Goal: Explore Transformers library by HuggingFace for deep learning model creation and Sentimer

Dataset:https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews

```
import pandas as pd
import numpy as np
import torch
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473

Enter your authorization code:
··········
Mounted at /content/drive
```

```
import os
os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"]="1"
device = 'cuda' if torch.cuda.is_available() else 'cpu'
```

```
dataset = pd.read_csv("/content/drive/My Drive/pytorch_tutorials/PyTorch Sentiment Analysi
```

```
dataset.head()
```

	Unnamed: 0	Clothing ID	Age	Title	Review Text	Ratin
0	0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	
1	1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	
2	2	1077	60	Some major design flaws	I had such high hopes for this dress and reall...	
3	3	1049	50	My favorite buy!	I love, love, love this jumpsuit. it's fun, fl...	
4	4	847	47	Flattering shirt	This shirt is very flattering to all due to th...	

As we are concerned about only text data analysis,we'll ignore all other columns and keep 'Review

```
dataset = dataset[['Review Text','Recommended IND']]
```

```
dataset =  dataset.rename(columns={'Review Text':'review','Recommended IND':'recommended'}
```

```
dataset.head()
dataset.head()
```

	review	recommended
0	Absolutely wonderful - silky and sexy and comf...	1
1	Love this dress! it's sooo pretty. i happene...	1
2	I had such high hopes for this dress and reall...	0
3	I love, love, love this jumpsuit. it's fun, fl...	1
4	This shirt is very flattering to all due to th...	1

```
dataset.shape
```

```
(23486, 2)
```

```
dataset.review.isnull().sum()
```

```
845
```

```
dataset = dataset.dropna(axis=0, subset=['review'])
```

```
dataset.recommended.value_counts()
```

```
1    18540
0     4101
Name: recommended, dtype: int64
```

```
'''Since dataset is huge to run on Google colab,let's take only few samples for our tutori
dataset = dataset.iloc[:2000,:]
```

To learn about BERT and different language models:
http://jalammar.github.io/illustrated-transformer/
http://jalammar.github.io/illustrated-bert/
This blog is the best explaination you could find on the internet.

Transformers library by HuggingFace provides many pretrained language models which can be fur
More info: https://github.com/huggingface/transformers

Now we'll do sentiment analysis/senetence classification in following 2 steps:

1. Load Pretrained DistilBert model architecture and fine-tune it further using logistic regressio
2. Load Pretrained DistilBert classification class and fine-tune it further using PyTorch

Difference between 1.DistilBert model architecure and 2.DistilBert classification class:
DistilBert model architecure provides last hidden states from the model.These last hidden states c
further.For example we are going to use them in logistic regression.
DistilBert classification class uses these last hidden state and initialises weights for classification
tuned further.

1. Load Pretrained DistilBert model architecture

Let's first install transformers library

```
!pip install transformers
```

```
Collecting transformers
  Downloading https://files.pythonhosted.org/packages/a3/78/92cedda05552398352ed97849
     |████████████████████████████████| 573kB 3.3MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.6/dist-packages (fr
Collecting sacremoses
  Downloading https://files.pythonhosted.org/packages/99/50/93509f906a40bffd7d175f97f
     |████████████████████████████████| 890kB 41.7MB/s
Collecting sentencepiece
  Downloading https://files.pythonhosted.org/packages/74/f4/2d5214cbf13d06e7cb2c20d84
     |████████████████████████████████| 1.0MB 41.7MB/s
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: boto3 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: dataclasses; python_version < "3.7" in /usr/local/lib/
Collecting tokenizers==0.5.2
  Downloading https://files.pythonhosted.org/packages/d1/3f/73c881ea4723e43c1e9acf317
     |████████████████████████████████| 3.7MB 33.6MB/s
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from sa
Requirement already satisfied: click in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: joblib in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: botocore<1.16.0,>=1.15.38 in /usr/local/lib/python3.6/
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /usr/local/lib/python3.6/dis
Requirement already satisfied: s3transfer<0.4.0,>=0.3.0 in /usr/local/lib/python3.6/d
Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.6/dist-
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.
Building wheels for collected packages: sacremoses
  Building wheel for sacremoses (setup.py) ... done
  Created wheel for sacremoses: filename=sacremoses-0.0.41-cp36-none-any.whl size=893
  Stored in directory: /root/.cache/pip/wheels/22/5a/d4/b020a81249de7dc63758a34222fea
Successfully built sacremoses
Installing collected packages: sacremoses, sentencepiece, tokenizers, transformers
Successfully installed sacremoses-0.0.41 sentencepiece-0.1.85 tokenizers-0.5.2 transf
```

```
from transformers import DistilBertModel,DistilBertTokenizer
```

```
model = DistilBertModel.from_pretrained('distilbert-base-uncased')
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
```

```
Downloading: 100%                    546/546 [00:00<00:00, 1.59kB/s]

Downloading: 100%                    268M/268M [00:07<00:00, 38.3MB/s]

Downloading: 100%                    232k/232k [00:00<00:00, 1.58MB/s]
```

```
#Tokenization:Convert words in 'review' column to numbers
tokenized_reviews = dataset.review.apply(lambda x: tokenizer.encode(x,add_special_tokens=T
```

```
#Padding:To make all sentences of same length.This is only required for batch creation
'''First we need to find maximum length of senetence/review.'''
max_len = max(map(len,tokenized_reviews))
print(max_len)
```

```
148
```

```
np.array(tokenized_reviews).shape
```

```
(2000,)
```

```
padded_reviews = np.array([ i+[0]*(max_len-len(i))  for i in tokenized_reviews])
```

```
np.array(padded_reviews).shape
```

```
(2000, 148)
```

```
padded_reviews[0]
```

```
array([  101,  7078,  6919,  1011, 18848,  1998,  7916,  1998,  6625,
         102,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0])
```

```
#Masking:To tell DistilBert to ignore padding. This is called attention masking.Basically
attention_masked_reviews = np.where(padded_reviews!=0,1,0)
```

```
np.array(attention_masked_reviews).shape
np.array(attention_masked_reviews).shape
```

```
(2000, 148)
```

```
attention_masked_reviews[0]
```

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

```
# Get last hidden states
input_ids = torch.tensor(padded_reviews).to(device)
attention_mask = torch.tensor(attention_masked_reviews).to(device)

with torch.no_grad():
  last_hidden_states = model(input_ids,attention_mask=attention_mask)
```

But we don't need all hidden states.We only need last hidden state of first token 'CLS' of each sent
purpose.

```
print(type(last_hidden_states))
```

```
<class 'tuple'>
```

```
last_hidden_states[0].shape
```

```
torch.Size([2000, 148, 768])
```

```
'''Dimension of hidden state axbxc
a = number of reviews  2000
b = number of tokens  148
c = number of hidden units  768 '''
X = last_hidden_states[0][:,0,:].numpy()
y = dataset.recommended


print(X.shape)
print(y.shape)
```

```
(2000, 768)
(2000,)
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
#Logistic Regression
X_train,X_test,y_train,y_test = train_test_split(X,y)
log_model = LogisticRegression(max_iter=1500)
log_model.fit(X_train,y_train)
preds = log_model.predict(X_test)
from sklearn import metrics
```

```
print(metrics.roc_auc_score(y_test, preds))
```

```
0.7583053910727451
```

2.Load Pretrained DistilBert classification class

```
from transformers import DistilBertForSequenceClassification,DistilBertTokenizer
model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased',num_
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
```

```
tokenized_reviews = dataset.review.apply(lambda x: tokenizer.encode(x,add_special_tokens=T
max_len = max(map(len,tokenized_reviews))
padded_reviews = np.array([ i+[0]*(max_len-len(i))  for i in tokenized_reviews])
attention_masked_reviews = np.where(padded_reviews!=0,1,0)
```

```
#Dataset preparation
from torch.utils.data import Dataset, TensorDataset,DataLoader
from sklearn.model_selection import train_test_split

X = torch.tensor(padded_reviews)
X_attention = torch.tensor(attention_masked_reviews)
#y = torch.tensor(np.array(dataset.recommended.values))
y = torch.tensor(np.array(dataset.recommended.values)[:,np.newaxis], dtype=torch.float32)

X_train,X_test,y_train,y_test = X[:1500],X[1500:],y[:1500],y[1500:]
X_train_attention,X_test_attention = X_attention[:1500],X_attention[1500:]

train_data = TensorDataset(X_train, X_train_attention, y_train)

train_loader = DataLoader(train_data,batch_size=16, shuffle=True)
```

```
y_train.shape
```

```
torch.Size([1500, 1])
```

```
#Model training
NUM_EPOCHS = 1
LEARNING_RATE = 0.01
optimizer =torch.optim.SGD(model.parameters(), lr=LEARNING_RATE)
loss_fn = torch.nn.BCEWithLogitsLoss()

for i in range(NUM_EPOCHS):
  model.train()
  for X_batch,X_attention_batch,y_batch in train_loader:
    output = model(X_batch,attention_mask=X_attention_batch,labels=None)
    y_pred = output[0]
    #print(y_pred)
    loss = loss_fn(y_pred,y_batch)
    loss.backward()
```

```
        optimizer.step()
        optimizer.zero_grad()



#Evaluation
test_dataset = TensorDataset(X_test, X_test_attention)
test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False)

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

preds = np.zeros([len(test_dataset), 1])
model.eval()
for i, (x_batch, x_mask) in enumerate(test_loader):
    outputs = model(x_batch.to(device),
                    attention_mask=x_mask.to(device))
    y_pred = sigmoid(outputs[0].detach().cpu().numpy())
    preds[i*16:(i+1)*16, :] = y_pred

from sklearn import metrics
print(metrics.roc_auc_score(y_test, preds))
```

```
0.8644878078114195
```