 'label/mean': 0.5005, 'loss': 0.59469026, 'precision': 0.99
913436, 'prediction/mean': 0.50085956, 'recall': 0.999334, 'global_step':
500}
Eval Metrics (Validation): {'accuracy': 0.902, 'accuracy_baseline': 0.50
5, 'auc': 0.95116436, 'auc_precision_recall': 0.95380425, 'average_loss':
0.4201498, 'label/mean': 0.495, 'loss': 52.518726, 'precision': 0.889063
1, 'prediction/mean': 0.5109552, 'recall': 0.91636366, 'global_step': 50
0}
------------------------------------------------------------------------
--------------------------
Training for step = 500
Train Time (s): 453.92580556869507
Eval Metrics (Train): {'accuracy': 0.99993336, 'accuracy_baseline': 0.500
5, 'auc': 0.9999666, 'auc_precision_recall': 0.99996656, 'average_loss':
 0.0005400647, 'label/mean': 0.5005, 'loss': 0.068944424, 'precision': 0.
9999334, 'prediction/mean': 0.50049335, 'recall': 0.9999334, 'global_ste
p': 1000}
Eval Metrics (Validation): {'accuracy': 0.902, 'accuracy_baseline': 0.50
5, 'auc': 0.93404496, 'auc_precision_recall': 0.94373906, 'average_loss':
0.5568309, 'label/mean': 0.495, 'loss': 69.60386, 'precision': 0.8968413,
'prediction/mean': 0.50135154, 'recall': 0.90626264, 'global_step': 1000}
------------------------------------------------------------------------
--------------------------
Training for step = 1000
Train Time (s): 492.0408351421356
Eval Metrics (Train): {'accuracy': 0.9999667, 'accuracy_baseline': 0.500
```

```
5, 'auc': 1.0, 'auc_precision_recall': 1.0, 'average_loss': 0.0002376532,
'label/mean': 0.5005, 'loss': 0.030338706, 'precision': 0.9999334, 'predi
ction/mean': 0.50052303, 'recall': 1.0, 'global_step': 1500}
Eval Metrics (Validation): {'accuracy': 0.901, 'accuracy_baseline': 0.50
5, 'auc': 0.9300043, 'auc_precision_recall': 0.9413148, 'average_loss':
 0.61009365, 'label/mean': 0.495, 'loss': 76.2617, 'precision': 0.895051
9, 'prediction/mean': 0.5020793, 'recall': 0.90626264, 'global_step': 150
0}
--------------------------------------------------------------------------
---------------------------
Training for step = 1500
Train Time (s): 463.3303544521332
Eval Metrics (Train): {'accuracy': 1.0, 'accuracy_baseline': 0.5005, 'au
c': 1.0, 'auc_precision_recall': 1.0, 'average_loss': 4.684698e-05, 'labe
l/mean': 0.5005, 'loss': 0.0059804656, 'precision': 1.0, 'prediction/mea
n': 0.5004949, 'recall': 1.0, 'global_step': 2000}
Eval Metrics (Validation): {'accuracy': 0.8994, 'accuracy_baseline': 0.50
5, 'auc': 0.92747986, 'auc_precision_recall': 0.9392976, 'average_loss':
 0.6486424, 'label/mean': 0.495, 'loss': 81.08031, 'precision': 0.893141
9, 'prediction/mean': 0.5022174, 'recall': 0.9050505, 'global_step': 200
0}
```

Model Evaluations

In [23]:

```
results_df = pd.DataFrame.from_dict(results, orient="index")
results_df
```

Out[23]:

	Model Dir	Training Accuracy	Test Accuracy	Training AUC	Test AUC	Training Precision	Test Precision
nnlm-en-dim128	/storage/models/nnlm-en-dim128_f/	0.822667	0.807467	0.905994	0.888478	0.829515	0.815452
nnlm-en-dim128-with-training	/storage/models/nnlm-en-dim128_t/	0.999767	0.868200	0.999847	0.919358	1.000000	0.871592
use-512	/storage/models/use-512_f/	0.885667	0.862067	0.955663	0.940320	0.899621	0.876540
use-512-with-training	/storage/models/use-512_t/	1.000000	0.904933	1.000000	0.930509	1.000000	0.901425

In [24]:

```
best_model_dir = results_df[results_df['Test Accuracy'] == results_df['Test A
best_model_dir
```

Out[24]:

```
'/storage/models/use-512_t/'
```

In [25]:

```
embedding_feature = hub.text_embedding_column(
        key='sentence', module_spec="https://tfhub.dev/google/universal-sente

dnn = tf.estimator.DNNClassifier(
            hidden_units=[512, 128],
            feature_columns=[embedding_feature],
            n_classes=2,
            activation_fn=tf.nn.relu,
            dropout=0.1,
            optimizer=tf.train.AdagradOptimizer(learning_rate=0.005),
            model_dir=best_model_dir)
dnn
```

Out[25]:

```
<tensorflow_estimator.python.estimator.canned.dnn.DNNClassifier at 0x2b3b7d
90128>
```

In [26]:

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

In [27]:

```
predictions = get_predictions(estimator=dnn, input_fn=predict_test_input_fn)
predictions[:10]
```

Out[27]:

```
[0, 1, 0, 1, 1, 0, 1, 1, 1, 1]
```

In [28]:

```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\finan\appdata\local\prog
rams\python\python37\lib\site-packages (0.9.0)
Requirement already satisfied: pandas>=0.15.2 in c:\users\finan\appdata\loc
al\programs\python\python37\lib\site-packages (from seaborn) (0.24.2)
Requirement already satisfied: scipy>=0.14.0 in c:\users\finan\appdata\loca
l\programs\python\python37\lib\site-packages (from seaborn) (1.2.1)
Requirement already satisfied: numpy>=1.9.3 in c:\users\finan\appdata\local
\programs\python\python37\lib\site-packages (from seaborn) (1.16.2)
Requirement already satisfied: matplotlib>=1.4.3 in c:\users\finan\appdata
\local\programs\python\python37\lib\site-packages (from seaborn) (3.0.3)
Requirement already satisfied: python-dateutil>=2.5.0 in c:\users\finan\app
data\local\programs\python\python37\lib\site-packages (from pandas>=0.15.2-
>seaborn) (2.8.0)
Requirement already satisfied: pytz>=2011k in c:\users\finan\appdata\local
\programs\python\python37\lib\site-packages (from pandas>=0.15.2->seaborn)
(2018.9)
Requirement already satisfied: cycler>=0.10 in c:\users\finan\appdata\local
\programs\python\python37\lib\site-packages (from matplotlib>=1.4.3->seabor
n) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\finan\appdata\local\programs\python\python37\lib\site-packages (fr
om matplotlib>=1.4.3->seaborn) (2.3.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\finan\appdata
\local\programs\python\python37\lib\site-packages (from matplotlib>=1.4.3->
seaborn) (1.0.1)
Requirement already satisfied: six>=1.5 in c:\users\finan\appdata\local\pro
grams\python\python37\lib\site-packages (from python-dateutil>=2.5.0->panda
s>=0.15.2->seaborn) (1.12.0)
Requirement already satisfied: setuptools in c:\users\finan\appdata\local\p
rograms\python\python37\lib\site-packages (from kiwisolver>=1.0.1->matplotl
ib>=1.4.3->seaborn) (41.0.0)

WARNING: You are using pip version 19.1, however version 19.1.1 is availabl
e.
You should consider upgrading via the 'python -m pip install --upgrade pip'
command.
```

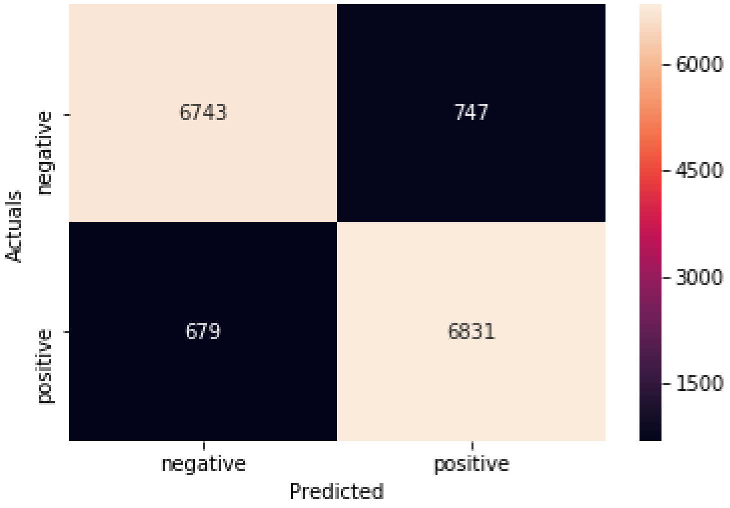
In [29]:

```
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline

with tf.Session() as session:
    cm = tf.confusion_matrix(test_sentiments, predictions).eval()

LABELS = ['negative', 'positive']
sns.heatmap(cm, annot=True, xticklabels=LABELS, yticklabels=LABELS, fmt='g')
xl = plt.xlabel("Predicted")
yl = plt.ylabel("Actuals")
```



In [30]:

```
from sklearn.metrics import classification_report

print(classification_report(y_true=test_sentiments, y_pred=predictions, targe
```

```
              precision    recall  f1-score   support

    negative       0.91      0.90      0.90      7490
    positive       0.90      0.91      0.91      7510

   micro avg       0.90      0.90      0.90     15000
   macro avg       0.90      0.90      0.90     15000
weighted avg       0.90      0.90      0.90     15000
```

In [ ]: